FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Object tracking using 3D point clouds and RGB images for autonomous driving

Daniel Ferreira Brandão

Mestrado em Engenharia Informática e Computação

Supervisor: Luís Filipe Teixeira

July 29, 2022

Object tracking using 3D point clouds and RGB images for autonomous driving

Daniel Ferreira Brandão

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. João Paulo Fernandes
External Examiner: Luis Rosado
Supervisor: Prof. Luís Filipe Teixeira

July 29, 2022

Abstract

Autonomous driving development has been a strong focus for the automotive industry in the past years. Vehicles with autonomous capabilities make streets safer since they are less susceptible to traffic accidents caused by human errors. Object tracking is a necessary component of an autonomous driving system, allowing it to be aware of other traffic participants. Since other high-level autonomous driving modules rely on tracking to make decisions, it is imperative that the tracking is done accurately. Autonomous vehicles are often equipped with sensors and devices that capture 3D information about their surroundings. 3D point clouds captured by LiDARs are commonly used for 3D object detection and tracking. LiDAR sensors measure the time-of-flight of their emitted lasers, working independently of illumination. However, the data from this sensor is sparse, has a low range, and is often noisy. In order to attenuate these problems, some 3D object tracking frameworks fuse 3D point cloud data with RGB images since cameras have a dense and rich visual signal, performing better in distant object tracking. Furthermore, fusing information from multiple sensors might reduce the impact caused by occlusions in one of the sensors.

This project analyzes and evaluates a State-of-the-Art sensor fusion tracking framework, using multiple 2D and 3D object detectors. Moreover, we perform an analysis on how sensor fusion mitigates the tracking performance reduction caused by occlusions.

The results obtained show that the combination of Point-GNN 3D detector and the 2D detections obtained from TrackRCNN outperform every other setup in the tracking task. The sensor fusion occlusion analysis demonstrates that sensor fusion significantly improves the tracking performance when one of the sensors is occluded.

Keywords: Autonomous Driving, 3D Object Tracking, Computer Vision, Machine Learning

Resumo

O desenvolvimento de condução autónoma tem sido um forte foco para a indústria automóvel nos últimos anos. Os veículos com capacidades autónomas tornam as ruas mais seguras, uma vez que são menos susceptíveis a acidentes de viação causados por erros humanos. Object Tracking é uma componente necessária de um sistema de condução autónomo, permitindo-lhe estar ciente dos outros participantes no trânsito. Uma vez que outros módulos de condução autónoma de alto nível dependem do Tracking para tomar decisões, é imperativo que o tracking seja feito com precisão. Os veículos autónomos são frequentemente equipados com sensores e dispositivos que captam a informação 3D do seu ambiente. As 3D Point Clouds capturadas pelos LiDARs são normalmente utilizadas para a detecção e tracking de objectos 3D. Os sensores LiDAR medem o tempo de voo dos seus lasers emitidos, trabalhando independentemente da iluminação. No entanto, os dados deste sensor são escassos, têm um alcance baixo, e são frequentemente ruidosos.A fim de atenuar estes problemas, algumas estruturas de 3D Object Tracking fundem dados de 3D Point Clouds com imagens RGB, uma vez que as câmaras têm um sinal visual denso e rico, tendo um melhor desempenho no rastreio de objectos distantes. Além disso, a fusão de informação de múltiplos sensores pode reduzir o impacto causado por oclusões num dos sensores.

Este projecto analisa e avalia uma framework de tracking com fusão de sensores de última geração, utilizando múltiplos detectores de objectos 2D e 3D. Além disso, efectuamos uma análise sobre como a fusão de sensores atenua a redução do desempenho do tracking causado por oclusões.

Os resultados obtidos mostram que a combinação do detector 3D Point-GNN e as detecções 2D obtidas a partir do TrackRCNN superam todas as outras configurações na tarefa de tracking. A análise de oclusão por fusão de sensores demonstra que a fusão de sensores melhora significativamente o desempenho do tracking quando um dos sensores é ocluído.

Keywords: Autonomous Driving, 3D Object Tracking, Computer Vision, Machine Learning

Acknowledgements

I would like to thank my supervisor, Dr. Luís Filipe Teixeira, for his advice, supervision and availability.

I am also extremely grateful to my family and friends for their unwavering support.

This work is supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 047264; Funding Reference: POCI-01-0247-FEDER-047264]

Daniel Ferreira Brandão

"The best way to predict the future is to implement it."

David H. Hansson

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
DL	Deep Learning
DNN	Deep Neural Networks
ETE	End-to-End
FPS	Frames-per-Second
GNN	Graph Neural Network
LiDAR	Light Detection and Ranging
ML	Machine Learning
MLP	Multi Layer Perceptron
MOT	Multi-Oject Tracking
RADAR	Radio Detection and Ranging
TBT	Tracking-By-Detection

Chapter 1

Introduction

According to National Highway Traffic Safety Administration (NHTSA), an estimated 94% of road accidents happen due to human error [40]. Furthermore, Thomas et al. [42] reported that, in Europe, more than 50% road accidents are associated with timing errors of the driver. These types of road accidents make for a considerable portion of the estimated 1.35 million people worldwide that die due to traffic accidents each year. Autonomous driving systems intend to provide a safer road experience by developing autonomous driving systems that replace parts of the driving activity, preventing human failures. In recent years, much progress in autonomous driving systems has been made, thanks to the improvements in Artificial Intelligence and Deep Learning technologies. In order to be fit for the context of autonomous driving, these systems must be accurate and efficient, making their development a challenging task.

1.1 Context

The automotive industry has been researching and developing perceptive systems in recent years, aiming to equip vehicles with autonomous driving systems. There is a great urge to improve these systems with the main objectives of making the streets safer and providing a more liberating road experience for the driver. The end goal of this ongoing research is achieving full automation in autonomous driving, making the human driver a mere passenger. To classify the levels of automation of an automation system, a standard adopted by SAE is used. There are six levels of automation, ranging from 0 to 5, whereas level 0 refers to a vehicle without driving automation and level 5 to full automation. Figure 1.1 shows the mentioned levels. In level 0, there is no automation system, only depending on the driver to operate the vehicle. In Levels 1-2, the autonomous driving systems assist the human drivers, but they are still responsible for monitoring the environment. The system only carries out this activity in levels 3-5. In level 5, full automation, there is no necessary interaction between the driver and the system. Achieving this level is the aim of various companies, to make it part of a consumer product or for transportation purposes.

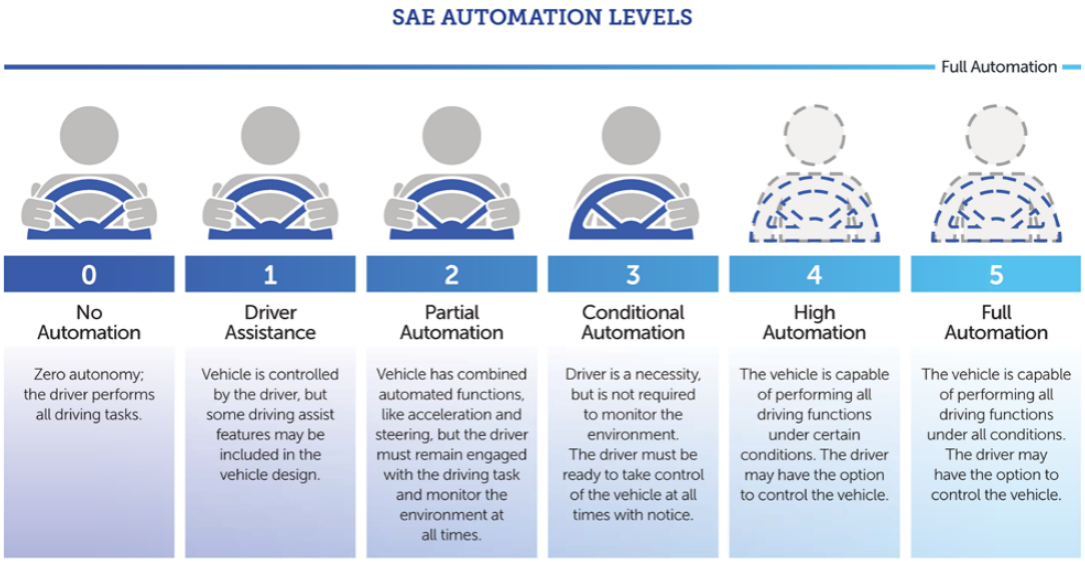


Figure 1.1: Levels of automation set forth in SAE J3016 standard, extracted from [13]

Typically, autonomous cars are equipped with one or multiple sensors to capture the data needed for perception, e.g., Cameras, Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RADAR), and ultra-sound sensors. These sensors make self-driving systems able to create a 3D visualization in real-time. While the choice of sensors used in autonomous driving vehicles is not the same amongst companies, recently, many of them have been equipped with LiDAR sensors. These sensors provide 3D information about the surrounding environment but are expensive and provide a sparse and often noisy signal. On the other hand, stereo cameras are cheaper, provide a richer signal, and have mature and reliable perception systems developed for their type of data [17]. Furthermore, every piece of information needed to drive autonomously (roads, signals, etc.) is built for the human eye. Nevertheless, Cameras have inferior performance under rough weather conditions and weak depth estimation.

In order to perform its task, a self-driving vehicle needs various modules. The vehicle needs to be aware of its surroundings, making perception systems necessary. Since every other module of an autonomous vehicle is dependent on perception, the perception system must be highly reliable and accurate. Autonomous driving systems interact with other traffic participants (e.g., cars, pedestrians) and for this reason, the autonomous driving system must have a system that detects and tracks objects in the surroundings. Tracking multiple objects allows the vehicle to be aware of other traffic participants, enabling navigation, planning, localization, and traffic behavior analysis. If the system is not tracking particular objects correctly, the decisions could result in destructive actions.

1.2 Motivation

As previously mentioned in Section 1.1, autonomous driving systems must know their surroundings in order to make decisions. Using the data from the vehicle sensors and various perception tasks, the system can understand the environment. A fundamental problem of environmental perception is Multi-Object Tracking (MOT) which is the process of identifying multiple objects in a scene and tracking them through time. The structured data extracted can then be used by other higher-level autonomous driving modules. While image-based 2D Object tracking is a well-developed area and 3D Object tracking using 3D point clouds research is gaining traction, the use of multi-modal data, more specifically 3D Point Clouds and RGB images, for MOT is still relatively new. Moreover, the fusion of the information from these modalities can help mitigate the performance drop caused by occlusions.

1.3 Objectives

The main goal of this dissertation is to study different 3D MOT approaches in the context of autonomous driving. Since almost all MOT approaches rely on object detection in the first step, different 3D and 2D object detectors are evaluated in the context of MOT. This evaluation is done using a State-of-the-Art MOT framework on a benchmark dataset for tracking methods. Furthermore, an analysis of how sensor fusion mitigates occlusion performance reduction in MOT is done.

1.4 Document Structure

This document is organized into six chapters. This first Chapter presents the context of this dissertation, motivation, and goals. Chapter 2 explains some theoretical background, used in the following chapters. Chapter 3 goes into detail about 3D Object Detection and Tracking and its respective State-of-the-Art. In Chapter 4 the methodology that is being used in this dissertation is presented, and Chapter 5 shows and discusses the results obtained. Finally, Chapter 6 presents the main conclusions and identifies future research paths.

Chapter 2

Background

This chapter covers the underlying concepts essential for the Multi-Object Tracking task. Most of the concepts explained work for both the context of 2D and 3D object tracking.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are a paradigm of Machine Learning (ML) based on biological models of the human brain. These networks provide learning capability to computers without being explicitly programmed.

Analogical to biological neural networks, ANN are composed of nodes connected by links called artificial neurons. The perceptron, introduced in 1957, is the base unit of a neural network, as seen in Figure 2.1. These are connected by weighted links that adjust with learning, increasing or decreasing the strength of the connection, using a learning algorithm, such as the Perceptron learning algorithm and the Back-propagation algorithm [32]. The perceptron computes the weighted sum of the inputs along with a bias value and then applies an activation function [1].

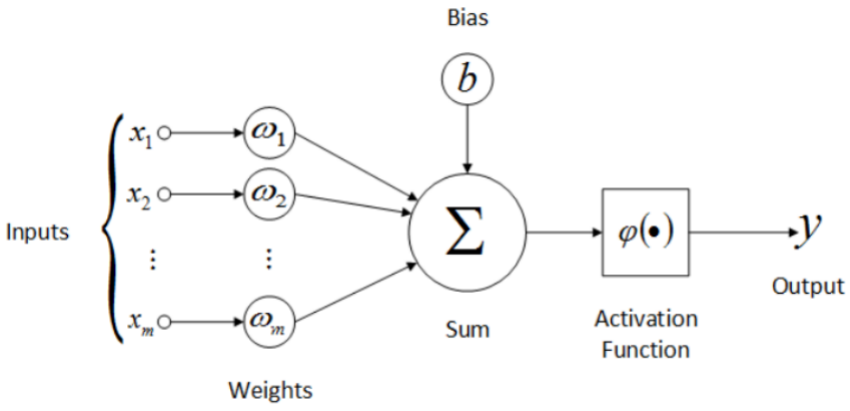


Figure 2.1: Perceptron overview, extracted from [26].

The obtain the output, y of a perceptron i, the following equation is used:

$$y = \varphi((\sum_{j=0}^{n} w_{ij}x_j) + b_i) \tag{2.1}$$

Where $\varphi()$, w_{ij} and b_i are the activation function, weight associated with the input x_j and bias of the neuron, respectively.

To simulate tasks done by the human brain, ANNs commonly have multiple layers of connected perceptrons: An input layer, one or multiple hidden layers, and an output layer. The number of perceptrons, as well as the number of hidden layers, is variable, depending on the task that is being solved.

Depending on how an ANN system is being trained, it can be categorized into different paradigms [1]:

- *Supervised Neural Network*: The network is trained using the inputs and outputs. When fully trained, it is supplied with testing data, previously unseen data, to anticipate the result.
- *Unsupervised Neural Network*: The network is trained without the output. Its goal is to establish a correlation between the incoming data and then group it appropriately. When new data is presented as input, it recognizes its feature and assigns it to one of the categories.
- *Reinforced Neural Network*: Learns with past decisions by getting penalized for bad decisions and rewarded for good choices. The weights producing correct outputs are increased and vice-versa.

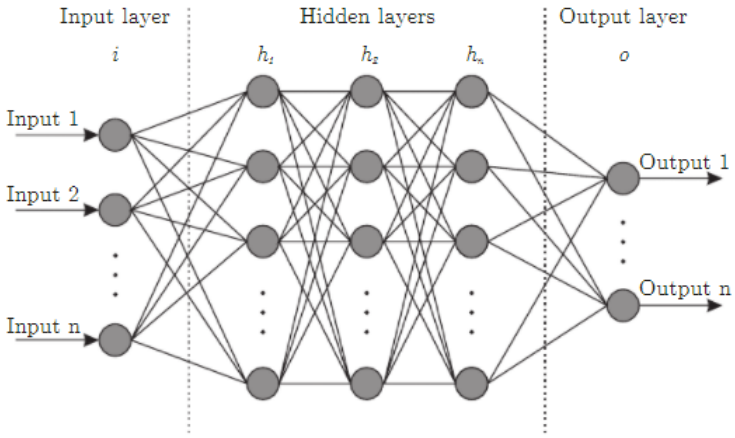


Figure 2.2: Example of a Deep Neural Network architecture, with three hidden layers, extracted from [4].

2.2 Deep Learning

Deep learning (DL) is a subset of ML, where the network has two or more hidden layers, as can be seen in Figure 2.2. Due to increased computational power and better algorithms to train these

networks, DL methods have reemerged in recent years. With multiple layers, Deep Neural Networks (DNN) can ingest and process unstructured data, eliminating data pre-processing typically involved with machine learning. These networks benefit from large amounts of data, learning more relevant characteristics, opening the doors to solving more complex problems and processing more difficult data. DL has various applications ranging from Natural Language Processing to Investment Modeling, with Computer Vision being one of the most prominent.

