DISSERTATION

LEARNED PERCEPTION SYSTEMS FOR SELF-DRIVING VEHICLES

Submitted by

Mohamed Chaabane

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2022

Doctoral Committee:

Advisor: Ross J. Beveridge

Stephen O'Hara
Nathaniel Blanchard
Chuck Anderson
Atadero Rebecca

ABSTRACT

LEARNED PERCEPTION SYSTEMS FOR SELF-DRIVING VEHICLES

Building self-driving vehicles is one of the most impactful technological challenges of modern artificial intelligence. Self-driving vehicles are widely anticipated to revolutionize the way people and freight move. In this dissertation, we present a collection of work that aims to improve the capability of the perception module, an essential module for safe and reliable autonomous driving. Specifically, it focuses on two perception topics: 1) Geo-localization (mapping) of spatially-compact static objects, and 2) Multi-target object detection and tracking of moving objects in the scene.

Accurately estimating the position of static objects, such as traffic lights, from the moving camera of a self-driving car is a challenging problem. In this dissertation, we present a system that improves the localization of static objects by jointly optimizing the components of the system via learning. Our system is comprised of networks that perform: 1) 5DoF object pose estimation from a single image, 2) association of objects between pairs of frames, and 3) multi-object tracking to produce the final geo-localization of the static objects within the scene. We evaluate our approach using a publicly available data set, focusing on traffic lights due to data availability. For each component, we compare against contemporary alternatives and show significantly improved performance. We also show that the end-to-end system performance is further improved via joint training of the constituent models.

Next, we propose an efficient joint detection and tracking model named DEFT, or “Detection Embeddings for Tracking." The proposed approach relies on an appearance-based object matching network jointly learned with an underlying object detection network. An LSTM is also added to capture motion constraints. DEFT has comparable accuracy and speed to the top methods on 2D online tracking leaderboards while having significant advantages in robustness when applied

to more challenging tracking data. DEFT raises the bar on the nuScenes monocular 3D tracking challenge, more than doubling the performance of the previous top method (3.8x on AMOTA, 2.1x on MOTAR). We analyze the difference in performance between DEFT and the next best-published method on nuScenes and find that DEFT is more robust to occlusions and large inter-frame displacements, making it a superior choice for many use-cases.

Third, we present an end-to-end model to solve the tasks of detection, tracking, and sequence modeling from raw sensor data, called Attention-based DEFT. Attention-based DEFT extends the original DEFT by adding an attentional encoder module that uses attention to compute tracklet embedding that 1) jointly reasons about the tracklet dependencies and interaction with other objects present in the scene and 2) captures the context and temporal information of the tracklet's past observations. The experimental results show that Attention-based DEFT performs favorably against or comparable to state-of-the-art trackers. Reasoning about the interactions between the actors in the scene allows Attention-based DEFT to boost the model tracking performance in heavily crowded and complex interactive scenes. We validate the sequence modeling effectiveness of the proposed approach by showing its superiority for velocity estimation task over other baseline methods on both simple and complex scenes. The experiments demonstrate the effectiveness of Attention-based DEFT for capturing spatio-temporal interaction of the crowd for velocity estimation task, which helps it to be more robust to handle complexities in densely crowded scenes. The experimental results show that all the joint models in this dissertation perform better than solving each problem independently.

ACKNOWLEDGEMENTS

I would like to thank my adviser, Dr. Ross Beveridge, for his invaluable guidance, supportive feedback, constant encouragement, and for allowing me to work on exciting research. I owe much to him for the freedom he gave me to pursue my own interests. I am very grateful to him for his input on how to improve my research skills and work. I want to express special thanks to Dr. Stephen O'Hara for providing valuable advice to progress with my research and continually motivating me to develop creative ideas. I greatly enjoyed all of our discussions in the supervision meetings, and his continued guidance is sincerely appreciated. Working under his mentorship helped me significantly to grow as a scientist. Going forward, I am sure that I will take a lot of what I have learned from him in my future endeavors. I would also like to thank other members of my committee, Dr. Nathaniel Blanchard, Dr. Chuck Anderson, and Dr. Atadero Rebecca. They provided me with constructive feedback on my research exam, preliminary exam, and final exam and provided me valuable suggestions to improve my work. I extend my gratitude to Dr. Lionel Gueguen, whom I had the chance to work with for some time during my thesis work. He is one of the most intelligent, kind, dedicated, and inspirational figures in my life. I was lucky to work with him, and I learned a lot from his vast experience.

My deepest gratitude goes to my parents, Samir and Lamia, for all their support and encouragement throughout these years. Similarly, thanks to my brother Omar for his incredible support all along with my PhD. Finally, I especially want to thank my wife and best friend, Ameni, for her kindness and optimism. She always makes me happy, smile, and feel special even during challenging moments and situations. She is the hero of this story. This dissertation is dedicated to her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Autonomous vehicles have grown a lot over the past decade from the rapid development of technologies in computer vision and machine learning. While achieving fully autonomous driving with no human intervention is years away, some autonomous driving technologies have already been integrated into vehicles in the form of Advanced Driver Assistance Systems (ADAS) which have shown to be able to accomplish autonomous driving in several scenarios such as highway driving. These technologies aim to reduce vehicular accidents caused by human error. Every year, around 1.35 million people lose their lives in vehicular accidents, and up to 50 million people are injured [9]. Ninety-four percent of crashes on the roads are due to risky driving and to errors that drivers make while behind the wheel. Autonomous vehicles have the potential to reduce risky driver behaviors. Moreover, fully autonomous vehicles are widely anticipated to free the drivers from managing the progress of the vehicle, enhancing human productivity. On average, Americans spend around 300 hours driving per year, equivalent to more than seven potentially productive weeks of work. Fully autonomous vehicles are also expected to help increase savings in time, energy, and efficiency in transportation, along with many other anticipated benefits.