2.3 Convolutional Neural Networks

A Convolutional Neural Network, often known as CNN or ConvNet, is a type of deep learning network architecture that specializes in processing data with a grid-like layout, such as an image. CNNs reduce the requirement for manual feature extraction since the features are learned directly. Furthermore, CNNs generate extremely accurate recognition results. The architecture is designed so that simpler patterns (lines, curves, etc.) are detected first, followed by more complicated patterns (faces, objects, etc.). This is accomplished using convolutional layers and pooling layers. An example of a CNN is depicted in figure 2.3.

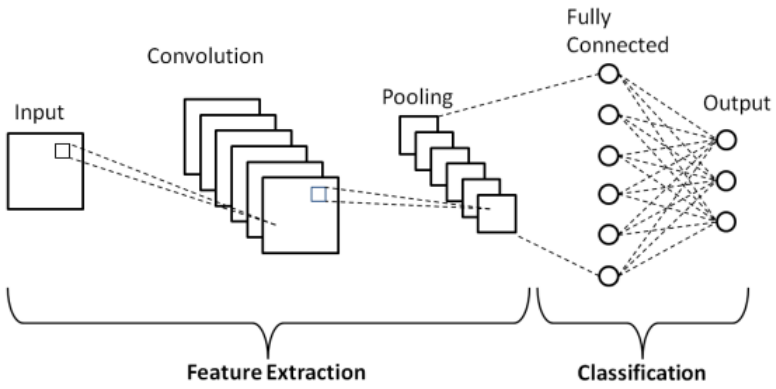


Figure 2.3: Example of a CNN, extracted from [38].

The CNN's main building block is the convolution layer, carrying the majority of the computational load on the network. This layer Converts the input tensor (e.g., an RGB picture) to a feature map via a convolution operation. Because linking every pixel of the input image to the neurons on the first convolutional layer would result in an exponential increase in the number of network parameters, convolutional layers employ weight sharing between neurons.

The pooling layer's major goal is to reduce the number of trainable parameters by shrinking the spatial size of the image, lowering the computational cost. It reduces the size of its input tensor by using local (or global) pooling, effectively compressing the information in a feature map. Maximum or average operations are typically used to accomplish this. These types of layers do not have associated weights and, consequently, do not learn any parameters.

2.4 Object detection

Object detection is a computer vision task that aims to detect an instance of visual objects in a scene. These objects are located and classified into classes (e.g., Humans, Cars, Animals), as shown in Figure 2.4. Thus, object detection can be classified as a regression problem to estimate the position and size of an object's bounding box and its classification. Object detection provides the basis of multiple computer vision tasks, including instance segmentation, event detection, and object tracking [57]. Consequently, object detection supports an extended range of applications such as robot vision, security, autonomous driving, and augmented reality [21].

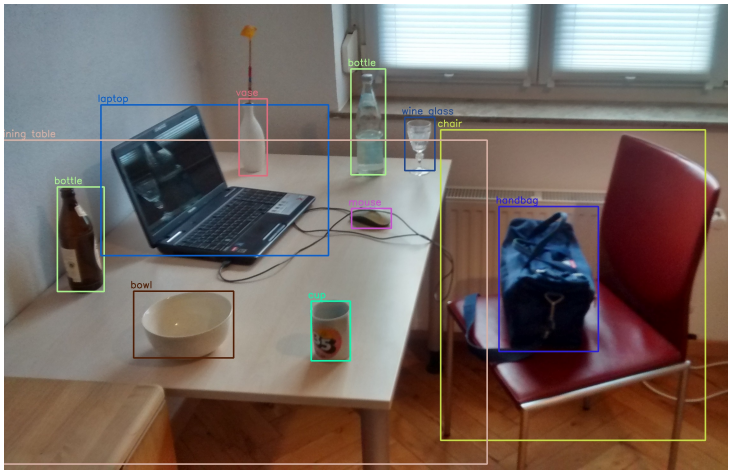


Figure 2.4: Example of a Object Detection task, extracted from [47].

Object detection techniques can be categorized into two approaches: Traditional based methods and Deep Learning Algorithms. The latter is now the most used approach, following the successful application of deep neural networks to the object detection task in 2014 [12].

Object detection methods are often divided into three stages: informative region selection, feature extraction, and classification. First, the image is scanned to generate region proposals, which are bounding boxes indicating likely item positions. The features from the region proposals are extracted using an algorithm in the second phase, Feature Extraction. Finally, a classifier is utilized to differentiate a target object from all other categories.

Most deep learning approaches use deep learning neural networks, especially Convolutional Neural Networks (CNN), for the feature extraction phase. This type of architecture is referred to as a two-stage object detector. Conversely, a One-Stage object detector proposes predicted boxes directly from the input image, skipping the region proposal step, such as YOLO [29]. Consequently, these types of methods are less computationally expensive, making them suitable for real-time applications, but achieve lower performance.

2.5 Multi-Object Tracking

The aim of Multi-Object Tracking (MOT) is to locate multiple objects in a scene, yielding their identities and maintaining their individual trajectories with high accuracy in a video. In other words, MOT automatically identifies objects and interprets them as a set of trajectories. It is a task that is necessary for high-level tasks such as action recognition, pose estimation, and behavior analysis [25]. Its practical applications include visual surveillance, autonomous driving systems, and virtual reality.

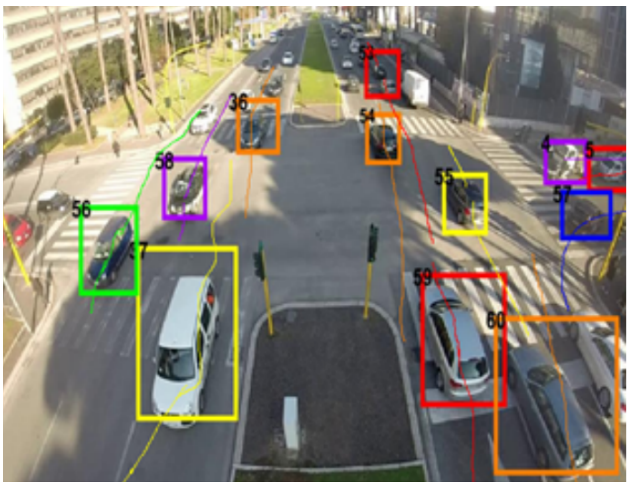


Figure 2.5: Example of a Object Tracking task, extracted from [9].

MOT frameworks follow three main paradigms [25]:

- *Detection-free tracking*: A fixed number of object detections are inserted manually in a frame, then these are tracked in the following frames.
- *Tracking-By-Detection*: In the first phase, objects are detected in a single frame, using 2D or 3D Object detectors. The second phase associates objects across frames by applying a filtering algorithm, which is often the hardest step in the tracking-by-detection paradigm. This is the most popular approach for MOT frameworks since new objects are automatically discovered, and, likewise, disappearing objects are terminated automatically. Moreover, pre-trained Object detectors can be used.
- *End-to-End*: In this approach, one model is responsible for both tasks. While harder to implement, it can have better tracking accuracy since the whole network can be trained using tracking metrics, and it is not reliant on the quality of the object trackers.

These frameworks can also be classified on their processing mode, with the difference being whether future frames can be analyzed when handling the current frame [25].

- *Online tracking*: In online tracking, frames are handled in a sequential manner. Only information from the current and previous frames is used. These methods are less accurate than offline tracking methods but are needed for online tasks (e.g., autonomous driving)

- *Offline tracking*: A batch of frames, including future frames, is used for tracking in the current frame. This approach requires all the frames being processed to be obtained beforehand.

Chapter 3

Object Detection and Multi-Object Tracking with 3D Point Clouds

This chapter discusses 3D Object Detection and 3D Multi-Object Tracking, as well as 3D Point clouds and how they are used in those subjects.

3.1 3D Point Clouds

3D Point Clouds are a set of data points in 3D space. The point cloud of a scene is the collection of 3D points around objects in that scene, shown in Figure 3.1. Each point is represented with XYZ coordinates but can have additional information such as the light pulse intensity and RGB values. Light Detection and Ranging (LiDAR) sensors are the most common device to capture 3D Point Cloud data. These sensors work by emitting pulsed light waves from a laser and measuring the time it takes to travel back. The light rays are reflected on the scene's object, indicating their distance from the sensor. LiDAR is commonly used for mapping areas in multiple disciplines, such as geography, engineering, control, and navigation.

Despite providing depth to the data, 3D Point clouds have considerable shortcomings. Point clouds are often sparse since sample points are not uniformly distributed in the 3D space, have high dimensionality, low resolution, and are unstructured, which makes creating perception models for this type of data challenging. When Point cloud data is not sparse, it can easily reach hundreds of millions of points. Moreover, if the data is acquired from a single viewpoint, objects located behind the surrounding environment will not be captured by the LiDAR since the first object hit by the laser will block them. This phenomenon is named occlusion, and since Autonomous driving vehicles only capture data from a single viewpoint, it affects perception methods in the context of autonomous driving. The low resolution of LiDAR sensors also makes detecting faraway objects harder, especially compared to RGB Images provided by a camera sensor [44].

As previously mentioned, 3D Point clouds are a set of data points, unordered and without any structure. A point in a 3D Point Cloud is often represented as $p_i = (x, y, z, p)$, where p is the light

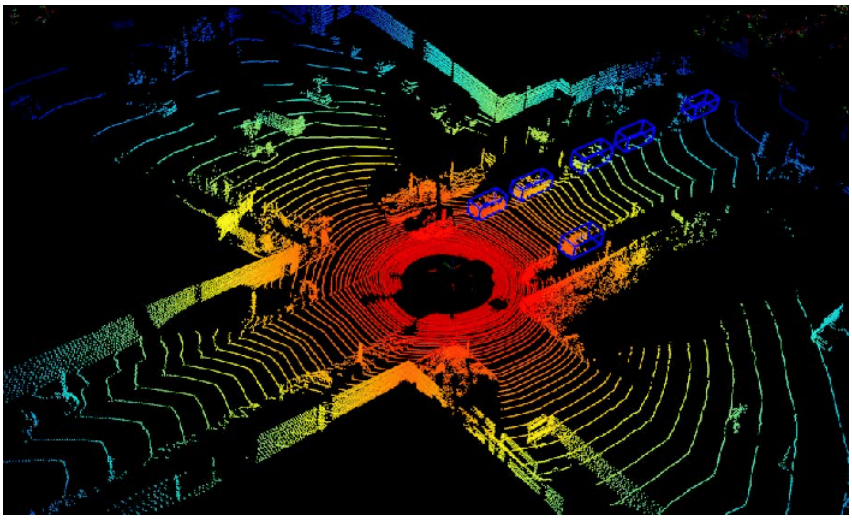


Figure 3.1: Example of a 3D Point Cloud scene, extracted from [45].

pulse intensity. To use this data in perception methods, 3D Point clouds are often represented in four different matrix representations [6]:

- *Raw Points*: Every single 3D point in the set is listed as one row in a matrix. This is the simplest and less processed way to represent 3D Point clouds in a matrix. Furthermore, it preserves all the original information. However, this method does not take advantage of the geometric properties of 3D Point Clouds.
- *3D Voxelization*: The 3D space is discretized into voxels that represent a 3D point cloud. Usually, the 3D space is partitioned into equally sized non-overlapping voxels, as presented in Figure 3.2 (a). Voxels are coupled with a natural hierarchical structure in this method, which reduces storage space. The loss of resolution is the most significant disadvantage.
- *Range view*: This approach models how a LiDAR captures 3D points. The 3D points are rearranged into a 2D range-view image. Each pixel in the range-view image corresponds to a frustum in the 3D space, as seen in Figure 3.2 (b). This approach makes for a compact range-view image; however, modeling an unorganized point cloud is hard.
- *Bird's eye view*: This approach is a subset of voxelization that disregards height. It projects 3D voxels to a Bird's eye view (BEV) image, presented in Figure 3.2 (c). The main advantages of this approach is that it is possible to apply 2D perception methods, easy to merge information, and the object's size is independent of range, unlike the range view approach.

3.1.1 Sensor Fusion

While LiDAR's 3D Point Clouds provide 3D information of the surrounding environment. These sensors are less affected by weather variation and illumination changes than cameras. However, they also present a lot of downsides, mentioned in 3.1, such as sparse data and low resolution.

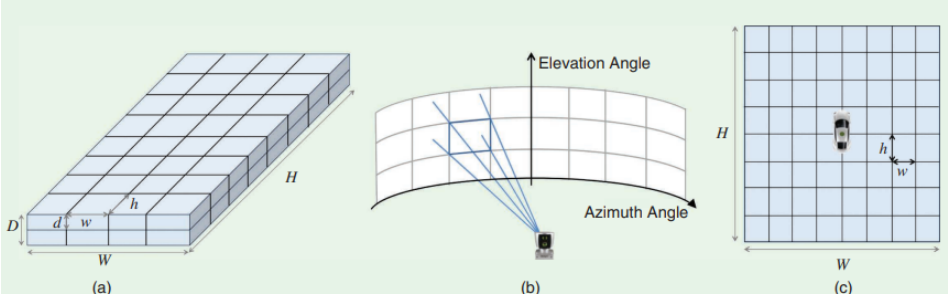


Figure 3.2: Overview of the most used ways to represent point clouds in matrix representations, extracted from [6] (a) 3D Voxelization (b) Range view (c) Bird's eye view.

Conversely, RGB images have a rich signal, provide texture information, and are long-range. Fusing the data between these two sensor modalities benefits perceptions methods, namely MOT methods, by complementing the shortcomings of each sensor. There are three main approaches to fusing LiDAR and RGB data:

- *Early Fusion*: Early Fusion approaches fuse raw or pre-processed data from both sensors before feeding it into the framework. Since only the network to process the fused data is needed, this approach has a low computational cost compared to other methods. Since sensors can have different sampling rates and defects, extra pre-processing is needed.
  - *Sequential Fusion*: Sequential fusion is a type of early fusion that merges the extracted information from one module with the data from the other sensor.
- *Late Fusion*: Conversely to early fusion, late fusion approaches consist of having different networks for each sensor and then combining the output into a single parameter. This approach has high flexibility since it allows for choice in architectures that process each type of sensor data that can also be trained separately, without interfering with each other. However, since it requires multiple networks, these models are less computationally efficient than early fusion methods.
- *Deep Fusion*: A deep fusion architecture combines early and late fusion methods, fusing data or feature representations from different modalities multiple times across the network. Consequently, the network can learn cross modalities with varying features along with the network.

3.2 3D Object Detection

Self-driving cars need to be completely aware of their surroundings. For these vehicles to identify which road elements, such as cars, cyclists, and pedestrians, are present in the scene, 3D object detection is essential. The process of 3D object detection entails identifying oriented 3D bounding boxes corresponding to different objects in each scene formed by LiDAR data. Contrary to 2D

object detection, a more developed area, 3D object detection remains an open subject with plenty of room for improvement and study.

3D object detectors are commonly classified into three types based on the way they handle point cloud representation: grid-based, set-based (or point-based), and graph-based networks

3.2.1 Set-Based Methods

Set-based methods are sometimes known as point-based methods since they use the original point cloud representation. These approaches are often intended to receive unaltered point clouds with the goal of extracting features, classifying 3D shapes, and segmenting them. The LiDAR point cloud retains its unstructured form; however, when translated into a defined size, its representation becomes more compact. This is performed by subsampling the point cloud from its original size to a smaller fixed size of N points using random sampling and Furthest Point Sampling (FPS) [54].

PointNet [7], proposed in 2016, is a revolutionary effort in Deep Learning networks for point clouds that has influenced a wide spectrum of point cloud perception methods. Despite the fact that it was created for segmentation and classification, most 3D object detectors currently employ PointNet or variations of it to extract features from unaltered point cloud data. Point clouds are a naturally unordered set of vectors that are sparse and locality sensitive. As a result, PointNet was created to interpret 3D points regardless of their order. The core concept is to use a group of MLPs, global max-pooling, and spatial transformer networks to ensure input-wise permutation invariance and independent extraction of pointwise features from a set of 3D points. Nevertheless, PointNet does not capture spatial and geometric knowledge between local points. PointNet++ [28], introduced in 2017, proposed an abstraction level, shown in Figure 3.3, consisting of a sampling layer, a grouping layer, and the PointNet-based learning layer, aimed to counteract this issue.