A self-driving vehicle needs to accomplish various tasks that a human would otherwise perform with errors. However, to achieve these tasks multiple functionalities need to be realized. This dissertation focuses on one of the key functionalities, known as perception, which allows self-driving vehicles to sense their surroundings. Perception is still very challenging due to the rigorously targeted low error rate, and it is a challenge that has drawn the attention of significant efforts within the research community. This dissertation focuses on two driving-related perception topics: 1) Geo-localization (mapping) of spatially-compact static objects and 2) Multi-target object detection and tracking of moving objects in the scene.

1.1 Brief History of Autonomous Driving

Similar to the invention of the gasoline-powered automobile by Karl Benz in 1886, self-driving technology will have a significant impact on our mobility. Since the 1980s, researchers and companies alike have been striving to achieve Level 5 autonomy for self-driving vehicles (i.e., autonomous operation with no human intervention) [10]. Self-driving competitions and road tests held worldwide help identify the capabilities and limits in both hardware and software to bring safer self-driving vehicles on our roads. Even though there are still several challenges to overcome to achieve fully autonomous driving, some autonomous driving technologies have reached commercial success, enriching driving comfort and safety.

In 1986, the first self-driving vehicle was developed by the Navlab team at Carnegie Mellon University. It was designed to be controlled by a computer. The same team achieved another breakthrough in the "No Hands Across America" tour in 1995 by driving 2,849 miles over seven days 98% autonomously. A trained neural network was used to control the steering wheel while human drivers guided its acceleration and braking.

In 2004, The Defense Advanced Research Projects Agency (DARPA) launched a major competition, called DARPA Grand Challenge, which boosted the innovation from universities around this field. Endurance driving across the desert, static obstacles, and following the roadway were the main focus of this challenge. None of the teams was able to achieve the goal. A year later, the competition was repeated on another road (but with similar challenges), and five teams successfully completed the challenge. The DARPA Grand Challenge competitions provided a modern, fair testing opportunity to assess the state-of-the-art in autonomous vehicles in real-world environments, especially as road tests were not allowed at that time.

In 2007, DARPA held a new challenge, called DARPA Urban challenge, focused on real urban environments, to include much of what a typical human driver must perform. This competition offered an excellent opportunity for researchers to evaluate the flaws and challenges of autonomous driving in complex urban environments. The autonomous vehicles in the competition were developed with various capabilities such as localization, object detection,

multi-object tracking and lane keeping. Although this competition was the closest to real-world environments, it was limited to only low-speed driving and uncomplicated scenarios that lack common roadway obstacles, such as traffic lights, bicycles, and pedestrians. Only four teams managed to complete the challenge and still showed reliability issues.

More inventions have become popular over time, and many companies have joined the challenge of developing a fully autonomous commercial vehicle. Some popular nowadays include Aurora, Waymo, Aptiv, Zoox, and units of automobile companies such as Tesla, Toyota, Ford, and General Motors.

1.2 Motivation

Autonomous driving systems are generally composed of five main modules, shown in Figure 1.1: localization, perception, prediction, planning, and vehicle control. The autonomy process is started by localization and perception. Localization computes the global location of the vehicle in order to determine its precise location on the map. Perception allows self-driving vehicles to sense their surroundings. The perception module uses sensors to scan and analyze its surrounding environment and provides situational awareness of dynamic actors (e.g., pedestrians, animals, vehicles). The following module is prediction. The prediction module consists of the prediction of the next trajectories or paths of the dynamic actors as well as the prediction of what actions are feasible to plan a path accordingly. The outputs of the perception and prediction modules are fed to the planning module, which determines possible and ideal routes for the vehicle to drive. Finally, the determined route is translated by the vehicle control module into driving commands for the vehicle, such as acceleration and steering angle.

Based on the various experiments on the field, including the DARPA Urban Challenge, it is safe to say that perception is the most crucial function for safe and reliable driving as it serves to translate the sensor data into useful and consistent information, which affects directly the performances of the planning and vehicle control modules. In this dissertation, we are interested in improving the capability of the perception module to deal with the complexity of real urban

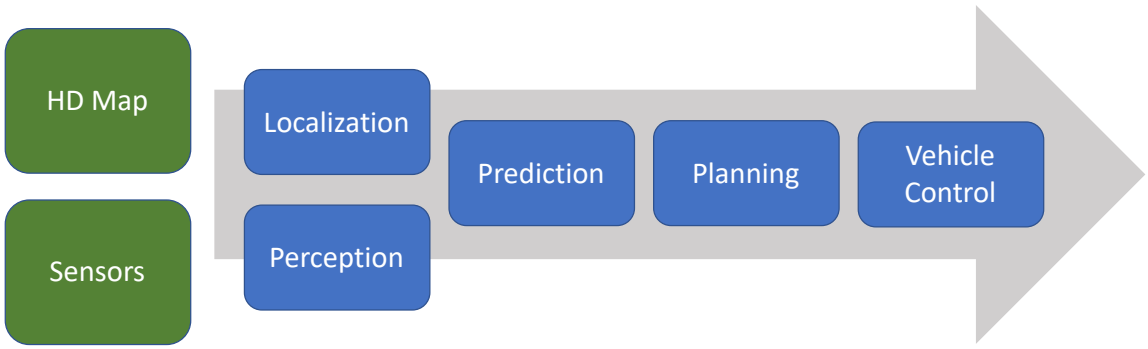

Figure 1.1: Overview of the autonomous driving system.

environments. Specifically, we will tackle one of the critical tasks of vehicle environment perception, the multi-target object detection and tracking of moving objects in the scene. Multi-target object detection and tracking aims to localize all objects of interest (e.g., pedestrians, vehicles) and reliably recover their trajectories in a video sequence while maintaining their identities. Multi-target object detection and tracking can be very challenging in complex urban environments, given the changes of the number of objects across the video frames, object occlusion, appearance changes and high inter-frame motion due to vehicle and objects motion, presence of multiple similar-looking objects, cluttering, detection failures, etc. These challenges motivated us to develop a joint detection and tracking model that is more robust than the current top alternatives in real urban environments. Moreover, for autonomous driving systems, latency is a primary concern. With latency as a key factor, the tracking approach has to be fast enough to maintain the frame rate of the cameras. In many self-driving data sets such as KITTI [11] and nuScenes [3] this is on the order of 10Hz to allow synchronization with LiDAR.