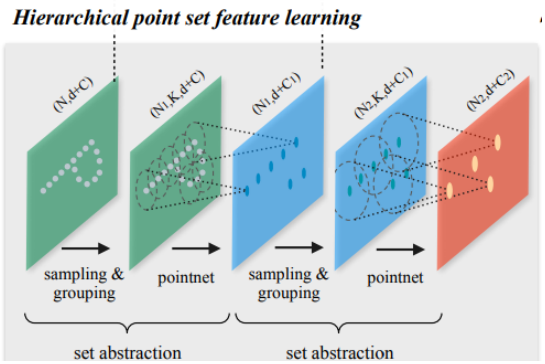


Figure 3.3: PointNet++ set feature learning module, extracted from [28].

Since both PointNet and PointNet++ were not developed for 3D object identification, they are not 3D object detectors. However, they did introduce important feature extraction processes, such as the PointNet architecture and set abstraction layers, which are important components used by a large number of 3D object detection methods.

3.2.2 Grid-Based Methods

The process of allocating points to voxels is known as voxelization. Grid-based or 3D voxel-based approaches transform point clouds into regular grids to be processed by 2D or 3D convolutional neural networks. Partitioning 3D space using a Cartesian or cylindrical coordinate frame yields voxels with cuboid or cylindrical slice shapes, respectively.

VoxelNet [56] is an important pillar for grid-based methods. The voxel feature learning network, depicted in figure 3.5, is the centerpiece of VoxelNet. It employs the PointNet [7] approach to integrating the points' features in the voxel via learnable Voxel Feature Encoding (VFE) layers. The VFE is fed a set number of points sampled from each voxel as input. Each point has three dimensions, reflectivity and a relative offset from the physical center point. The VFE translates raw LiDAR data to a 3D voxelization-based representation while simultaneously learning point-wise features in each voxel using a PointNet. Finally, an area proposal network is linked to predict 3D bounding boxes. Despite their efficiency and utility, 3D convolutions are computationally expensive, impacting the network's inference time significantly. Furthermore, due to the sparsity of point clouds, voxel representation generates a considerable number of blank voxels, resulting in needless calculations.

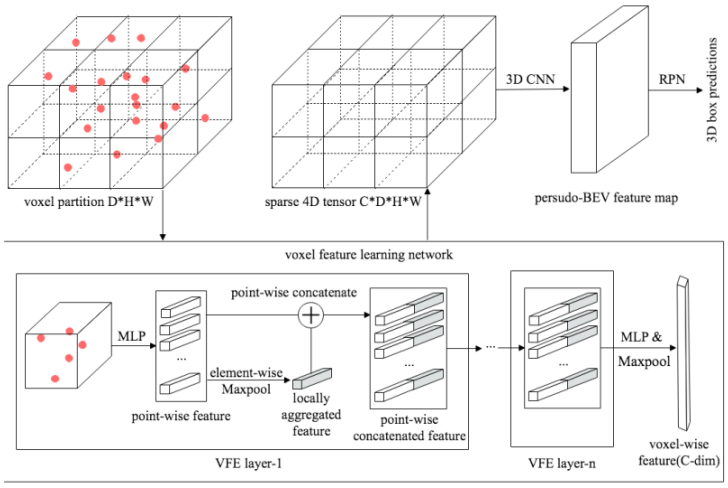


Figure 3.4: VoxelNet arquitecture overview, extracted from [56].

To mitigate this issue, Yan et al. proposed SECOND [51], in which 3D convolutions in VoxelNet [56] are substituted by submanifold convolutional layers and sparse convolution layers. SECOND enhances VoxelNet's performance and achieves a significantly faster speed, but its 3D convolutions remain an efficiency bottleneck.

PointPillars [20] eliminates 3D convolutions by considering the pseudo-BEV map as a voxelized representation, allowing for end-to-end learning with only 2D convolutions, resulting in higher speed, being 3 times as fast as SECOND [51].

PV-RCNN [35] captures point-wise characteristics from PointNet-based networks to enrich 3D voxel CNN proposals. The point-wise feature is encoded from a limited collection of key

points sampled from the entire point cloud by the FPS method. The keypoint feature is a concatenation of three features: a raw point cloud recovered by PointNet [7], a bird-view feature retrieved by voxelization, and z-axis feature aggregation, and aggregation of neighbor voxel features with differing receptive fields extracted by PointNet.

To save follow-up computations and to encode representative scene properties, Voxel R-CNN [8] summarizes the 3D scene using a 3D voxel CNN into a compact set of keypoints using a novel voxel set abstraction module. Given the voxel CNN's high-quality 3D proposals, RoI-grid pooling is proposed to abstract proposal-specific properties from keypoints to RoI-grid points via keypoint set abstraction with multiple receptive fields. The RoI-grid feature points encode far richer context information than standard pooling processes, allowing for more accurate estimation of object confidences and locations.

Part-A^2 Net [36], which stands for Part Aware and Part Aggregation Network, is a two-stage detector that comes in two varieties: Anchor-based and Anchor-free, all of which follow the same architectural principles. Part-A^2 Net [36] uses free intra-object annotations and point-wise semantic annotations to monitor voxel feature learning. The voxelized point cloud is fed into the first stage of Part-A^2 Net [36]. A backbone network with sparse convolution and deconvolution learns to segment foreground points and estimate the intra-object part placement of all foreground points. In the first stage, 3D object proposals are generated concurrently. In the second stage Part-A^2 Net [36] partitions each candidate 3D box into multiple voxels to pool all features from both non-empty and empty voxels for improved geometric information capture.

3.2.3 Graph-Based Methods

The point cloud is transformed into a graph in this representation. Points are regarded nodes, while connections between points that are located within a given radius are considered edges. Because building a graph from raw point clouds is computationally inefficient, a down-sampled point cloud is utilized instead, usually after voxelization.

Point-GNN [37], a one-stage detector, employs a graph-based technique from start to finish. After building a graph after voxelization, the initial feature of each vertex is calculated using MLPs, similarly to PointNet [7]. For high-dimension feature extraction, a 3DBN based on a graph neural network(GNN) architecture composed of MLPs is used. Point-GNN, unlike traditional GNNs, is engineered to encode spatial information with learned high-dimensional characteristics. The GNN is run for a set number of iterations. Because the MLPs differ for each iteration, weights are not shared between iterations. In an Anchor-free approach, two separate MLPs are utilized at the conclusion of GNN, one for classification and one for per class 3D Bounding box regression.

3.3 3D Multi Object Tracking

The purpose of 3D MOT is to associate 3D detections in a sequence, as seen in figure 3.6. 3D multi-object tracking is an important component in an autonomous driving system since it gives critical information to various onboard modules such as perception, prediction, and planning. The

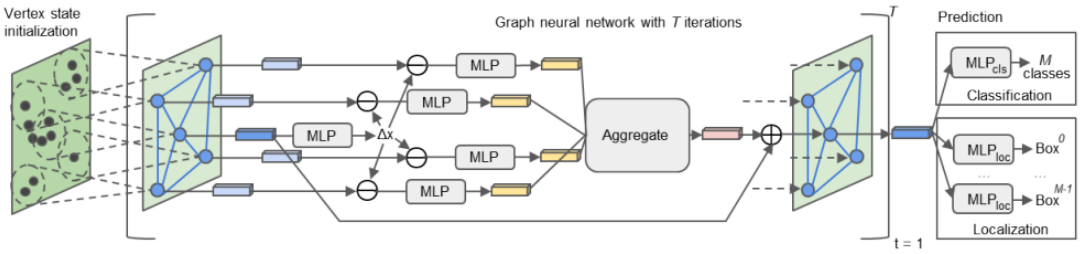


Figure 3.5: Point-GNN arquitecture overview, extracted from [37].

difference is that the input detections are in 3D space rather than the image plane. As a result, 3D MOT systems may acquire motion and appearance information in 3D space while avoiding perspective distortion. LiDAR is the most popular sensor used by self-driving vehicles to perceive their environment.

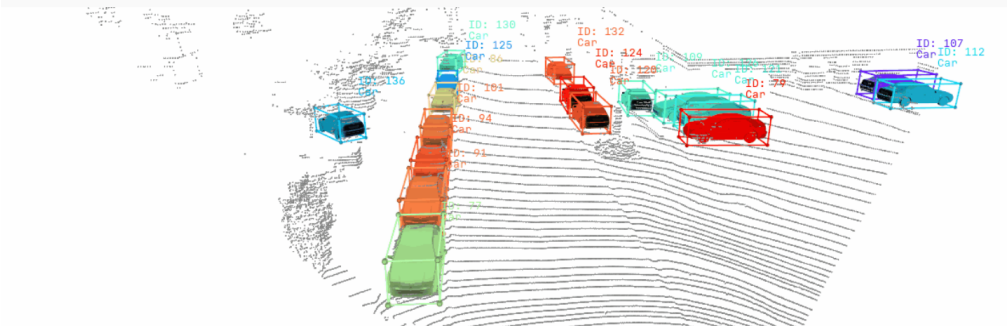


Figure 3.6: Example of 3D MOT task, extracted from [15].

The tracking-by-detection architecture, explained in section 2.5, is used by the majority of 3D multi-object tracking systems. They provide 3D object detection results as input to tracking techniques. Various distance metrics are employed in the data association step to discover the matching track-detection pairs.

3.3.1 Evaluation Metrics

In order to evaluate and compare the performance of MOT frameworks, there is a necessity to use various tracking metrics. Moreover, while developing a system, it is important to observe the influence of different parameters and modules in said system. The HOTA (Higher Order Tracking Accuracy) [24] metrics were introduced in 2021, designed to address many of the shortcomings of previous measures, such as overemphasizing the importance of either detection or association. The previously tracking metrics included the CLEARMOT metrics [3], the Identity metrics [31] and the Track mAP metrics [52].

HOTA may be considered as a mixture of three IoU scores. It splits the process of evaluating tracking into three subtasks: localization, detection, and association, and assigns a score to each using an IoU (intersection over union) formulation. It then aggregates the three IoU scores for each subtask to provide the final HOTA score.

The localization subtask measures the spatial alignment between one predicted detection and one ground-truth detection. Localization IoU (Loc-IoU) is utilized to measure localization accuracy and can be computed as the ratio of the intersection (overlap) of the two detections to the entire area covered by each of them (union). A visualization is depicted in figure 3.7.

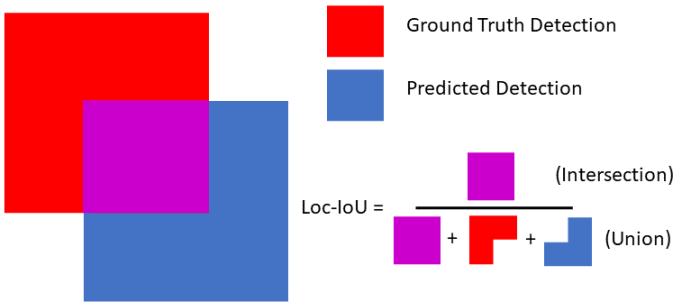


Figure 3.7: Loc-IoU calculation visualisation, extracted from [22].

This notion can be extended from bounding boxes to segmentation masks; increasing the Loc-IoU score improves the spatial alignment of anticipated and ground-truth detections. The Localization Accuracy (LocA) can be calculated by averaging the Loc-IoU across all pairs of matching predicted and ground-truth detections in the whole dataset, as seen in the following equation:

$$\text{LocA} = \frac{1}{|\text{TP}|} \sum_{c \in \text{TP}} Loc{-}IoU(c) \tag{3.1}$$

Where TP and c are True positives and a dataset instance, respectively.

Detection measures the alignment between the set of all predicted detections and the set of all ground-truth detections. For assessing detection accuracy, detection IoU (Det-IoU) is often utilized. A location threshold (e.g., Loc-IoU > 0.5) is used to define the set of predicted detections intersecting with ground-truth detections. One predicted detection, however, may overlap with more than one ground-truth detection (and vice-versa). To address this, the Hungarian method [19] is used to find a one-to-one match between predicted and true detections. These matching detection pairs are known as True Positives (TP), and they represent the intersection of the two sets of detections. False Positives (FP) are predicted detections that do not match, and False Negatives (FN) are ground-truth detections that do not match (FN). Det-Iou can be defined as:

$$Det{-}IoU = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{3.2}$$

While Loc-IoU assesses the alignment of a single expected and ground-truth detection, Det-IoU measures the alignment of all predicted and ground-truth detections. Detection Accuracy (DetA) can be obtained by calculating Det-IoU using the whole dataset, defined as:

$$DetA = Det{-}IoU = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{3.3}$$

The ability of a tracker to link detections over time into the same identities (IDs) is measured by association. This can be measured by taking a predicted detection and a ground-truth detection that have been matched together (using the Hungarian matching) and evaluating the alignment between the anticipated detection's whole track and the ground-truth detection's entire track, which can be represented in another IoU formulation.

The intersection of two tracks may be quantified as the number of True Positive matches between the two tracks, named True Positive Associations (TPA). Detections that are either mismatched to other ground-truth tracks or to none are False Positive Associations (FPA), and the remaining detections in the ground-truth track are False Negative Associations (FNA). The Association IoU (Ass-IoU) may be computed in the same manner as shown:

$$Ass{-}IoU = \frac{|TPA|}{|TPA| + |FNA| + |FPA|} \tag{3.4}$$

The overall Association Accuracy (AssA) may be calculated by averaging the Ass-IoU over all pairs of matching predicted and ground-truth detections in the whole dataset:

$$\text{AssA} = \frac{1}{|\text{TP}|} \sum_{c \in \text{TP}} Ass{-}IoU(c) \tag{3.5}$$

Where TP and c are True positives and a dataset instance, respectively.

HOTA is a combination of all three IoU scores previously defined. Every subcomponent mentioned is extremely important for measuring tracking performance, hence the importance of having a single metric that combines all of them. HOTA can be calculated as follows:

$$HOTA_\alpha = \sqrt{DetA_\alpha \cdot AssA_\alpha} = \sqrt{\frac{\sum_{c \in \text{TP}_\alpha} Ass{-}IoU_\alpha(c)}{|\text{TP}_\alpha| + |\text{FN}_\alpha| + \text{FP}_\alpha|}} \tag{3.6}$$

$$\text{HOTA} = \int_{0<\alpha\leq 1} \text{HOTA}_\alpha \approx \frac{1}{19} \sum_{\alpha=0.05}^{0.95} \text{HOTA}_\alpha \tag{3.7}$$

Where α is the location threshold.

There are also metrics with concepts commonly used in object detection. Detection recall (DetRe) is the proportion of ground-truth detections that were accurately predicted, whereas detection accuracy (DetPr) is the percentage of right detection predictions made, defined as follows:

$$DetRe = \frac{|TP|}{|TP| + |FN|} \tag{3.8}$$

$$DetPr = \frac{|TP|}{|TP| + |FP|} \tag{3.9}$$

As for the association counterpart, association recall (AssRe) measures how well predicted trajectories cover ground-truth trajectories, and association precision (AssPr) measures how well predicted trajectories keep tracking the same ground-truth trajectories. These metrics can be defined as:

$$\text{AssRe} = \frac{1}{|\text{TP}|} \sum_{c \in \text{TP}} \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)|} \tag{3.10}$$

$$\text{AssPr} = \frac{1}{|\text{TP}|} \sum_{c \in \text{TP}} \frac{|TPA(c)|}{|TPA(c)| + |FPA(c)|} \tag{3.11}$$

3.3.2 Benchmark Datasets

Deep learning algorithms, such as 3D object detection networks, require a vast amount of data for training. Because it is done manually, collecting and annotating data can be an expensive and laborious procedure. Fortunately, there are a number of datasets available for research.