In addition to their live sensors, most autonomy systems for self-driving vehicles heavily depend on HD maps or "High Definition" maps when planning their actions (Figure 1.1). This latter helps ensure the safety and compliance of driving policies. HD maps are very precise inventories that contain detailed information on static actors such as road lanes, boundaries, traffic signs and lights. The knowledge of the static actors' semantics and exact positions in the world is crucial for the autonomy system's planning. However, HD maps are quite expensive, and great efforts have to be made to keep them up to date as many changes constantly occur in the world. A

single stop sign or traffic light that is missed in the HD map can cause serious issues to the perception module, which can endanger the safety of autonomous driving. Also, the dependence of self-driving vehicles on HD maps limits their ability to react to new situations and adapt to rapidly changing environments such as construction zones. Thus, towards an improved safety of self-driving systems, onboard perception systems could also be used to perceive static objects. This allows the system to combine the benefits provided by both perception and mapping for traffic-control features – timeliness of real-time perception, human-verified accuracy of the map. In this dissertation, we will study the perception of spatially compact static objects, such as traffic lights and signs. The term "spatially compact" is used to distinguish such objects from things like lane lines or road edge boundaries.

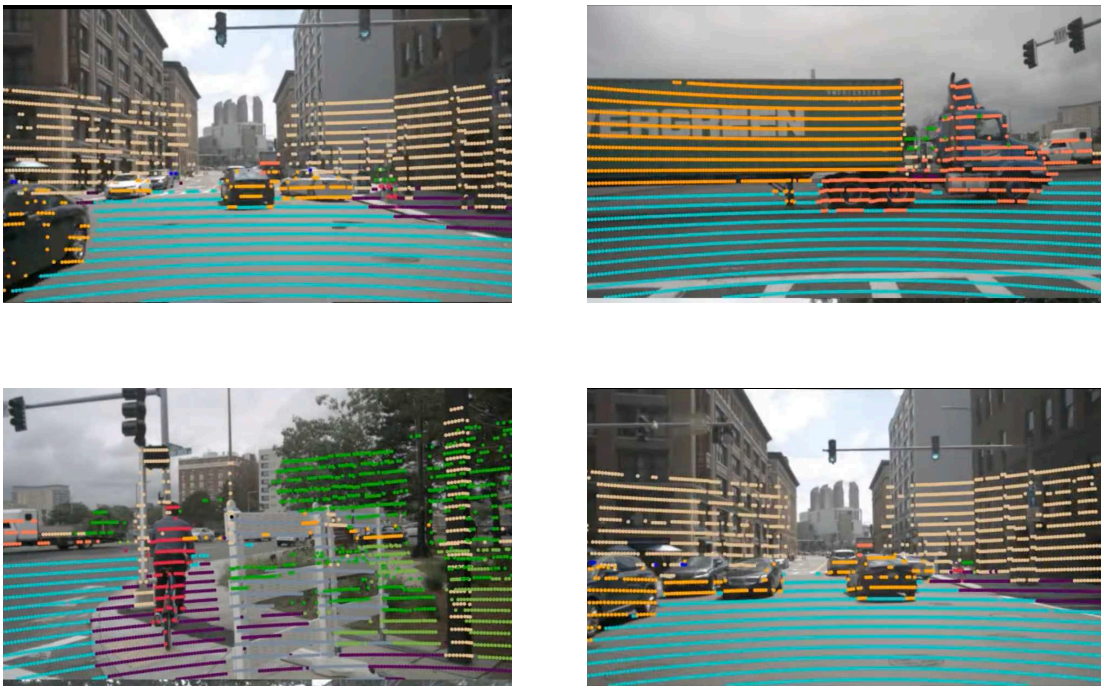

Figure 1.2: Examples of LiDAR point clouds data projected onto camera coordinate frame, from nuScenes dataset [3]. Note that for some systems LiDAR point clouds do not cover areas or objects that are at a certain height from the vehicle such as traffic lights. The images are taken from the nuScenes website (https://www.nuscenes.org).

Perception depends heavily on the number and type of sensors used by the vehicle. These sensors generally consist of cameras, Radar, LiDAR and GPS. Recently, there has been much

perception research based on the use of either LiDAR [12], or cameras [6, 13] or combining both [14, 15]. In this dissertation, we focus only on cameras (monocular imagery) as LiDAR is substantially expensive while cameras are cheap, smaller, more versatile, and have a higher resolution than any other used sensors. Also, this dissertation focuses on geo-locating static objects such as traffic lights, and for some systems, LiDAR point clouds do not cover them (see Figure 1.2) as they are generally at a certain height from the vehicle which is not covered by the LiDAR point clouds.

1.3 Contributions

This dissertation aims to tackle the perceptual challenges of self-driving vehicles. In summary, the contributions of this dissertation are as follows:

- For the problem of developing an accurate objects geo-localization system suitable for online autonomous driving, an end-to-end approach is proposed; It is mainly composed of three networks: 1) a pose regression network for estimating 5D poses of static objects from geo-located RGB inputs, 2) objects matching network for matching objects between pairs of video frames combining multi-resolution appearance features and geometric features from the proposed pose regression network, and 3) multi-object tracking network to produce the final geo-localization of the static objects within the scene. For each component, we compare against contemporary alternatives and show significantly improved performance.

- The experimental results show that jointly optimizing the pose regression and object matching models with a multi-task loss function improves the performance of the individual components as well as the system-level performance via joint learning. Also, we show that adding geometric cues explicitly for matching static objects helps in increasing the matching accuracy over appearance alone.

- Static objects geo-localization data set is created, which can help accelerate static objects geo-localization domain and provide a common benchmark for evaluating different

approaches. To the best of our knowledge, it is the first publicly available data set for this application.