3.3.2.1 KITTI

Despite its 2012 release, KITTI [10] has become the standard dataset for perception tasks like 2D or 3D object detection, scene flow, depth evaluation, tracking, and so on. Cutting-edge models are still tested and assessed using this dataset. It was created by carefully sampling a fraction of the raw sensor input and calibration data, labeling it, and structuring it according to the needs of each activity.

The KITTI dataset is especially relevant to autonomous driving tasks, as it is recorded in the streets of Karlsruhe, Germany, using a LiDAR sensor, two forward-facing stereo cameras, and a GPS, as shown in figure 3.8.

Figure 3.8: Sensor setup used to record the KITTI dataset, extracted from [11].

The tracking benchmark was released in 2013 and consists of 21 training sequences and 29 test sequences, with a total of 8026 frames and 11125 frames, respectively. Every frame contains a 3D Point Cloud, a 2D image and calibration information. Despite the fact that eight different classes are labeled, only the classes 'Car' and 'Pedestrian' are examined in the benchmark since only those classes have enough instances tagged for a full evaluation.

3.3.2.2 nuScenes

NuScenes [5] is a more recent public multi-model dataset for detection and tracking in the context of autonomous driving. It is larger and more robust than KITTI [10], providing illumination and weather changes. Its scenes are recorded with six cameras, five radars, 1 LiDAR, and a GPS, all with a full 360-degree field of view. NuScenes comprises 1000 scenes, each 20 seconds long, for 23 classes and eight attributes.

3.3.3 Multi Object Tracking for Autonomous Driving

This subsection goes into detail about recent developments in MOT methods used for Autonomous Driving. An overview of these methods' performance and their properties can be seen in tables 3.1 and 3.2, respectively.

3.3.3.1 Beyond Pixels

In recent years, significant improvements have been made in 3D MOT. Beyond Pixels, [33] proposed a fast and straightforward yet accurate and robust MOT applied to urban road scenarios by taking advantage of objects' shapes and poses temporal consistency. This algorithm uses the tracking-by-detection approach, explained in section 2.5, where the objects are detected in the first phase and then associated with objects from previous frames. This technique uses LiDAR and RGB data to take advantage of some characteristics inherent to the object and its movements, such as object pose, shape, and motion.

First, the paper uses two different object detectors to compare results - RCC [30] and SubCNN [50]. A threshold for confidence scores filters the detections obtained, and a NMS is used to reduce multiple detections around the same object. To obtain the associations in the second phase, the method computes the associated cost for every object as follows:

- *3D-2D cost*: measures the overlap of the target's expected 2D region with a given 3D detection;
- *3D-3D cost*: measures the overlap of the target's expected 3D bounding box with a given 3D detection;
- *Appearance cost*: uses a DNN to measure appearance between 2D regions;
- *Shape and Pose cost*: explores additional information provided by an external algorithm that computes a vector of deformation coefficients. The similarity between a target object and a given detection is calculated with that information.

After computing the costs, the detections and tracked objects are associated using a Hungarian algorithm [19].

In terms of results, this paper [33] achieved a HOTA of 63.74% on the KITTI dataset [10], as can be seen in table 3.1. Despite its good results, this method has a high number of identity

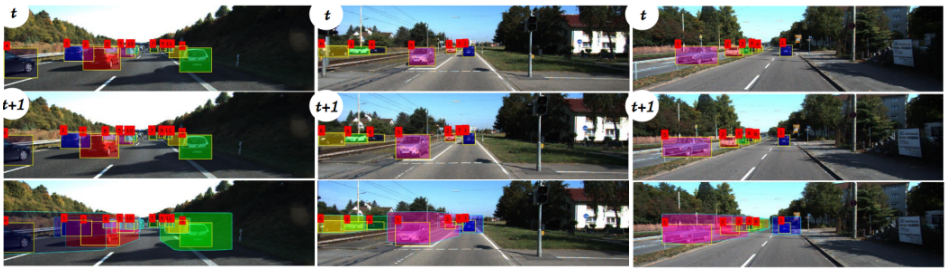


Figure 3.9: Overview of Beyond Pixels Method, extracted from [33].

(ID) switches and can only run at 3 FPS, which is lower than the 10 FPS threshold of real-time tracking. Beyond suggests that the ID switches problem is the result of the proposed approach of implementing an online tracker, but more developed online trackers mitigated this issue by using a tracker with a more sophisticated cost association.

3.3.3.2 FANTrack

In 2019, FANTrack [2] used siamese networks to model the similarities between tracked objects and detections, as well as DLNN to mitigate the association problem in MOT. This method also takes as an input LiDAR and RGB data. Here, multiple branches inside the similarity network for the association cost problem are applied:

- *Bounding box branch*: outputs a vector of dimension features for detections and targets, later used to calculate bounding box similarities;
- *Appearance branch*: outputs a vector of 2D visual cues for both detections and targets;
- *Importance branch*: receives as input the first two branches, and its goal is to determine the relative relevance of each branch.

A similarity map of each target is generated, where the outputs of the previous branches are considered, i.e., the weight of the first two branches (bounding box and appearance) is set by the latter branch (importance). Then, the targets and detections are associated to create pairs, using another siamese network that outputs the probability for each target-detection pair. Finally, it updates the tracking information, a future state prediction is made for each target, and tracks are initiated and pruned if a new object is detected or stops being detected, respectively. The 3D Object detector used in the paper was Aggregate View Object Detection (AVOD) [18].

In terms of results, this paper achieved a worse performance (HOTA of 60.85% - table 3.1) than Beyond's approach; however, it had considerably fewer ID switches and ran at higher a refresh rate (25 FPS) suitable for autonomous driving. It is also penalized by the KITTI evaluations because they are done in 2D, while this approach provides inferences in 3D.

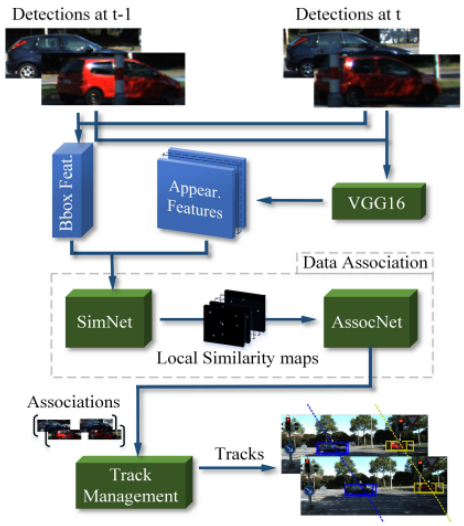


Figure 3.10: Overview of FANTrack Method, extracted from [2].

3.3.3.3 AB3DMOT

AB3DMOT [46], a fast and effective 3D MOT, was proposed in 2020. This architecture, unlike the previous, only uses 3D data and presents great results compared to them. The method has several modules, composing a simple pipeline, as can be seen in Figure 3.11:

- *Object detector*: Outputs detected objects - their center, orientation, size and detection confidence;
- *State prediction*: Module that predicts object's next frame state - its position, direction, and velocity - using a Kalman filter. Only predicts the velocity after the second frame, with a constant velocity model;
- *State update*: This module updated object data with a weighted average of both detection and tracking in order to account for detection and tracking uncertainty;
- *Data association*: Associates detections with tracked objects using a Hungarian algorithm. Rejects pairs that have an intersection over union (IoU) lower than a threshold. Outputs three types of results: detections and tracks matched, tracks unmatched, and detections unmatched (new objects).
- *Birth and death memory*: Starts and deletes tracks, with the aim of avoiding creating false positives and false negatives, since unmatched defections could be a detector error and detectors could miss detection an object, respectively. To prevent this, new tacks are only created after detecting the same object a certain number of times. Likewise, tracks are only deleted if it is not matched with detections for a certain number of times in a row.

Except for the pre-trained 3D detection module, this system requires no training and may be utilized for inference right away.

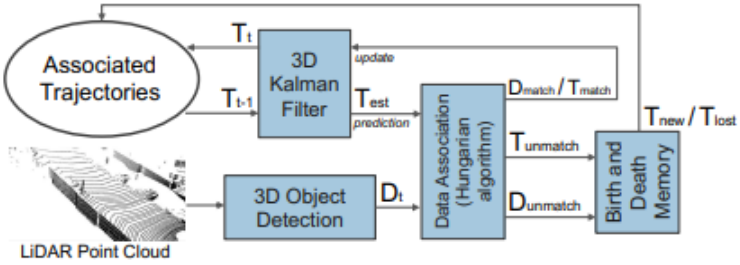


Figure 3.11: Overview of AB3DMOT Method, extracted from [46].

In terms of results, this paper achieved 69.99% HOTA score 3D MOT performance on the KITTI dataset, as we can see in table 3.1. It also achieved an impressive 313 ID switches and had a low computation cost, running at 207 FPS.

3.3.3.4 mmMOT

Multi-modality MOT (mmMOT) [55] is a 3D MOT framework that works with various optional sensor modalities (Camera, LiDAR, radar). The main idea of this approach is to preserve reliability by independent multisensor feature extraction and improve accuracy via modality fusion. Consequently, it also follows the tracking-by-detection paradigm. This framework contains four modules, presented in Figure 3.12:

- *Object detector*: Detections for each modality are obtained by different 3D object detectors;
- *Feature extractor*: First features from each modality are extracted using independent feature extractors. The features are then combined with three different fusion modules in the robust fusion module. The first concatenates the features, the second fuses them with addition, and the third uses an attention mechanism in order to guide the information fusion from different sensor modalities since, depending on the situation, the sensor's information might have different significance.
- *Adjacency Estimation*: The adjacency estimator calculates the confidence, affinity, start, and finish scores in the min-cost flow graph depending on each modality using the multi-modality features as input.
- *Optimization*: The framework uses linear programming to determine the best solution from the min-cost flow graph using the prediction score from the neural network as an input.

This method achieves an 62.05% HOTA value on the KITTI tracking dataset, as presented in table 3.1.

3.3.3.5 EagerMOT

EagerMOT [17] is another tracking-by-detection MOT framework that fuses LiDAR and RGB data with state-of-the-art performance. This approach uses pre-trained object detectors to merge

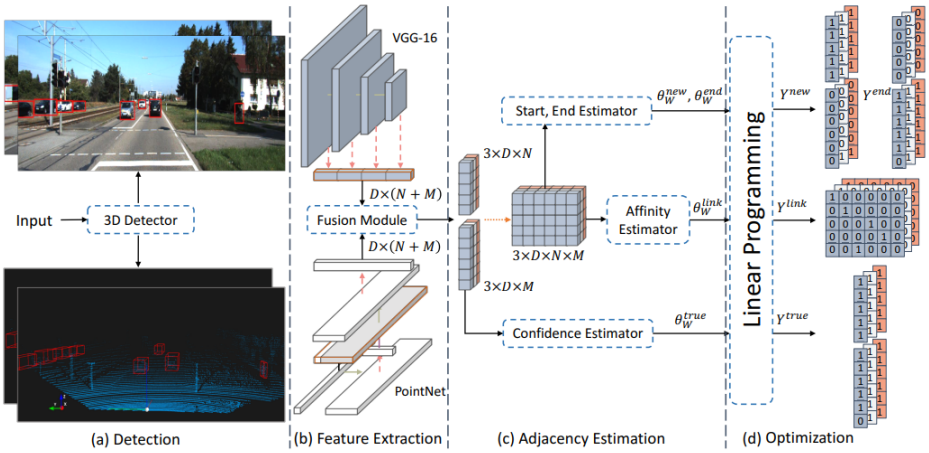


Figure 3.12: Overview of mmMOT Method, extracted from [55].

complimentary 2D and 3D object evidence. The data passes through the following modules after receiving object detections from various sensor modalities as an input(Figure 3.11):

- *Fusion module*: This module fuses the 2D and 3D detections into fused object instances. To associate the detections in 3D and 2D, they are greedily associated using their overlap in the image domain. The greedy association sorts all possible detection pairings by their overlap in descending order. Pairs are considered one by one and only joined if their overlap is above a certain threshold and neither detections as been matched yet.

- *Two-stage data association module*: This module allows the method to update object tracks using two stages. First, instances with 3D information are matched to existing tracks. The module greedily pairs detected instances with tracks based on the scaled distance between instances' oriented bounding boxes and tracks' predicted oriented boxes. In the second phase, tracks that were unmatched in the previous step are matched with instances localized only in 2D. This association stage greedily associate instances to remaining tracks based on the 2D IoU criterion. With the two-stage data association module, the method is able to recover from temporary occlusions and maintain an approximate 3D location when one of the detectors fails.

- *Track management module*: This module manages Object trajectories and life cycles. The rules used are as follows: If a track has not been updated with any instance in a certain amount of frames, it is discarded; If a track is associated with an instance in the current frame and has been updated with 2D information in the last few frames it is considered confirmed; Detected instances start new tracks if they were never matched before.

The experiments were done with Point-GNN [37] and Point R-CNN [34] 3D object detectors and RRC [30] 2D detector. This setup obtained a 74.39% HOTA score on the KITTI dataset as we can see in table 3.1.

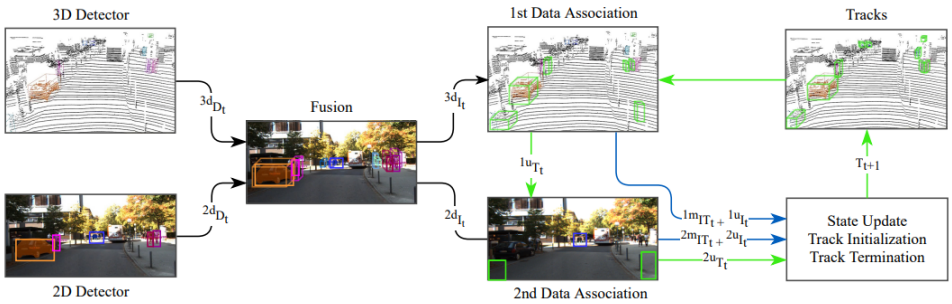


Figure 3.13: Overview of EagerMOT Method, extracted from [46].

3.3.3.6 PC-TCNN

The accuracy of a MOT framework relies on the quality of detections inside it. To circumvent this issue, PC-TCNN [49] was proposed, being an end-to-end tracklet-based (short trajectory) network instead of the tracking-by-detection approach that the previous methods followed. This method is a tracklet proposal Convolutional Neural Network, exploiting the temporal-consistent features to improve tracklet tracking accuracy. Since the detection and tracking are joint, they are trained together, improving thus the overall tracking performance. This method requires LiDAR and GPS data.

This framework consists of three modules:

- *Tracklet proposal generation*: Contrastly to most online approaches, instead of only using the previous and current frame, the input is a sequence of point cloud frames. These are encoded into birds' eye view feature maps, and a 3D object proposal generation and motion regression are computed to generate a set of tracklet proposals.

- *Tracklet proposal refinement*: The goal of this module is to increase the accuracy of tracklet detections. A tracklet feature aggregation technique addresses this by capturing features from the spatial-temporal region of interest on the point cloud series, resulting in more accurate object detection and localization in tracklets.

- *Tracklet association*: To create the final tracking results, this module links the refined tracklets with earlier trajectories (initialized with an empty set at the first timestamp). A greedy matching technique connects tracklets and trajectories based on their 3D IoU. If the tracklets are not matched, the following frame will begin a new trajectory.

PC-TCNN provides state-of-the-art performance with a HOTA of 80.90 %in the 'Car' class of KITTI tracking dataset [10].

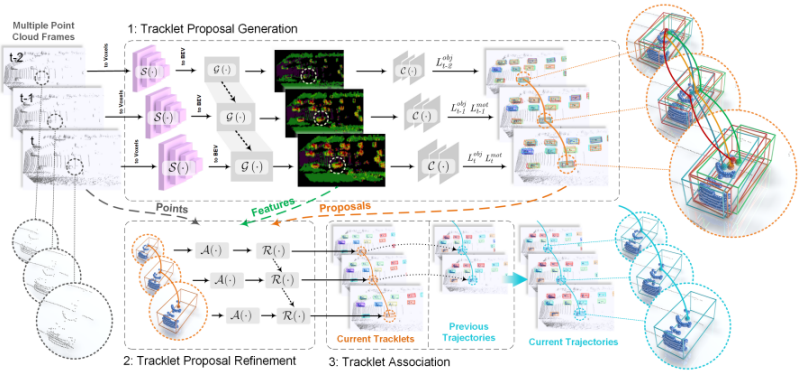


Figure 3.14: Overview of PC-TCNN Method, extracted from [49].

Table 3.1: Comparison of MOT methods performance on KITTI dataset for 'Car' class [10].

Method	**HOTA** ↑	**DetA** ↑	**AssA** ↑	**DetRe** ↑	**DetPr** ↑	**AssRe** ↑	**AssPr** ↑	**LocA** ↑	**Dets**	**IDs**
BeyondPixels [33]	63.75 %	72.87 %	56.40 %	76.58 %	85.38 %	59.05 %	86.70 %	86.90 %	30850	1560
FANTrack [2]	60.85 %	64.36 %	58.69 %	69.17 %	80.82 %	60.78 %	88.94 %	84.72 %	29435	1582
AB3DMot [46]	69.99 %	71.13 %	69.33 %	75.66 %	84.40 %	72.31 %	89.02 %	86.85 %	13619	313
mmMOT [55]	62.05 %	72.29 %	54.02 %	76.17 %	84.89 %	58.98 %	82.40 %	86.58 %	30860	1484
EagerMOT [17]	74.39 %	75.27 %	74.16 %	78.77 %	**86.42 %**	76.24 %	**91.05 %**	87.17 %	16352	724
PC-TCNN [49]	**80.90 %**	**78.46 %**	**84.13 %**	**84.22 %**	84.58 %	**87.46 %**	90.47 %	**87.48 %**	34245	777

Table 3.2: Comparison of MOT methods properties, the approaches are Tracking-By-Detection (TBD) and End-to-End (ETE).

Method	**Modality**	**Approach**
BeyondPixels [33]	LiDAR + RGB	TBD
FANTrack [2]	LiDAR + RGB	TBD
AB3DMot [46]	LiDAR	TBD
mmMOT [55]	LiDAR + RGB	TBD
EagerMOT [17]	LiDAR + RGB	TBD
PC-TCNN [49]	LiDAR + GPS	ETE

Chapter 4

Methodology

This chapter describes the techniques used in the experiments conducted in this thesis. The adopted framework for MOT is introduced at the beginning of the chapter. Lastly, the methodology of the proposed techniques is described in depth in the chapter's final section.

4.1 EagerMOT

As previously stated, EagerMOT [17] is a robust and modular sensor fusion framework for object tracking with SOTA performance. Another advantage of this framework is its adaptability and flexible architecture. Since the only required inputs are the dataset and its corresponding 2D and 3D detections, it is possible to create a multitude of experiments by changing the 3D or 2D detections used as input.

4.2 Proposal

4.2.1 Framework Analysis

Despite obtaining impressive results, the original EagerMOT [17] article only experiments with a small number of 2D and 3D detectors: Point-GNN and Point R-CNN for the 3D source and RRC and Track-RCNN for the 2D source. Therefore, part of this project evaluates the performance of EagerMOT [17] using various 3D and 2D Object detectors for both Car and Pedestrian tracking. The results obtained give insight into the performance of these detectors in the context of object tracking, as well as how the 3D detectors perform in combination with 2D detectors.

4.2.2 Sensor Fusion Occlusion Analysis

One of the most difficult challenges in MOT are occlusions caused by scene structures and other objects that are being tracked, as can be seen in figure 4.1. The latter is most common in the context of autonomous driving since vehicles that are being tracked often occlude other objects in the scene and vice-versa. These occlusions can be classified as partial occlusions or complete

occlusions, depending on how visible the object is to the sensor. Ideally, MOT methods keep track of occluded objects and keep their identification until they are visible again by a sensor.

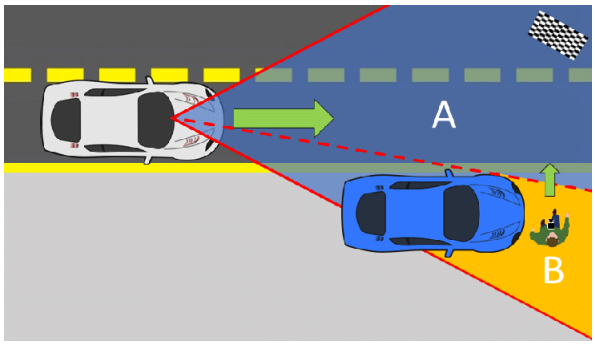


Figure 4.1: Example of a full occlusion of a pedestrian caused by another tracked object, extracted from [27].

As mentioned in section 3.1.1, fusing two sensor modalities has multiple advantages for perception tasks. Additionally, having multiple sensors can mitigate the occlusion problem because of the different nature, perspective, and position of each sensor, making some objects visible to only one sensor.

As mentioned before, EagerMOT [17] makes use of sensor fusion from 2D and 3D sources, which can be beneficial in occlusion mitigation. Therefore, an analysis of how artificial occlusions in one of the sensors affects the overall performance of the framework is conducted. Moreover, the results are compared with the same scenarios but without using the information from the non-occluded sensor, helping us understand if the sensor fusion performed by EagerMOT [17] is mitigating the occlusion problem.

4.3 Experimental Setup

4.3.1 Dataset

The dataset chosen for the project was the KITTI [10] tracking dataset, more specifically, the training split of the dataset. While other datasets commonly used for autonomous driving tasks such as nuScenes [5] offer some advantages over KITTI [10], such as denser information and more categories, the KITTI [10] tracking dataset has more support and, consequently, more readily available 3D and 2D detectors. Furthermore, KITTI [10] continues to serve as a benchmark for all proposed object tracking methods, and it serves as the primary benchmark for comparing new architectures. For the same reason, the training split of the dataset was chosen since other tracking methods often make detections on the training split of the tracking dataset available. Since EagerMOT [17] does not have a training phase, the usual distinction between the training and testing split is not relevant, and all the methods can be fairly compared using it.

The KITTI [10] tracking dataset is made of different sequences with multiple frames each. Each frame contains both 3D and 2D information in the format of a 3D point cloud and an image, correspondingly, as depicted in figures 4.2 and 4.3.

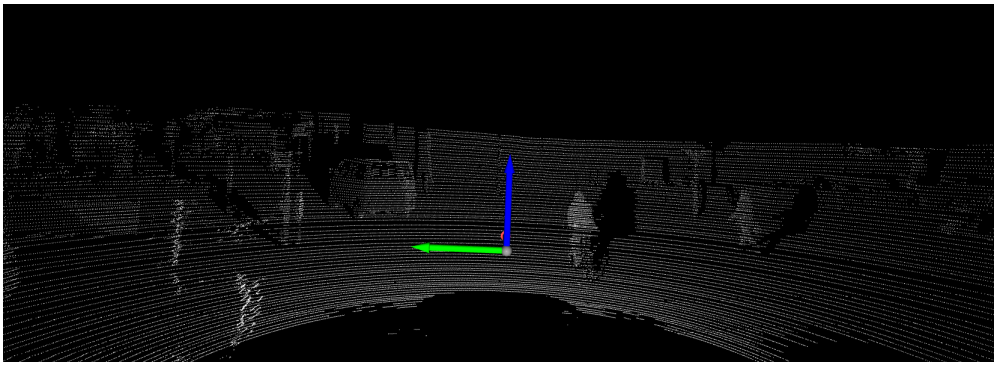


Figure 4.2: Visualisation of a 3D point cloud in the KITTI [10] tracking dataset.

Figure 4.3: 2D image in the KITTI [10] tracking dataset.

The training split consists of 21 sequences containing a total of 8027 frames. A summary of this dataset is shown in table 4.1. The largest sequence contains 1058 frames, while the smallest one contains 77 frames.

Table 4.1: Training split of the KITTI [10] tracking dataset properties.

Property	**Value**
Sequences	21
Frames	8027
Car detections	24070
Car tracks	564
Pedestrian detections	11109
Pedestrian tracks	167

4.3.2 3D and 2D Detections

The object detectors used for the Framework analysis and Sensor Fusion Occlusion Analysis experiments are PV-RCNN [35], Voxel R-CNN [8], Second [51], Second IoU [51], Part-A^2-free [36],

Part-A^2-anchor [36], Point-GNN [37] and PointPillars [20] as 3D sources, and RRC [30] and TrackRCNN [39] for the 2D detections. This set of detectors covers most of the 3D and 2D detectors commonly used in the field of object detection in recent years. All of the 3D detectors are briefly explained in section 3.2. Second IoU is a 3D object detector based on Second [51] but optimized by Intersection-over-Union (IoU). TrackRCNN [39] is a Multi-Object Tracking and Segmentation method, based on the object instance segmentation, Mask R-CNN [14]. An overview of the 3D and 2D detectors, along with their performance in the KITTI [10] test 3D detection benchmark is shown on tables 4.2 and 4.3, respectively.

In order to evaluate the performance of the different model combinations, the HOTA metrics mentioned in section 3.3.1 are used. While all HOTA sub-metrics (detection, association, and location) are analyzed in the results, in order to observe where each combination prevails, the performance ranking of the different models will be only based on HOTA, as it balances the effect of all subcategories into a single unified metric for comparing trackers.

Table 4.2: Comparative results of 3D object detection on the KITTI [10] test 3D detection benchmark. The letters 'E,' 'M,' and 'H' represent easy, moderate, and difficult difficulties, respectively. Values not present are represented by '-'. Cars require an 3D bounding box overlap of 70% and pedestrians require a 3D bounding box overlap of 50%, adapted from [11].

Method	**Type**	**Car**			**Pedestrian**		
		M	E	H	M	E	H
PV-RCNN [35]	Grid-Based	82.01	90.13	77.53	47.02	55.84	42.94
Voxel R-CNN [8]	Grid-Based	81.62	90.9	77.06	-	-	-
Second [51]	Grid-Based	79.46	87.44	73.97	48.96	38.78	34.91
Second IoU [51]	Grid-Based	-	-	-	-	-	-
Part-A^2-free [36]	Grid-Based	78.96	88.48	78.36	-	-	-
Part-A^2-anchor [36]	Grid-Based	79.47	89.47	78.54	51.12	59.72	48.04
Point-GNN [37]	Graph-Based	79.36	87.78	74.15	51.92	43.77	40.14
PointPillars [20]	Grid-Based	74.31	82.58	68.99	51.45	41.92	38.89

Table 4.3: Comparative results of 2D object detection on the KITTI [10] test 3D detection benchmark. The letters 'E,' 'M,' and 'H' represent easy, moderate, and difficult difficulties, respectively. Values not present are represented by '-'. Cars require an overlap of 70% and pedestrians require a 3D bounding box overlap of 50%, adapted from [11].

Method	**Car**			**Pedestrian**		
	M	E	H	M	E	H
RRC [30]	93.40	95.68	87.37	76.61	85.98	71.47
TrackRCNN [39]	-	-	-	-	-	-

4.3.3 Artificial Occlusions

The experiments on sensor fusion occlusion mitigation analysis will be conducted using the best performing pair of 3D and 2D detectors, the pair that has the highest HOTA value, in the Ea-

gerMOT [17] framework Analysis experiments, as it demonstrates EagerMOT [17] best tracking capabilities.

In order to simulate occlusions on the 3D sensor, random frames are removed from the 3D detections used as input. Removing a whole frame of detections mimics every object detected on that frame being occluded from that frame of detections. Consequently, one of the variables of these experiments is the percentage of removed frames, occ_{ratio} If the frames are removed consecutively, objects will be occluded for various frames in a row. Because in real occlusions, objects are often occluded in this manner, the experiments will include another variable, occ_{frames}, representing the number of frames removed in a row. occ_{frames} does not influence the ratio of removed frames, only the pattern on which frames are removed from detections. For example, if $occ_{ratio} = 0.1$ and $occ_{frames} = 5$, the ratio of removed frames will still be 0.1.

These experiments are also made using only the best performant 3D detector, excluding the 2D detector. The values used for occ_{ratio} are 0.00, 0.05, 0.10, 0.15, 0.20, meaning the percentage of removed frames will range from 0% to 20% while occ_{frames} values are 1, 3 and 5, meaning the number of consecutive frames removed will range from 1 to 5 frames in a row.

The HOTA metrics are used to compare the results obtained, similarly to the framework analysis experiment. Special attention is taken into the association sub-metrics (AssA, AssRe), as they directly measure the effect of generated occlusions since these metrics measure how well a tracker links detections over time into the same identities.

Additionally, observations on EagerMOT [17]'s fusion phase during these experiments are done. More specifically, the percentage of fused instances (with 3D and 2D information), instances with only 3D, and instances with only 2D information in each experiment. These results help visualize how the artificial occlusions on one of the sensors impact the fusion phase of the method.

4.3.3.1 Limitations

Due to the way the EagerMOT [17] two-stage data association module functions, tracks can not be created using only 2D information. Thus, experiments where the 2D sensor is artificially occluded while the 3D is not are not relevant since it is not possible to compare the results to those without using the non-occluded sensor.

4.3.4 Training Configurations

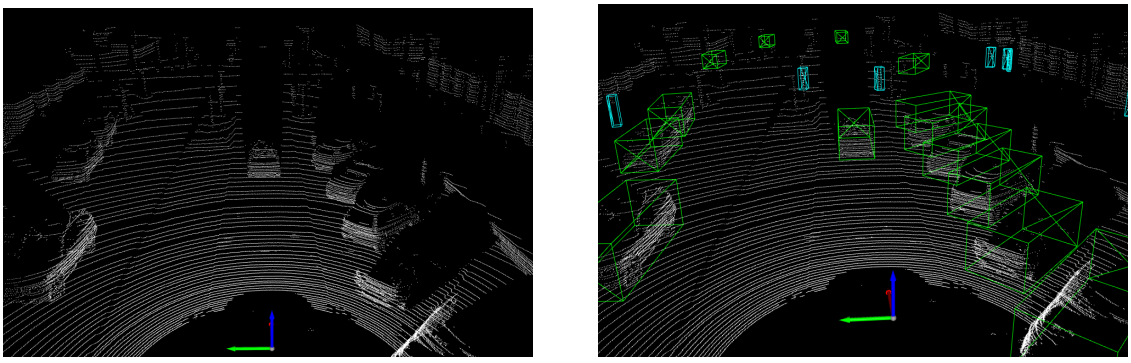
This section discusses the training configurations of the models, including the hardware, framework, data, and hyperparameters.

4.3.4.1 Hardware

The project was developed using a machine equipped with an AMD EPYC 75F3 32-Core Processor, 8 NVIDIA A100 Tensor Core Graphics Processing Units (GPUs), with 40GB each and 1TB of RAM. This machine served as the development of all the experiments made, as well as 3D point cloud experiments.

4.3.4.2 Framework

Part of the 2D and 3D detections from the detectors that are used in the KITTI [10] tracking dataset were available directly from other researchers. In particular, PV-RCNN [35] detections were available in P3CT [48] repository, RRC [30] 2D detections were available in MOTSFusion repository [23], TrackRCNN [43] detections were available in its repository and Point-GNN [37]'s detections were available in EagerMOT [17] own repository. The remaining methods detections were all conducted using the OpenPCDet [41] repository. OpenPCDet [41] is a 3D object detection codebase created by Open-MMLab aimed to solve the lack of a general 3D object detection codebase. OpenPCDet, which was released in 2019, supports both single and multi-modality state-of-the-art 3D object detectors. The software was modified in order to work with the KITTI [10] tracking dataset directly. Voxel R-CNN [8] model was only trained for the 'Car' class. Therefore, it is not evaluated in the 'Pedestrian' experiments. Figure 4.4 depicts an example of detections obtained in the tracking dataset using one of the supported methods.



(a) 3D Point cloud (b) Detections on the 3D Point Cloud

Figure 4.4: Example of detections obtained with OpenPCDet [41] framework using PVRCNN 3D detector, on the KITTI [10] tracking dataset. Green bounding boxes represent Cars, while light blue bounding boxes represent pedestrians.