- For the problem of multi-object tracking, we present a novel end-to-end approach, called DEFT, in which embeddings for each object are extracted from the multi-scale backbone of a detector network and used as appearance features in an object-to-track association sub-network. DEFT is simple, online, fast, and has competitive performance to top scoring methods in popular 2D tracking benchmarks. Because DEFT keeps a memory of appearance embeddings over time, it is more robust to occlusions and large inter-frame displacements than the top alternatives. This robustness allows DEFT to dramatically outperform competing methods on the challenging nuScenes 3D monocular vision tracking benchmark. DEFT is extensible to new detectors as they arise and adds only modest latency to the underlying detection network. DEFT runs at approximately 12.5Hz on all data sets.

- We generate difficulty scores that help describe how challenging each video is with respect to a specific factor. Occlusion, displacement, and crowdedness scores are created for videos in the nuScenes validation dataset (they can be applied to any other dataset). Based on these factors, the data set is then divided into partitions (easy/medium/hard). Analyzing the performance on these partitions helps obtain a more informative evaluation to validate the advantages of the proposed modules over state-of-the-art methods. We aim that these partitions will help researchers working in the multi-object tracking and other domains evaluate and compare their approaches' strengths and weaknesses.

- We propose an end-to-end model to solve the tasks of detection, tracking, and sequence modeling from raw sensor data. The proposed model, called Attention-based DEFT, extends the original DEFT by adding an attentional encoder module which uses attention to compute tracklet embedding that 1) jointly reasons about the tracklet dependencies and interaction with other objects present in the scene and 2) captures the context and temporal information of the tracklet's past observations. We conduct exhaustive experiments and

ablations on two popular benchmarks (KITTI and nuScenes) to validate the effectiveness of Attention-based DEFT. In terms of the tracking performance, the results suggest that the proposed approach performs favorably against or comparable to state-of-the-art methods. We show that reasoning about the interactions between the actors in the scene improves the model tracking performance significantly in highly crowded scenes while worsening its performance in hard scenes with respect to occlusions. We also show the superiority of Attention-based DEFT for sequence modeling tasks (velocity estimation) over other baseline methods on both simple and complex scenes. The experimental analysis validates the effectiveness of Attention-based DEFT for capturing spatio-temporal interaction of the crowd for velocity estimation task, which helps it to be more robust to handle complexities in heavily crowded scenes. Attention-based DEFT runs at approximately 12Hz on all datasets.

- The experimental results demonstrate that all the joint models in this dissertation perform better than solving each problem independently.

1.4 Dissertation Structure

The organization of the rest of this dissertation is as follows: Chapter 2 provides the necessary background gathered during the prestudy of this dissertation. We start with object detection, then go through static objects geo-localization, then multi-object tracking, and finish the chapter with joint detection and tracking. Chapter 3 presents an end-to-end learning approach for 5D pose estimation, tracking, and localization of spatially-compact static objects. Chapter 4 introduces an effective and efficient joint detection and tracking model that relies on an appearance-based object matching network jointly learned with an underlying object detection network. Chapter 5 describes an end-to-end model to solve the tasks of detection, tracking, and sequence modeling from raw sensor data, followed by Chapter 6 with our conclusions and future research directions.

Chapter 2

Related Work

This chapter aims to explore the perception component of the autonomy system. First, we will discuss different approaches used to solve object detection task (§ 2.1), which is a prerequisite for the perception of static and dynamic actors. Then, we will provide the necessary background to explore the various strategies of perception for static actors including geo-localization of spatially-compact static objects (§ 2.2). Finally, we will explore the perception module for dynamic actors, which aims to identify and keep track of moving objects in the scene (§ 2.3 and § 2.4).

2.1 Object Detection

Object detection is an important computer vision task that deals with identifying an object in an image along with its localization and classification. Object detection has been studied extensively in the past decades. The traditional approach is to use hand-crafted features for object detection. Hand-crafted features usually require advanced research and years of domain-specific expertise to design complex feature representations. With the appearance of Deep Learning (DL), these features rapidly became outdated as modern deep neural networks began naturally learning robust and high-level feature representations of an image. The employment of DL has revolutionized the object detection field. In the following, we will briefly review traditional object detectors in § 2.1.1, then discuss the more recent DL based object detectors in § 2.1.2. Finally, in § 2.1.3, we will cover object detectors used for object pose estimation.

2.1.1 Traditional Object Detectors

The initial methods for object detection were based on alignment techniques to find the correspondence between the model of the object and the image using straightforward features, such as key-points [16], edges [17], or templates [18].

Before 1990, object detection methods were based on geometric representations [19], and then the focus has shifted towards the utilization of statistical classifiers such as SVM [20], neural networks [21] and Adaboost [22], based essentially on appearance features. The appearance features moved from global representations [23] to local representations, starting with the Scale Invariant Feature Transform (SIFT) feature [16]. In 2001, Viola and Jones [22] presented a Machine Learning (ML) based object detector, called Viola-Jones (VJ) detector, which is based on the Haar-like features along with the cascaded AdaBoost classifier. The VJ detector was introduced for real-time face detection, and it is a major source of inspiration for many subsequent ML-based object detectors. The VJ detector and early ML-based object detectors perform an exhaustive search with a sliding window to find potential objects in all feasible regions and scales.

Histogram of Oriented Gradients (HOG) detectors [24] were then proposed and were able to achieve relatively high accuracy but low speed. To detect objects of variable sizes, the HOG detector rescales the input image multiple times while using a single detection window. Later, Felzenszwalb *et al.* [25] extended the HOG detector and proposed a Deformable Part based Model (DPM), which represents objects by component parts arranged in a deformable configuration. The DPM adapts the detection logic of "divide and conquer", where the training can be inferred as the learning of an appropriate manner of decomposing the object in different parts and the testing as an ensemble of detections on different components of the objects.

Many traditional object detectors usually employ an aggregation of the local features (e.g., SIFT, Haar-like features, HOG) by simple concatenation or feature pooling encoders such as the Bag of Words (BoW) approach [26] and Spatial Pyramid Matching (SPM) of BoW models [27].

2.1.2 Deep Learning Based Object Detectors

After 2010, object detection began to plateau as the performance of hand-crafted features became saturated. However, in 2012, the revival of Convolution Neural Networks (CNNs) inspired researchers in the object detection domain. The CNN-based model AlexNet, proposed by Krizhevsjy *et al.* [28], won the image classification competition of the image dataset

ImageNet [29] with a considerable gap of 11% accuracy over the next best approach using traditional algorithms.