The models used were available in a model zoo in the OpenPCDet [41] repository, presented in table 4.4, along with their training times. All models were trained with 8 GTX 1080Ti GPUs.

Table 4.4: OpenPCDet 3D Detector pre-trained models.

Model	**Training Time**
Voxel R-CNN	2.2 hours
Second	1.7 hours
Second-IoU	-
Part-A2-free	3.8 hours
Part-A2-anchor	4.3 hours
PointPillars	1.2 hours

After obtaining the detections for each model, EagerMOT [17] is used to obtain the tracking results. The format of some detections file structures or individual detection formats varies,

consequently, support for these formats was added to the EagerMOT [17] software.

TrackEval [16] was used in order to evaluate the results of each experiment. TrackEval provides code for a number of different tracking evaluation metrics, including the HOTA metrics, as well as support for multiple benchmarks. The output of the evaluation is detailed and includes plots for easy reading.

For the occlusion experiments, a tool was developed that takes as an input the 3D detections, occ_{ratio}, and occ_{frames} and outputs the respective detections with removed frames.

4.3.4.3 Hyperparameters

The EagerMOT [17] framework was run using the optimal hyperparameters for the KITTI [10] tracking dataset according to the authors, depicted in table 4.5. The overlap fusion threshold was set at $\theta_{fusion} = 0.01$, the assossiation maximum 3D threshhold and 2D threshold was set at $\theta_{3d} = 0.01$ and $\theta_{2d} = 0.3$, respectively. The maximum age of any track (2D or 3D) was set at $Age_{max} = 3$ and the maximum lifespan for a track without 2D information was set to $Age_{2d} = 3$. The minimum confidence threshold for car and pedestrian detections were set at $Threshold_{car} = -3.5$ and $Threshold_{pedestrian} = -0.3$, respectively.

Table 4.5: EagerMOT [17] hyperparameters used in the experiments.

Hyperparameter	**Value**
θ_{fusion}	0.01
θ_{3d}	0.01
θ_{2d}	0.3
Age_{max}	3
Age_{2d}	3
$Threshold_{car}$	-3.5
$Threshold_{pedestrian}$	-0.3

Chapter 5

Results and Discussion

This chapter describes and discusses the results obtained from the framework analysis and sensor fusion occlusion analysis.

5.1 Framework Analysis

This section compares and discusses the tracking performance obtained by the different 3D and 2D detectors using EagerMOT [17] as the framework.

Table 5.1 depicts the performance of EagerMOT [17] using different 3D detectors and RRC [30] as the 2D detector, in the 'Car' class. Point-GNN [37] is the best performing 3D detector coupled with RRC [30] with $HOTA = 78.037$. This combination outperforms the other 3D detectors in all three tracking subtasks: detection, association, and localization, with higher DetA, AssA, and LocA, respectively. This combination also boasts the lower number of total detections ($Dets = 23977$) and IDs ($IDs = 1023$). Compared to the ground-truth value (section 4.3.1), the number of total detections is slightly lower ($GTDets = 24070$), while the track values are almost double ($GTIDs = 564$), indicating that some tracks got split in multiple. Following Point-GNN [37], the best performing detectors are Voxel R-CNN [8] and PV-RCNN [35]. After that Part-A^2-free [36] outperforms the anchor counterpart despite having a lower Association score. The bottom performing 3D models are Second IoU [51], Second [51] and PointPillars [20]. We can observe that the HOTA score is heavily impacted by the detection sub-metric, as it is the component that varies the most between Detectors (from $DetA = 76.802$ in Point-GNN [37] to $DetA = 45.572$ in PointPillars [20]). This can be partly explained by the increasing number of IDs as the detection sub-metric and, consequently, the HOTA metric decreases across 3D Detectors, indicating a large number of False Positive (FP) instances on lower-performing detectors.

Table 5.2 shows the same experiment as the previous table but depicts the results in the 'Pedestrian' class of the KITTI [10] tracking dataset. Similarly to the 'Car' class, the combination of Point-GNN [37] along with RRC [30] was the best performing with $HOTA = 47.449$.

Table 5.1: Comparison of EagerMOT [17] tracking performance in KITTI [10] tracking dataset 'Car' class using different 3D detectors and RRC [30] as the 2D detector.

3D Detector	HOTA ↑	DetA ↑	AssA ↑	DetRe ↑	DetPr ↑	AssRe ↑	AssPr ↑	LocA ↑	Dets	IDs
Point-GNN [37]	**78.037**	**76.802**	**79.515**	**83.964**	**84.29**	**83.202**	**89.931**	**88.796**	23977	1023
Voxel R-CNN [8]	61.971	54.652	70.664	77.988	59.682	80.642	79.541	84.728	31453	3629
PV-RCNN [35]	59.794	51.564	69.742	78.312	55.824	80.764	78.424	84.592	33766	4038
Part-A^2-free [36]	56.21	46.44	68.329	76.744	51.125	79.39	78.842	85.713	36132	5819
Part-A^2-anchor [36]	54.772	43.235	69.787	77.694	46.417	80.81	78.397	84.458	40289	7767
Second IoU [51]	50.425	37.203	68.612	78.734	39.223	80.072	77.134	84.229	48317	10906
Second [51]	50.107	36.158	69.705	76.096	38.788	81.108	77.805	84.466	47221	11059
PointPillars [20]	45.572	30.403	68.521	77.919	31.851	79.872	76.912	83.957	58884	12648

Contrary to the 'Car' class, this combination does not completely outperform in every HOTA sub-metric, having the lowest location performance of every combination with $LocA = 73.435$, meaning this combination has the lowest spacial alignment between predicted and ground-truth detections. Once again the tracking performance using Part-A^2-free [36] and Part-A^2-anchor [36] is similar, with Part-A^2-anchor [36] performing slightly better overall in the 'Pedestrian' category ($HOTA = 39.813$ compared to Part-A^2-anchor [36] $HOTA = 38.869$). As expected, PV-RCNN [35] has worse tracking performance relative to other detectors in the 'Pedestrian' class, as its detection performance, seen in table 4.2, is noticeably weaker than in the 'Car' class. The worst performing Detector combinations are the same as in the 'Car' class with Second IoU [51] followed by Second [51] and PointPillars [20].

Table 5.2: Comparison of EagerMOT [17] tracking performance in KITTI [10] tracking dataset 'Pedestrian' class using different 3D detectors and RRC [30] as the 2D detector.

3D Detector	HOTA ↑	DetA ↑	AssA ↑	DetRe ↑	DetPr ↑	AssRe ↑	AssPr ↑	LocA ↑	Dets	IDs
Point-GNN [37]	**47.449**	**48.333**	**46.963**	59.604	**56.028**	**50.583**	69.201	73.435	11818	701
Part-A^2-free [36]	39.813	42.55	37.624	57.921	50.633	40.887	**71.064**	74.521	12708	1883
Part-A^2-anchor [36]	38.869	40.149	37.978	**61.366**	45.291	41.547	70.232	74.598	15052	2244
PV-RCNN [35]	36.376	36.719	36.34	58.947	42.243	39.523	70.493	**74.682**	15502	2552
Second IoU [51]	28.695	24.104	34.398	57.61	26.56	37.588	69.549	74.002	24096	5547
Second [51]	28.616	22.608	36.404	58.832	24.561	39.974	69.45	74.086	26610	6359
PointPillars [20]	23.747	15.537	36.557	57.259	16.552	40.276	67.773	73.747	38429	9405

Tables 5.3 and 5.4 show the EagerMOT [17] tracking performance in KITTI [10] tracking dataset 'Car' and 'Pedestrian' classes, respetively, using different 3D detectors and TrackRCNN [39] as the 2D detector. Overall the results using TrackRCNN [39] instead of RRC [30] as the 2D Detector are similar. The 3D detector performance rankings for both 'Car' and 'Pedestrian' classes remain unchanged except for Second [51] and Second IoU [51] in the 'Car' class, which switched places, albeit with a similar HOTA score ($HOTA = 50.432$ and $HOTA = 50.317$, for Second [51] and Second IoU [51], respectively). Combining the 2D sensor with detections from Point-GNN [37] still have the best tracking results with $HOTA = 78.332$ and $HOTA = 48.196$, in 'Car' and 'Pedestrian' classes, respectively. This combination boasts the best overall performance with a 0.295 and 0.747 increase in HOTA compared to using RRC [30] as the 2D detector, in 'Car' and 'Pedestrian' classes, respectively. Similarly to the RRC [30] results, the detection submetric (DetA) influences the most the HOTA score ranging from $DetA = 76.158$ in Point-GNN [37] to

$DetA = 30.232$ in PointPillars [20] in the 'Car' class. The best performing combinations are, again, the ones with less number of detections in both 'Car' and 'Pedestrian' classes. The number of detections using the worst 3D model (PointPillars [20]) is more than triple the number of detections obtained with Point-GNN [37] ($Dets = 38777$ compared to $Dets = 12127$) in the 'Pedestrian' class, compared to 11109 ground truth detections.

Table 5.3: Comparison of EagerMOT [17] tracking performance in KITTI [10] tracking dataset 'Car' class using different 3D detectors and TrackRCNN [39] as the 2D detector.

3D Detector	HOTA ↑	DetA ↑	AssA ↑	DetRe ↑	DetPr ↑	AssRe ↑	AssPr ↑	LocA ↑	Dets	IDs
Point-GNN [37]	**78.332**	**76.158**	**80.784**	**84.661**	**82.789**	**85.087**	**89.186**	**88.706**	24614	1003
Voxel R-CNN [8]	62.218	54.429	71.509	79.348	58.667	81.762	79.351	84.733	32555	3707
PV-RCNN [35]	59.973	51.243	70.592	78.791	55.194	81.903	78.176	84.541	34361	4035
Part-A^2-free [36]	56.623	46.452	69.322	77.615	50.726	80.442	78.956	85.583	36829	5806
Part-A^2-anchor [36]	54.582	42.918	69.802	78.232	45.878	80.984	78.203	84.483	41045	7748
Second [51]	50.423	36.318	70.268	79.271	38.201	81.698	77.872	84.526	49948	11684
Second IoU [51]	50.317	37	68.68	78.276	39.119	80.262	77.017	84.246	48163	10643
PointPillars [20]	45.603	30.232	68.991	78.348	31.602	80.485	76.856	84.006	59675	12624

The 3D detectors performance discrepancy between the tracking and detection (table 4.2) tasks might indicate some are better generalized relative to others. Detectors are often fine-tuned to achieve the highest performance in each benchmark dataset and might not be as performant in other datasets, in this case, the KITTI [10] tracking dataset. The graph-based detector Point-GNN [37] outperforms all the other 3D detectors in both 'Car' and 'Pedestrian' classes, despite having one of the lower scores in the 'Car' class in the detection benchmark, showing better adaptability to the tracking dataset compared to others.

Table 5.4: Comparison of EagerMOT [17] tracking performance in KITTI [10] tracking dataset 'Pedestrian' class using different 3D detectors and TrackRCNN [39] as the 2D detector.

3D Detector	HOTA ↑	DetA ↑	AssA ↑	DetRe ↑	DetPr ↑	AssRe ↑	AssPr ↑	LocA ↑	Dets	IDs
Point-GNN [37]	**48.196**	**48.989**	**47.812**	60.909	**55.796**	**51.669**	68.97	73.494	12127	720
Part-A^2-free [36]	40.355	43.16	38.109	59.188	50.505	41.422	**70.94**	74.484	13019	1917
Part-A^2-anchor [36]	39.062	40.276	38.239	**61.821**	45.2	41.843	70.144	74.586	15194	2267
PV-RCNN [35]	36.632	36.95	36.625	59.62	42.189	39.861	70.456	**74.657**	15699	2582
Second IoU [51]	29.238	24.783	34.739	58.562	27.167	37.953	69.405	73.967	23947	5427
Second [51]	28.67	22.383	36.916	59.719	24.149	40.643	69.294	74.084	27472	6628
PointPillars [20]	24.081	15.887	36.781	58.72	16.822	40.533	67.698	73.701	38777	9437

In Table 5.5 we can see the average difference between TrackRCNN [39] and RRC [30] as the 2D detector in the last experiments, in the 'Car' and 'Pedestrian' KITTI [10] tracking dataset classes. The average HOTA value is higher in both classes using TrackRCNN [39] detections, especially the 'Pedestrian' class with a 0.381 average increase. For the detection HOTA sub-metric (DetA), TrackRCNN [39] has a lower average score compared to RRC [30] (-0.213) in the 'Car' class but higher in the 'Pedestrian' class (+0.347). TrackRCNN [39] outperforms RRC [30] in the association sub-metric with a 0.634 and 0.422 average increase in the 'Car' and 'Pedestrian' classes, respectively. The location sub-metric average difference is almost negligible, with only a -0.014 difference in both classes between TrackRCNN [39] and RRC [30]. The average number of detections is higher on average using TrackRCNN [39] 2D detections with 893.875 and 288.571

increases over RRC [30], in the 'Car' and 'Pedestrian' classes, respectively. The number of average IDs is also higher with TrackRCNN [39] in both classes, partly because of the increased number of detections.

Table 5.5: Average difference in EagerMOT [17] tracking performance between TrackRCNN [39] and RRC [30] as 2D detector in the 'Car' and 'Pedestrian' KITTI [10] tracking dataset classes. Positive values indicate higher metric value using TrackRCNN [39].

Object Classes	HOTA ↑	DetA ↑	AssA ↑	DetRe ↑	DetPr ↑	AssRe ↑	AssPr ↑	LocA ↑	Dets	IDs
Car	0.148	-0.213	0.634	0.886	-0.628	0.845	-0.171	-0.014	893.875	45.125
Pedestrian	0.381	0.347	0.422	1.000	-0.006	0.507	-0.122	-0.014	288.571	41.000

5.2 Sensor Fusion Occlusion Analysis

In this section, the results from the occlusion experiments are presented and discussed. These experiments were made using EagerMOT [17] with Point-GNN [37] as the 3D detector and TrackRCNN [39] as the 2D detector, since it was the best performing setup experimented in section 5.1, with $HOTA = 78.332$ and $HOTA = 48.196$ in the 'Car' and 'Pedestrian' classes, respectively. The full results are presented in appendix A.

Figure 5.1 shows how varying the occ_{ratio} (ratio of occluded frames) from 0 to 0.2 in 3 different occ_{frames} configurations ($occ_{frames} = 1$, $occ_{frames} = 3$ and $occ_{frames} = 5$) affects the overall HOTA score of EagerMOT [17] with and without using a 2D detector, in the 'Car' class. Without any occlusion ($occ_{ratio} = 0$) the difference in HOTA between using and not using 2D detections is not substantial with a 1.817 HOTA difference ($HOTA = 78.332$ with TrackRCNN [39] and $HOTA = 76.515$ without). With $occ_{frames} = 1$, the HOTA remains stable by varying the occ_{ratio} up to 0.2, while using 2D detections ($HOTA = 78.022$ with $occ_{ratio} = 0.05$ to $HOTA = 75.072$ with $occ_{ratio} = 0.20$). On the other hand, without using 2D detections, the HOTA score drops significantly with each increment of occ_{ratio} ($HOTA = 72.975$ with $occ_{ratio} = 0.05$ to $HOTA = 56.772$ with $occ_{ratio} = 0.20$). The difference in performance between using and not using 2D detections is increased by the occlusion ratio (1.817 with $occ_{ratio} = 0.00$ to 18.300 with $occ_{ratio} = 0.20$). The results with $occ_{frames} = 3$ and $occ_{frames} = 5$ are similar, with the major difference being a general drop off in performance with every occ_{ratio} compared to $occ_{frames} = 1$.

In figure 5.2 we can better visualize how varying occ_{frames} affects performance with $occ_{ratio} = 0.05$ and $occ_{ratio} = 0.15$. For both configurations, the performance lowers when occ_{frames} is increased from 1 to 3 with and without using TrackRCNN [39]. When increasing occ_{frames} from 3 to 5, the HOTA value using TrackRCNN [39] stabilizes, while the setup not using 2D information continues to have lower performance.