In 2014, Girshick *et al.* [30] proposed Region-based Convolutional Neural Networks (R–CNN). First, selective search-based image scanning is used to generate region proposals as bounding boxes. CNN is then employed to extract a fixed-size feature vector from each region, and finally, class-specific linear SVMs are employed to classify them. On the VOC07 dataset [31], R-CNN achieved significant improvements of mean Average Precision (mAP) from 33.7% to 58.5% over the winner of the challenge DPM. R-CNN runs at approximately 14 seconds per image with GPU.

The same authors improved the detection quality by proposing Fast R-CNN [32], which shares feature maps across object proposals. In Fast R-CNN, the whole input image is fed to CNN to generate feature maps. Then, a fixed-length feature vector is extracted from each region proposal with a region of interest (RoI) pooling layer. Each RoI feature vector is then fed into fully connected layers to predict the class of the proposed region and also refined bounding box positions. Fast R-CNN and R-CNN rely on selective search to generate the region proposals, which is a bottleneck in improving efficiency. Therefore, Ren *et al.* [33] proposed Faster R-CNN [33] which employs Region Proposal Network (RPN) to determine proposals instead of the traditional selective search algorithm. RPN is inspired by the work of Sermanet *et al.* [34] which demonstrated that CNNs are inherently efficient at computing a sliding window as by nature they share computations common to overlapping regions. RPN is a fully convolutional network, which can predict object bounds and scores at each position simultaneously. This allowed joint end-to-end training of the model for both detection and classification tasks. On the VOC07 dataset, Fast R-CNN and Faster R-CNN achieved 70.0% and 73.2% mAP, respectively. Faster R-CNN runs at approximately 0.2 seconds per image.

Lin *et al.* [35] proposed Feature Pyramid Network (FPN) based on Faster R-CNN. They exploited the inherent multi-scale, pyramidal hierarchy of CNNs to create feature pyramids with negligible extra cost. A top-down convolutional network structure with lateral connections was employed in FPN to extract multi-scale semantic features. The FPN model demonstrated strong

improvements in detecting objects with a variety of scales. It achieved state-of-the-art performance on the COCO dataset [36] by boosting the detection mAP from 42.7% (Faster R-CNN) to 61.1%.

All previously mentioned methods belong to two-stage approaches, consisting of two steps: 1) region proposal generation and 2) bounding box classification and regression. Two-stage detectors usually suffer from high computational time ($\leq$ 5 FPS). In contrast, Redmon *et al.* [37,38] proposed a one-stage detector that skips the proposal generation step and predicts class probabilities and bounding boxes directly from the input image using a simple CNN, called "You Only Look Once" (YOLO). YOLO divides the input image into a predefined number of grids. Each grid cell predicts probabilities (confidence scores) for a fixed number of bounding boxes (also called anchors). A post-processing step, called Non-Maximal Suppression (NMS), is then applied to high confidence bounding boxes to eliminate duplicate detections and produce the final detections. YOLO has shown to be much faster than two-stage approaches but less accurate. On the VOC07 dataset, a fast version of YOLO runs at nearly 155 FPS achieving 52.7% mAP, while its enhanced version runs at 45 FPS with 63.4% mAP.

Subsequent work by Liu *et al.* [39] on the Single Shot multi-box Detector (SSD) provided another one-stage detector similar to YOLO. SSD is a simple network that achieves real-time performance. SSD is based on a CNN that maps an image into multiple feature maps in order to generate a fixed number of boxes with different aspect ratios and scales. The authors demonstrated that a larger number of carefully selected default bounding boxes leads to better performance. SSD predicts class scores and box offsets for the predefined boxes (anchors), followed by an NMS to locate the object within the image. A fast version of SSD runs at 59 FPS with VOC07 mAP = 76.8% , and COCO mAP = 46.5%.

Despite their simplicity and high speed, the one-stage detectors have long struggled for inferior accuracy compared to two-stage detectors. Lin *et al.* [40] discovered that the main reason for the difference in accuracy between the two categories of detectors is the high imbalance between foreground and background classes during training. To tackle this issue, they proposed a new

loss called “focal loss” by modifying the classical cross-entropy loss so that the detector will focus more on hard, misclassified examples during training. Focal loss facilitated closing the gap in terms of accuracy between the two categories of detectors. The authors also presented the RetinaNet network, similar to the FPN architecture but in a one-stage setting. On the COCO dataset, RetinaNet achieved 61.1% mAP with a running time of 11 FPS.

All previously mentioned one-stage detectors use anchor boxes to serve as detection candidates, which are typically a very large set (e.g., more than 100k in RetinaNet), and only a small fraction of them will overlap with the ground truth. This often leads to foreground-background class imbalance during training, as Lin *et al.* [40] showed in their experiments. Anchor-based detectors also suffer from a large number of hyper-parameters and design choices. To solve these issues, Zhou *et al.* [13] proposed an anchor-free object detector, called CenterNet, which models the object by the center point of its bounding box. CenterNet uses keypoint estimation to determine center points and regresses to all other object properties, such as size and even 3D location, orientation. The input image is fed through a CNN to produce a heatmap, in which peaks correspond to center points. Image features at each peak are passed through a neural network to regress other object properties. CenterNet is end-to-end differentiable and performs competitively with state-of-the-art detectors with a higher inference speed. CenterNet achieved the best accuracy-speed trade-off on the COCO dataset, with 44.9% mAP at 142FPS, 55.1% mAP at 52 FPS, and 63.5% mAP at 1.4 FPS, depending on the employed backbone network.