In figure 5.3 the same experiment shown in figure 5.1 is done, but the performance is evaluated with association recall (AssRe). The degree to which predicted trajectories cover ground-truth trajectories is measured by association recall. A low AssRe, for example, will result when a tracker divides an object into many predicted tracks. This is a useful metric to analyze the effect of occlusions, as occlusions often make the tracker split an object track into two, before the occlusion

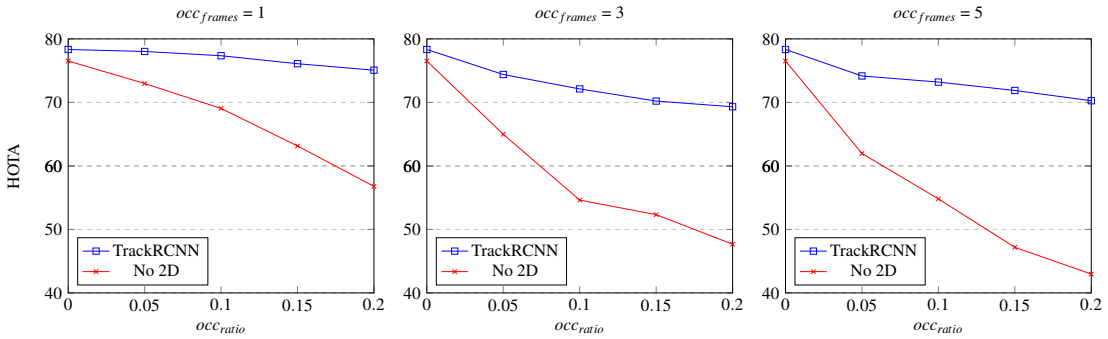


Figure 5.1: Comparison of EagerMOT [17] HOTA in KITTI [10] tracking dataset 'Car' class, with artificially occluded Point-GNN [37] detections with $occ_{frames} = 1$, $occ_{frames} = 3$ and $occ_{frames} = 5$ and varying occ_{ratio} with and without using TrackRCNN [39] as 2D detector.

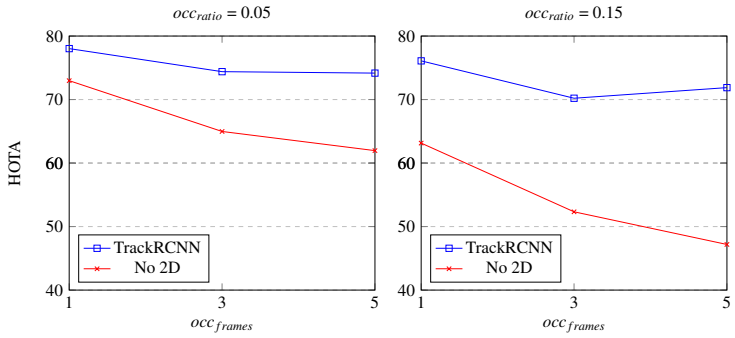


Figure 5.2: Comparison of EagerMOT [17] HOTA in KITTI [10] tracking dataset 'Car' class, with artificially occluded Point-GNN [37] detections with $occ_{ratio} = 0.05$, $occ_{ratio} = 0.15$ and varying occ_{frames} with and without using TrackRCNN [39] as 2D detector.

and after, wrongly recognizing it as a new object. We can observe that these results are extremely similar to those using HOTA as the metric, indicating that the association sub-metric is heavily correlated with the final performance. When comparing AssRe instead of HOTA, we can make similar observations, most important being the major drop in performance the setups without 2D detections have while occ_{ratio} increases, compared to those using TrackRCNN [39] 2D detections. For example, with $occ_{frames} = 5$ the AssRe value ranges from 85.087 to 71.126, with $occ_{ratio} = 0$ to $occ_{ratio} = 0.20$ with TrackRCNN [39] while the setup not using 2D detections ranges from 80.945 to 31.063, with $occ_{ratio} = 0$ to $occ_{ratio} = 0.20$. The difference in AssRe in this configurations starts at 4.142 with $occ_{ratio} = 0$, up to 40.063 with $occ_{ratio} = 0.20$. Similar results can be seen with $occ_{frames} = 1$ and $occ_{frames} = 3$. These results indicate that increasing occ_{ratio} has an enormous impact in dividing objects into multiple tracks as a result of increasing occlusions when not using the 2D sensor in the 'Car' class. On the other hand, when using a 2D sensor with the same occlusion properties, this issue is mitigated. This can be further visualized in figure 5.4, where the visualized metric is IDs. With $occ_{frames} = 1$ the number of IDs using TrackRCNN [39] stays almost the same across the different occ_{ratio} values, while the setup without 2D detections increases from $IDs = 1081$ to $IDs = 1209$, indicating some tracks are divided into multiple. With

$occ_{frames} = 3$ and $occ_{frames} = 5$ this effect is amplified in setups without 2D information, with maximum IDs values of 1720 and 2266. respectively. On the other hand, the experiments with TrackRCNN [39] had maximum ID values of 1174 and 1195, respectively, confirming the attenuation of occlusions using the 2D sensor.

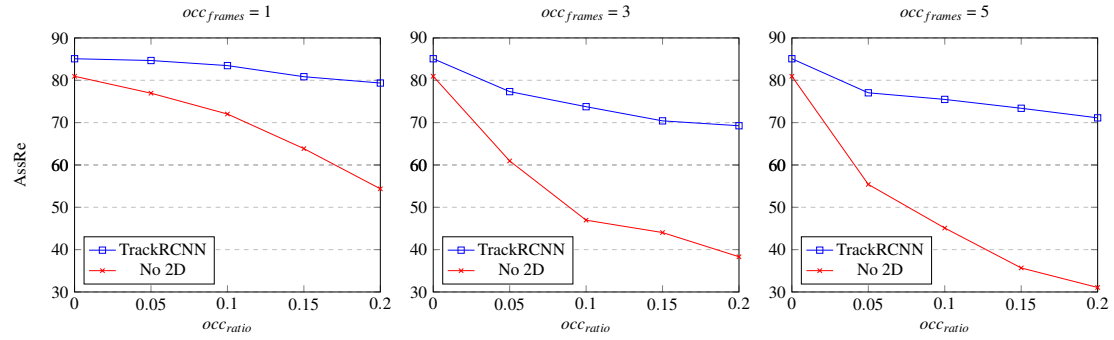


Figure 5.3: Comparison of EagerMOT [17] AssRe in KITTI [10] tracking dataset 'Car' class, with artificially occluded Point-GNN [37] detections with $occ_{frames} = 1$, $occ_{frames} = 3$ and $occ_{frames} = 5$ and varying occ_{ratio} with and without using TrackRCNN [39] as 2D detector.

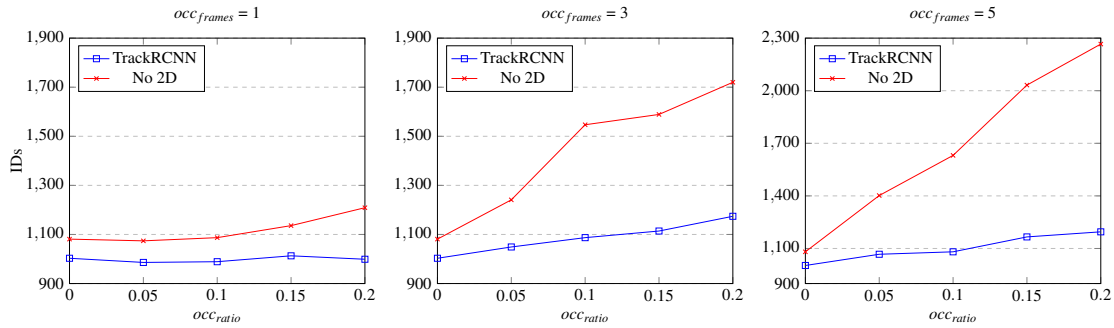


Figure 5.4: Comparison of EagerMOT [17] IDs in KITTI [10] tracking dataset 'Car' class, with artificially occluded Point-GNN [37] detections with $occ_{frames} = 1$, $occ_{frames} = 3$ and $occ_{frames} = 5$ and varying occ_{ratio} with and without using TrackRCNN [39] as 2D detector.

For the 'Pedestrian' class, figure 5.5 shows how varying the occ_{ratio} in 3 different occ_{frames} configurations affects the overall HOTA score of EagerMOT [17] with and without using a 2D detector. Without any occlusion ($occ_{ratio} = 0$) the HOTA values for the setups with and without 2D information is almost equal (48.196 and 47.449, respectively). With $occ_{frames} = 1$, the setup without 2D information follows similar results as the experiments with the 'Car' class, dropping in performance as occ_{ratio} increases, however the setup using TrackRCNN [39] failed to keep a stable HOTA score with increasing occ_{ratio}, as in the 'Car' class, ranging from $HOTA = 47.449$ with $occ_{ratio} = 0$ to $HOTA = 33.077$ with $occ_{ratio} = 0.20$. The results obtained with $occ_{frames} = 3$ and $occ_{frames} = 5$ are similar, but with higher performance drops with increasing occ_{ratio}, including the setup using TrackRCNN [39]. While the performance is still higher than the without 2D information counterpart in every experiment, the margin is not as high and the percentage of performance loss with each occ_{ratio} step is similar without using 2D detections.

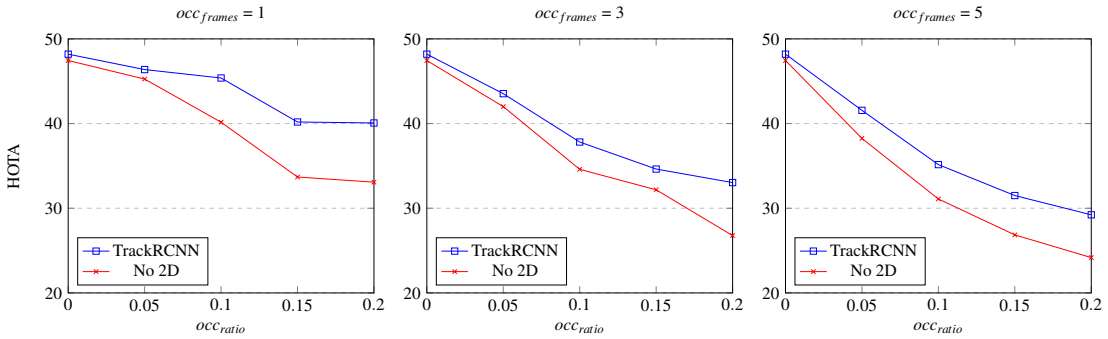


Figure 5.5: Comparison of EagerMOT [17] HOTA in KITTI [10] tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN [37] detections with $occ_{frames} = 1$, $occ_{frames} = 3$ and $occ_{frames} = 5$ and varying occ_{ratio} with and without using TrackRCNN [39] as 2D detector.

With figure 5.6 we can see how the association recall (AssRe) metric behaves with the previous experiment. This is useful to determine if the HOTA improvements in the 'Pedestrian' class are only from other sub-metrics (Location and Detection) or are impacted by the association sub-metric, which is a more direct metric to the issue of occlusion mitigation. Overall, while the AssRe value is always higher on the setup with 2D information, the margin is small, always being close to the baseline comparison with no occlusions ($AssRe = 51.669$ and $AssRe = 50.583$, with and without using TrackRCNN [39], respectively, with a 1.086 difference). Compared to the 'Car' class experiments, the 2D information does not seem to prevent track splitting nearly as much. For example, for $occ_{frames} = 5$ and $occ_{ratio} = 0.20$, the 'Car' class improved the AssRe score 128.97% by using 2D information, while the 'Pedestrian' class improved 21.28%. Lackluster 2D detections compared to the 3D 'Pedestrian' detections obtained with Point-GNN [37] could be a possible explanation for such observations. There is still mitigation to the occlusion problem in the 'Pedestrian' class, with fewer tracks being split by occlusions but not as significant as in the 'Car' class.

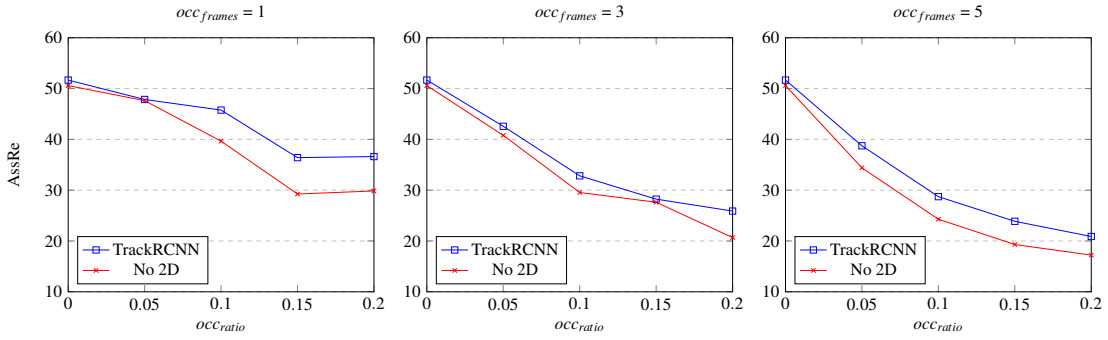


Figure 5.6: Comparison of EagerMOT [17] AssRe in KITTI [10] tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN [37] detections with $occ_{frames} = 1$, $occ_{frames} = 3$ and $occ_{frames} = 5$ and varying occ_{ratio} with and without using TrackRCNN [39] as 2D detector.

5.2.1 Fusion Analysis

During the occlusion experiments, the fusion module statistics were stored to observe how the instance fusion changed with the different artificial occlusion parameters occ_{frames} and occ_{ratio}. In table 5.6 we can see how changing occ_{ratio}, affects the percentage of instances with 3D and 2D information, instances with only 3D information and instances with only 2D information, with $occ_{frames} = 3$. By increasing occ_{ratio}, the percentage of instances with 3D information ('3D&2D' and '3D') lowers, while instances with only 2D information have a higher percentage. By removing 3D detections, instances that previously had information from both sensors now only have 2D information. On the other hand, instances that only had 3D information and were removed are completely lost. The results with other occ_{frames} are similar to those in table 5.6 and are presented in appendix A.

Table 5.6: Comparison of EagerMOT [17] instance association results in KITTI [10] tracking dataset , with artificially occluded Point-GNN [37] detections with $occ_{frames} = 3$ and varying occ_{ratio} using TrackRCNN [39] as 2D detector. '3D&2D', '3D' and '2D' represent instances with 3D and 2D information, instances with only 3D information and instances with only 2D information, respectively.

occ_{ratio}	**3D&2D**	**3D**	**2D**
0.00	51.95%	2.31%	45.74%
0.05	50.05%	2.14%	47.81%
0.10	48.73%	2.09%	49.18%
0.15	47.97%	2.07%	49.96%
0.20	43.61%	1.85%	54.54%

Chapter 6

Conclusions and Future Work

In recent years, the automotive and tech industries have been focusing on the development of autonomous driving. Self-driving vehicles must accurately assess their surroundings to react and make judgments. As a result, Multi-Object-Tracking is a crucial task in autonomous driving technology, making it possible for the system to be aware of the scene both spatially and temporally.

Self-driving vehicles are often equipped with LiDAR and Camera sensors which provide 3D and 2D information in the form of 3D Point Clouds and Images, respectively. Most MOT methods are developed to only use information from one of these sensors, not relying on the advantages provided by both sensors. RGB images have a richer signal, range, and texture information, while 3D Point Clouds are less affected by weather variation and illumination changes and provide 3D information. Furthermore, MOT methods can reduce the impact caused by occlusions on one of the sensors by using information from multiple sensors. EagerMOT [17] is a modular and adaptable MOT method that uses both 2D and 3D information, obtaining State-of-the-Art performance. Although the only required inputs are the detections provided by the 2D and 3D detectors, the original paper's authors only experimented with a limited amount of detectors.

This thesis analyses the tracking performance of EagerMOT [17] using multiple combinations of widely used 2D and 3D object detectors. The KITTI [10] tracking dataset and the HOTA tracking metrics were used to compare the performance between setups. A combination of Point-GNN [37] as the 3D detector and using the 2D detections from TrackRCNN [39] outperformed every other combination in all HOTA sub-metrics (Association, Detection, and Localization) in the 'Car' class while being the best overall setup in the 'Pedestrian' class.

Furthermore, an analysis of how sensor fusion mitigates occlusions was done by artificially occluding detections from the 3D sensor in EagerMOT [17]. Comparing results from setups with and without using 2D information, we observe that the performance difference relative to the setup without occlusions increases significantly in the setup using a 2D sensor (from a 2.347% HOTA difference without occlusions to 48.194% HOTA difference in the setup with a higher ratio and

more sequential occlusions), indicating a reduction of performance loss as a consequence of sensor fusion.

This dissertation's study and this specific topic of investigation could be supplemented in a variety of ways. The following are some prospective research topics:

1. Analyse EagerMOT [17] using larger and more robust datasets, such as nuScenes [5], to determine how different detectors perform in more extreme conditions, with illumination and weather changes.

2. Experiment with other State-of-the-Art 2D and 3D detectors such as 3DSSD [53]. While these methods were not at the time supported by OpenPCDet [41], they are methods with outstanding detection performance that should perform well in the tracking task using EagerMOT.

3. Develop and experiment with a more sophisticated occlusion tool that accurately represents real world occlusions. This tool should use ground truth data to occlude specific objects in the scene and not whole frames of objects. Using the output from this tool, occlusion analysis would be more accurate compared to real-world occlusions.

4. Conduct the occlusion experiments with occlusions in the 2D sensor. While it was impossible to conduct these experiments natively, a slight modification to the EagerMOT [17] software should make this analysis possible.

5. Compare the sensor fusion occlusion analysis performed in EagerMOT [17] with other sensor fusion MOT methods.

References

[1] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine learning from theory to algorithms: an overview. In *Journal of physics: conference series*, volume 1142, page 012012. IOP Publishing, 2018.

[2] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433. IEEE, 2019.

[3] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[4] Facundo Bre, Juan M Gimenez, and Víctor D Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158:1429–1441, 2018.

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[6] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38(1):68–86, 2020.

[7] Ian Cherabier, Christian Häne, Martin R. Oswald, and Marc Pollefeys. Multi-label semantic 3d reconstruction using voxel blocks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 601–610, 2016.

[8] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021.

[9] Yosra Dorai, Sami Gazzah, Frederic Chausse, and Najoua Essoukri Ben Amara. Tracking multi-object using tracklet and faster r-cnn: Phd forum. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 222–223, 2016.

[10] A. Geiger, P. Lenz, and R. Urtasun. **Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite**. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Welcome to the kitti vision benchmark suite! `http://www.cvlibs.net/datasets/kitti/index.php`, 2012.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[13] Ryan Harrington, Carmine Senatore, John Scanlon, and Ryan M Yee. The role of infrastructure in an automated vehicle future. *The Bridge*, 48(2), 2018.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[15] John. A project for 3d multi-object tracking with python. `https://pythonawesome.com/a-project-for-3d-multi-object-tracking-with-python/`, 2021.

[16] Arne Hoffhues Jonathon Luiten. Trackeval. `https://github.com/JonathonLuiten/TrackEval`, 2020.

[17] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taixé. Eagermot: 3d multi-object tracking via sensor fusion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11315–11321. IEEE, 2021.

[18] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[19] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2, 05 2012.

[20] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[21] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[22] Jonathon Luiten. How to evaluate tracking with the hota metricspermalink. `https://autonomousvision.github.io/hota-metrics`, 2021.

[23] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 5(2):1803–1810, 2020.

[24] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129(2):548–578, 2021.

[25] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[26] Tashapais medium. `https://tashapais.medium.com/self-learning-computers-and-the-covid-19-vaccine-youre-getting-f591f335a0ee`. Accessed: 2022-02-13.

[27] Andras Palffy, Julian Kooij, and Dariu Gavrila. Occlusion aware sensor fusion for early crossing pedestrian detection. pages 1768–1774, 06 2019.

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[30] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5420–5428, 2017.

[31] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[33] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna. **Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking**. *CoRR*, abs/1802.09298, 2018.

[34] S Shi, X Wang, H PointRCNN Li, et al. 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA*, pages 16–20, 2019.

[35] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[36] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2647–2664, 2020.

[37] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020.

[38] Shreya. Convolutional neural network(cnn). `https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-networkcnn/`, 2022.

[39] Bing Shuai, Andrew G Berneshawi, Davide Modolo, and Joseph Tighe. Multi-object tracking with siamese track-rcnn. *arXiv preprint arXiv:2004.07786*, 2020.

[40] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, 2015.

[41] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020.

[42] Pete Thomas, Andrew Morris, Rachel Talbot, and Helen Fagerlind. Identifying the causes of road crashes in europe. *Annals of advances in automotive medicine*, 57:13, 2013.

[43] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7942–7951, 2019.

[44] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9782–9792, 2021.

[45] Zhiyu Wang, Bin Dai, and Hao Fu. A fast approach for vehicle-like region proposal based on 3d lidar data. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 508–512. IEEE, 2015.

[46] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366. IEEE, 2020.

[47] Wikimedia commons. `https://commons.wikimedia.org/wiki/File:Detected-with-YOLO--Schreibtisch-mit-Objekten.jpg`. Accessed: 2022-02-15.

[48] Hai Wu, Wenkai Han, Chenglu Wen, Xin Li, and Cheng Wang. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[49] Hai Wu, Qing Li, Chenglu Wen, Xin Li, Xiaoliang Fan, and Cheng Wang. Tracklet proposal network for multi-object tracking on point clouds. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1165–1171, 2021.

[50] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 924–933. IEEE, 2017.

[51] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[52] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019.

[53] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020.

[54] Georgios Zamanakos, Lazaros Tsochatzidis, Angelos Amanatiadis, and Ioannis Pratikakis. A comprehensive survey of lidar-based 3d object detection methods with deep learning for autonomous driving. *Computers & Graphics*, 99:153–181, 2021.

[55] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2365–2374, 2019.

[56] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[57] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

Appendix A

Sensor Fusion Occlusion Results

Table A.1: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Car' class, with artificially occluded Point-GNN detections with $occ_{frames} = 1$, varying occ_{ratio} and using TrackRCNN as 2D detector.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	78.332	76.158	80.784	84.661	82.789	85.087	89.186	88.706	24614	1003
0.05	78.022	75.927	80.386	84.291	82.798	84.668	89.047	88.596	24504	986
0.10	77.348	75.701	79.243	83.872	82.867	83.467	89.065	88.507	24362	989
0.15	76.092	75.638	76.768	83.498	83.101	80.842	89.056	88.377	24185	1013
0.20	75.072	75.205	75.144	83.284	82.714	79.351	88.655	88.266	24236	999

Table A.2: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Car' class, with artificially occluded Point-GNN detections with $occ_{frames} = 3$, varying occ_{ratio} and using TrackRCNN as 2D detector.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	78.332	76.158	80.784	84.661	82.789	85.087	89.186	88.706	24614	1003
0.05	74.392	75.684	73.332	84.074	82.73	77.322	89.193	88.583	24461	1049
0.10	72.122	74.721	69.835	82.977	82.457	73.76	88.778	88.274	24222	1087
0.15	70.207	74.157	66.696	82.203	82.539	70.413	88.6	88.217	23972	1114
0.20	69.311	73.43	65.669	81.21	82.489	69.261	88.631	87.977	23697	1174

Table A.3: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Car' class, with artificially occluded Point-GNN detections with $occ_{frames} = 5$, varying occ_{ratio} and using TrackRCNN as 2D detector.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	78.332	76.158	80.784	84.661	82.789	85.087	89.186	88.706	24614	1003
0.05	74.164	75.554	73.013	84.007	82.596	77.039	89.082	88.508	24481	1067
0.10	73.198	75.197	71.459	83.538	82.564	75.48	88.88	88.381	24354	1081
0.15	71.877	74.577	69.509	82.652	82.563	73.389	88.828	88.192	24096	1166
0.20	70.262	73.682	67.238	81.773	82.193	71.126	88.357	87.932	23947	1195

Table A.4: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Car' class, with artificially occluded Point-GNN detections with $occ_{frames} = 1$, varying occ_{ratio} without using 2D detections.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	76.515	75.527	77.731	81.726	85.194	80.945	90.551	89.037	23090	1081
0.05	72.975	72.159	74.003	77.85	85.159	76.949	90.57	89.043	22004	1074
0.10	69.048	68.882	69.424	73.986	85.245	72.029	90.646	89.025	20891	1087
0.15	63.143	64.774	61.737	69.24	85.331	63.86	90.919	89.043	19531	1136
0.20	56.772	61.589	52.509	65.8	85.041	54.365	90.988	89.028	18624	1209

Table A.5: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Car' class, with artificially occluded Point-GNN detections with $occ_{frames} = 3$, varying occ_{ratio} without using 2D detections.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	76.515	75.527	77.731	81.726	85.194	80.945	90.551	89.037	23090	1081
0.05	64.975	72.232	58.667	77.921	85.19	60.951	91.192	89.062	22016	1241
0.10	54.617	66.145	45.318	70.986	85.104	46.957	91.848	89.084	20077	1547
0.15	52.325	64.768	42.488	69.205	85.376	44.033	92.029	89.041	19511	1589
0.20	47.668	61.389	37.22	65.416	85.264	38.319	92.472	89.016	18467	1720

Table A.6: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Car' class, with artificially occluded Point-GNN detections with $occ_{frames} = 5$, varying occ_{ratio} without using 2D detections.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	76.515	75.527	77.731	81.726	85.194	80.945	90.551	89.037	23090	1081
0.05	61.96	72.121	53.436	77.726	85.202	55.408	91.405	88.986	21958	1402
0.10	54.807	69.297	43.566	74.511	85.205	45.097	91.906	89.039	21049	1631
0.15	47.183	64.773	34.586	69.369	85.098	35.675	92.549	88.997	19621	2032
0.20	42.975	61.414	30.293	65.513	85.215	31.063	93.097	89.067	18505	2266

Table A.7: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN detections with $occ_{frames} = 1$, varying occ_{ratio} and using TrackRCNN as 2D detector.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	48.196	48.989	47.812	60.909	55.796	51.669	68.97	73.494	12127	720
0.05	46.377	48.982	44.378	60.27	56.345	47.852	69.283	73.486	11883	737
0.10	45.382	48.759	42.657	59.467	56.725	45.765	69.597	73.42	11646	776
0.15	40.185	48.04	34.185	58.018	57.083	36.407	70.625	73.416	11291	873
0.20	40.072	47.646	34.277	57.184	57.22	36.613	70.634	73.315	11102	926

Table A.8: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN detections with $occ_{frames} = 3$, varying occ_{ratio} and using TrackRCNN as 2D detector.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	48.196	48.989	47.812	60.909	55.796	51.669	68.97	73.494	12127	720
0.05	43.53	48.426	39.631	59.536	56.15	42.576	69.71	73.361	11779	792
0.10	37.82	47.206	30.891	57.641	56.219	32.823	70.502	73.335	11390	864
0.15	34.628	46.464	26.6	56.343	56.343	28.235	71.68	73.268	11109	940
0.20	33.031	45.83	24.541	54.951	56.628	25.886	71.71	73.047	10780	987

Table A.9: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN detections with $occ_{frames} = 5$, varying occ_{ratio} and using TrackRCNN as 2D detector.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	48.196	48.989	47.812	60.909	55.796	51.669	68.97	73.494	12127	720
0.05	41.562	48.345	36.242	59.715	55.934	38.734	70.317	73.438	11860	827
0.10	35.154	46.976	26.962	57.453	56.041	28.725	71.604	73.312	11389	984
0.15	31.499	45.64	22.526	55.106	56.312	23.872	72.991	73.153	10871	1203
0.20	29.225	44.877	19.786	54.007	56.314	20.883	73.521	73.174	10654	1292

Table A.10: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN detections with $occ_{frames} = 1$, varying occ_{ratio} without using 2D detections.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	47.449	48.333	46.963	59.604	56.028	50.583	69.201	73.435	11818	701
0.05	45.256	46.453	44.475	56.657	56.151	47.618	69.563	73.417	11209	709
0.10	40.161	44.07	37.111	52.922	56.475	39.646	70.346	73.451	10410	747
0.15	33.689	41.62	27.73	49.437	56.485	29.242	71.598	73.421	9723	815
0.20	33.077	39.284	28.264	46.002	56.757	29.848	71.228	73.402	9004	813

Table A.11: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN detections with $occ_{frames} = 3$, varying occ_{ratio} without using 2D detections.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	47.449	48.333	46.963	59.604	56.028	50.583	69.201	73.435	11818	701
0.05	42.001	46.71	38.214	57.04	56.155	40.782	69.979	73.422	11284	761
0.10	34.596	43.584	28.023	52.508	56.147	29.543	71.588	73.448	10389	847
0.15	32.178	40.369	26.12	47.943	56.069	27.623	72.703	73.367	9499	939
0.20	26.772	37.486	19.767	43.964	56.3	20.671	74.674	73.583	8675	1002

Table A.12: Comparison of EagerMOT tracking performance in Kitti tracking dataset 'Pedestrian' class, with artificially occluded Point-GNN detections with $occ_{frames} = 5$, varying occ_{ratio} without using 2D detections.

occ_{ratio}	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**Dets**	**IDs**
0.00	47.449	48.333	46.963	59.604	56.028	50.583	69.201	73.435	11818	701
0.05	38.249	45.877	32.457	55.891	56.072	34.396	70.95	73.414	11073	828
0.10	31.094	43.09	23.029	51.805	56.103	24.285	72.569	73.389	10258	987
0.15	26.854	40.039	18.555	47.538	56.115	19.303	74.826	73.477	9411	1117
0.20	24.174	37.149	16.403	43.338	56.428	17.218	76.016	73.441	8532	1160

Table A.13: Comparison of EagerMOT instance association results in Kitti tracking dataset, with artificially occluded Point-GNN detections with $occ_{frames} = 1$ and varying occ_{ratio} using TrackR-CNN as 2D detector. '3D&2D', '3D' and '2D' represent instances with 3D and 2D information, instances with only 3D information and instances with only 2D information, respectively.

occ_{ratio}	**3D&2D**	**3D**	**2D**
0.00	51.95%	2.31%	45.74%
0.05	49.33%	2.18%	48.49%
0.10	45.90%	2.07%	52.03%
0.15	44.97%	1.95%	53.08%
0.20	42.75%	1.84%	55.41%

Table A.14: Comparison of EagerMOT instance association results in Kitti tracking dataset , with artificially occluded Point-GNN detections with $occ_{frames} = 3$ and varying occ_{ratio} using TrackR-CNN as 2D detector. '3D&2D', '3D' and '2D' represent instances with 3D and 2D information, instances with only 3D information and instances with only 2D information, respectively.

occ_{ratio}	**3D&2D**	**3D**	**2D**
0.00	51.95%	2.31%	45.74%
0.05	50.05%	2.14%	47.81%
0.10	48.73%	2.09%	49.18%
0.15	47.97%	2.07%	49.96%
0.20	43.61%	1.85%	54.54%

Table A.15: Comparison of EagerMOT instance fusion results in Kitti tracking dataset , with artificially occluded Point-GNN [37] detections with , $occ_{frames} = 5$ and varying occ_{ratio} using TrackRCNN [39]as 2D detector. '3D&2D', '3D' and '2D' represent instances with 3D and 2D information, instances with only 3D information and instances with only 2D information, respectively.

occ_{ratio}	**3D&2D**	**3D**	**2D**
0.00	51.95%	2.31%	45.74%
0.05	50.04%	2.27%	47.69%
0.10	47.97%	2.09%	49.94%
0.15	45.17%	1.96%	52.87%
0.20	39.11%	1.72%	59.16%