2.1.3 Object Pose Estimation

Object pose estimation extends the task of object detection and also estimates the location and orientation of the detected objects relative to some coordinate system. Most state-of-the-art methods for object pose estimation [41–43] use 3D models of the objects. They generally consist of two stages: the first stage detects local 2D key-points of the object, and the second stage utilizes the 2D-3D correspondences between the predicted key-points and their matches in the 3D model of

the object and estimates the object pose using a classical algorithm such as the PnP algorithm [44]. These methods do not work well for autonomous driving application because of the presence of multiple types and sizes for each class of static objects (e.g., traffic lights, signs) in real-world scenarios, and in such cases, the pose estimation model cannot rely on the 3D model of the object.

Some end-to-end methods for object pose estimation that can work without the constraints of having the object's 3D model have been proposed. Kendall *et al.* [45] proposed PoseNet, a fully convolution neural network to directly regress the camera pose from a single RGB image. PoseNet is composed of a modified truncated GoogleNet [46] and can operate in real-time. However, estimating the 3D translation directly from the image features is challenging and not generalizable as objects can be located in any position in the image. To tackle this problem, Xiang *et al.* [47] decided to estimate the 3D translation by localizing the object center in the 2D RGB input and estimating the object distance from the camera. Their proposed approach, called PoseCNN, is composed of two blocks: the first block contains a CNN backbone to extract features from the input image, and the second block embeds the high-dimensional feature maps extracted from the first block into low-dimensional task-specific features. The second block consists of three network branches to estimate the object pose: semantic segmentation, 3D translation estimation, and 3D rotation regression. One of the findings of Xiang *et al.* [47] is that decoupling translation and rotation estimation into different branches has been shown to give better results and enabled PoseCNN to model the dependencies between them. However, end-to-end methods have shown weaker generalization and poor performance compared to two-stage approaches, and thus they are still not popular for pose estimation.

Given that there is room for improving the pose estimation accuracy for end-to-end methods and also the constraints of not using the 3D model data for our application, in Chapter 3, we propose an end-to-end pose regression network for estimating the poses of static objects from RGB inputs, which has shown to outperform PoseNet [45] and PoseCNN [47].

2.2 Static Object Geo-localization

In the past few years, tremendous efforts have been made in localizing street-level objects using multi-view geometry. Image triangulation has been established as a common technique for this task. Triangulation is the process of geo-locating a point in 3D space given its position in two or more images taken from cameras with known calibrations and poses. Timofte *et al.* [48] proposed an offline traffic sign mapping pipeline based on the epipolar geometry of multiple images. Traffic signs candidates are first extracted from the single-view image using hand-crafted features, and then a set of multi-view 3D hypotheses are generated using simple geometric and visual consistency between each pair of candidates. Finally, Minimum Description Length (MDL) optimization is employed to determine the optimal set of 3D hypotheses that best explains the overall set of 2D candidates.

In a similar attempt, Soheilian *et al.* [49] presented an offline system for detection and localization of traffic signs from calibrated multi-view images. First, color-based segmentation is applied to generate ROIs validated based on the shape and passed to a template matching algorithm to match them with a set of reference road signs. A stereo matching algorithm taking into consideration epipolar geometry is then applied to generate 3D hypotheses. The hypotheses that likely belong to the same 3D object are determined and clustered during this process. Finally, the hypotheses of the same cluster are combined to generate a unique 3D object by a multi-view algorithm incorporating apriori knowledge of the road signs' 3D shapes as constraints.

With the resurgence of DL, many studies employed DL for object detection and classification. Hebbalaguppe *et al.* [50] proposed a system to automatically update telecom inventory using object detection and triangulation-based method applied to Google Street-View (GSV) images. First, objects of interest are detected using Faster R-CNN [33] detector. Then, SIFT feature matching algorithm is used to find the point correspondence between pairs of images from two consecutive locations on the street. Triangulation is then applied to estimate the location of the object. A major drawback of this approach is that it works only with a pair of images and does not associate objects between more than two video frames.

All previously mentioned methods rely on triangulation from multiple camera views for objects geo-localization. They rely on simple visual and geometrical clues to perform matching when multiple objects are present. Therefore, when multiple identical objects are present together, these methods perform poorly. Hebbalaguppe *et al.* [50] for example, pointed out that using SIFT for visual matching has limited performance for reliable visual matching when there is a substantial view-point position change or with the presence of multiple, similar-looking objects.

This motivated many studies to propose novel triangulation approaches. For instance, Zhang *et al.* [51] proposed a system that consists of a RetinaNet object detector [40] to detect poles from street view images followed by a novel brute-force Line-Of-Bearing (LOB) method for object-localization. Finally, a geo-spatial aggregation algorithm is used to estimate the centroid of clusters of LOB intersection locations. However, the evaluation of this approach showed poor performance when objects are very close to each other. The experimental results showed that approximately 2.6% and 47% of the predicted locations are within 1 meter and 5 meters, respectively, from the ground truth locations.

Krylov *et al.* [5] presented a better triangulation approach, where they proposed a novel model by using a Markov Random Field (MRF) for triangulation and geo-localization of recurring static objects from GSV imagery. They used two CNNs: the first to perform semantic segmentation for objects detection and the second to estimate their distance from the camera. To geo-locate all the detected objects, they proposed an MRF model to integrate depth information into geometric triangulation from object instances discovered in multiple street view images. Finally, clustering is applied to merge geo-located objects in the same area that are likely to describe the same object. They tested their approach on traffic lights and telegraph poles, and they achieved high object recall rates and localization accuracy within 2 meters.

Some studies combined aerial imagery with street-level imagery for better objects geo-localization. Branson *et al.* [52] presented a system for tree detection and geo-localization using street-level panoramas combined with aerial imagery as input. First, they employed Faster R-CNN [33] to detect trees in each image/view, and then they applied a projection function to

project each detection from image space into a common geographic coordinate system. The union of all detections from different views are then back-projected into image space for each view, such that detection scores can be computed with known alignment between each view. Then, a Conditional Random Field (CRF) generates combined detections using multi-view scores and with the help of semantic map data (e.g., roads locations) and spatial context of neighboring objects. Their approach can detect around 70% of the street trees (within a 4 meters radius from the ground truth tree). However, this approach was built based on assumptions that are not always valid in a realistic environment. The authors assumed that the terrain is locally flat and that the camera is leveled against the ground (pitch = 0), and even when such assumptions are valid, they showed different kinds of causes that can yield to system failure.

All earlier mentioned approaches use multi-step and cascaded systems where they treat pose and image evidence independently. Most recent approaches use DL object detectors followed by a separate model to track or otherwise link observations across images without the complete support of information from the detector. Therefore, when multiple objects are very close to each other, geometric-only systems can fail due to the inherent spatial uncertainty of the features.

To tackle this issue, Nassar *et al.* [6] recently proposed an end-to-end trainable object geo-localization architecture that jointly learns object detection and instance re-identification in different views with learned geometric soft-constraints. Their architecture takes a pair of GSV images along with their geographic metadata as input and passes both through two blocks that share the same weights. Objects are first detected in the image pairs using SSD detector, then bounding boxes from each image are projected into the other image's space. For each block, the predicted boxes and projected boxes locations are passed to a learnable Multi-Layer Perceptron (MLP) to regress their real-world geo-coordinates. The projected predictions are also passed to another network for fine-tuning of the projections. The full model is end-to-end trainable with a multi-task loss function. Their approach showed very promising results: it achieved an error of 3.13 meters for geo-locating trees on the Pasadena dataset and 4.36 meters for geo-locating signs on the Mapillary dataset [53]. However, it only works with one pair of frames, making it

unsuitable for online autonomous driving systems where a sequence of more than two frames is used to geo-locate static objects in the scene. For the objects geo-localization model to be suitable for online autonomous driving systems, it should ideally detect static objects in the current frame, associate them with the detected objects in the previous frames (more than one prior frame), and geo-locate each tracked object.

To address the problem of having more than two video frames in the input, the same authors [54] presented an end-to-end method for object detection and geo-localization based on GNN. Their approach takes multiple images and their corresponding approximate camera poses as input. Their architecture comprises three components: an object detector, a GNN for objects association, and a geo-localization regressor where the full model is end-to-end trainable. First, objects are detected using an anchor-based object detector. Then, an undirected fully connected graph is constructed, which contains nodes composed of the features extracted from the detector's backbone concatenated with pose information and the predicted bounding box values. Then, the graph is passed through a Graph Neural Network (GNN) trained to classify a pair of objects into a matched or unmatched pair. In parallel, the detected bounding boxes are passed to a neural network to regress their geo-coordinates. This approach achieved better performance with less latency compared to their previous approach [6]. It reached an average geo-localization error of only 2.75 meters on the Pasadena dataset and 3.88 meters on the Mapillary dataset. Even though this method accepts any number of frames as input, it is not suitable for the autonomous driving application because, in order to geo-locate objects in a sequence of frames, it needs to have all frames as input to the model. For an online perception of static objects, their approach has to reprocess all frames in the memory with each new frame, which is inefficient and time-consuming.

The two approaches proposed by Nassar *et al.* [6, 54] have solved the issues of the multi-step or geometric-only systems and have shown that the end-to-end approach for object detection and geo-localization performs better. However, as we mentioned before, both are not suitable for autonomous driving application. In contrast, in Chapter 3, we propose an end-to-end method

for detection, tracking, and localization of spatially-compact static objects that is suitable for the online autonomous driving application.

Static objects geo-localization using multiple street-view pictures has been the focus of important prior work. Unfortunately, there is no common benchmark where different approaches can have a fair comparison, as they all use their private datasets. Thus, in this dissertation (Chapter 3), a public Traffic Lights Geo-localization (TLG) data set is created from nuScenes dataset [3], which can help to accelerate static objects geo-localization domain and provide a common benchmark for the evaluation of different approaches. In this dissertation, we compare our proposed approach with existing methods that have publicly available code.

2.3 Multi-Object Tracking

Multi-Object Tracking (MOT) is a challenging problem in computer vision that aims to localize all objects of interest (e.g., pedestrians, vehicles) and reliably recover their trajectories in a video sequence while maintaining their identities. MOT aims to output objects bounding boxes present in each frame (detection), identify where they are in each frame (localization), and assign a unique identity to each bounding box in order to determine whether objects in different frames belong to the same or different objects (association).

Multi-object tracking is generally evaluated using CLEAR-MOT metrics [1] which contains multiple performance measures, with Multi-Object Tracking Accuracy (MOTA) and Multi-Object Tracking Precision (MOTP) as the most used ones. MOTA is a summary of the overall tracking accuracy. The counts of False Negatives (FN), False Positives (FP), and Identity Switches (IDs) are summed and divided by the total number of ground-truth objects across all frames to compute a total error rate, which is then subtracted from 1 to compute the MOTA. IDs score is the number of times a trajectory gets a new predicted identity even though the ground truth identity remains the same. MOTP is the total position error for the associated hypotheses over all frames averaged by the total number of matches made. Mostly Tracked (MT) and Mostly Lost (ML) are also two widely used metrics proposed by Li *et al.* [55]. MT is given by the proportion of ground truth

trajectories tracked for more than 80% of their lifetime. ML is given by the proportion of ground truth trajectories tracked for less than 20% of their lifetime.

In CLEAR-MOT metrics, the confidence value of the detection is not taken into account, and all metrics are computed based on a single confidence threshold. Thus, Weng and Kitani [56] proposed the Average MOTA (AMOTA) and the Average MOTP (AMOTP), which are the averages of the MOTP and the MOTA across many thresholds. The AMOTA and AMOTP are calculated by integrating MOTA and MOTP values over all recall values, e.g., area under the MOTP over recall curve for computing AMOTP. They also proposed a new MOTA called recall-normalised MOTA (MOTAR) which uses a recall-normalization term to prevent the MOTA from being negative (the MOTA score can be negative when the number of errors is higher than the number of targets in the video). Recently, Jonathon *et al.* [57] showed that all previously mentioned metrics overemphasize the importance of either detection or association, and that is why they proposed the Higher-Order Tracking Accuracy (HOTA) metric, which explicitly balances the performance of the detection, association, and localization. The detailed definition of the HOTA score can be found in [57].

The standard approach employed in MOT algorithms is "tracking-by-detection" which is characterized by two main steps: 1) a detection step to localize objects in a single video frame, and 2) an association step to link objects detected in the current frame with those from previous frames.

Recently the performance of object detection methods has improved substantially (see § 2.1), and the majority of MOT methods have been focusing only on improving the association step. Indeed, many MOT benchmarks [58,59] force different methods to use their set of public detections to have a fair comparison at solely the association step, as the quality of detection can significantly affect the MOT results. Thus, many MOT algorithms ignore the detection step and formulate the tracking problem as a data association problem, i.e., detection hypotheses that are considered to belong to the same object are associated into object trajectory (known as "tracklet").

Although there are a large number and variety of methods solving the MOT problem, the data association step roughly contains the following three stages: i) Feature extraction stage: feature

extraction algorithms process each detection and/or tracklet into representative features. ii) Affinity estimation stage: statistical measuring of the similarity between pairs of detections and/or tracklets when representative features are extracted. iii) Association stage: the estimated affinity scores are used to link new detections and existing tracklets by assigning the same label to detections that belong to the same physical object.

The data association step is commonly formulated as a graph problem. Current detections and existing tracklets are represented as nodes of the graph, and edges between each pair of nodes are weighted with the similarity scores between them, computed using the representative extracted features. The association stage is then modeled as an optimization problem with respect to the weighted graph.

In prior works, researchers used simple distances between available appearance/motion features and focused on solving the various graph optimization problems [60, 61] that have been formulated to determine optimal trajectories. In recent years and especially with the resurgence of DL, the researchers' interest has shifted from developing novel optimization algorithms to using simple matching algorithms (e.g., Hungarian matching [4, 62–64], greedy matching [7, 65]) and focusing on learning features that are better for data association. This drift has been reflected with a significant increase in performance in the MOT frameworks. For the affinity estimation stage, while it is generally computed as the distance (e.g., Euclidean distance [65], Mahalanobis distance [66]) between extracted features, some recent MOT frameworks [67–69] used DL models to directly estimate similarity score, without specifying any explicit distance measure between the extracted features.

In the following, we will review the main works for MOT frameworks. This section is organized based on the used features for the association. First, the use of hand-crafted features for the association is briefly discussed in §2.3.1. §2.3.2 will cover the employment of classical CNNs to extract appearance features. Then, a variant of CNNs, known as Siamese CNNs, is examined in §2.3.3. §2.3.4 explores motion features that have also been widely used in MOT. Finally, the use of interaction features which is a recent trend in MOT frameworks is explored in §2.3.5.

2.3.1 Hand-crafted Features

The goal of feature extraction is to extract discriminative features that help to compute affinity/distance scores between different detections and/or tracklets. Traditional MOT methods mostly focused on feature selection. Among different kinds of features, appearance and motion have shown to be the best discriminative features. Hand-crafted features such as Intersection-Over-Union (IoU) [70] and spatial distance [60] were used as the motion features. For the appearance features, pre-defined color intensity [71] and histogram [72] were the most popular features. However, they do not deal with changes in imaging conditions, e.g., illumination changes. To overcome this, other hand-crafted features such as optical flow and HOG [24] have been employed. The optical flow feature was used to associate detections into short tracklets for more data association processing [73] or directly use it for data association [74]. HOG feature descriptor played a crucial role in the multi-pedestrian tracking problem [74]. Multiple works fused multiple handcrafted features [75, 76] for better discrimination, but they were still not robust enough.

2.3.2 CNN-based Appearance Features

With the recent success of DL in different computer vision tasks, recent MOT algorithms have used DL in various ways to improve tracking performance. More specifically, feature extraction using CNNs has been the most successful thanks to their powerful capability of extracting abstract, meaningful, and complex features.

In 2015, Chanho *et al.* [77] was one of the pioneers to utilize CNN-based appearance features along with the classical Multiple Hypotheses Tracking (MHT) algorithm. First, image patches of the detections are passed through a pre-trained CNN to extract appearance features, followed by Principal Component Analysis (PCA) to reduce the dimensionality. Then, the obtained features are fed to a classifier to compute a log-likelihood cost for a track hypothesis given a set of detections. This work demonstrated that a classical MHT implementation from the 90′s surprisingly performed well in comparison with state-of-the-art methods on standard benchmarks. When their paper was

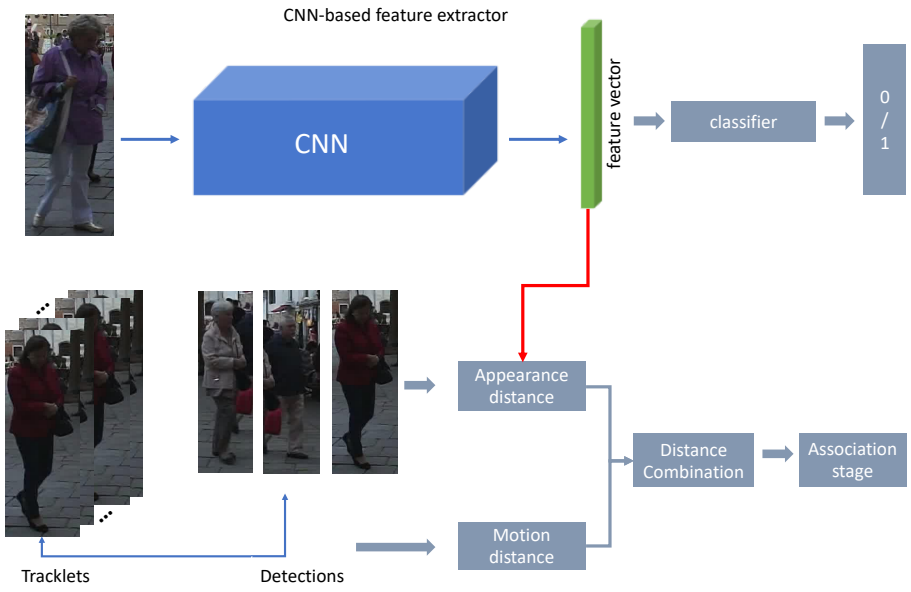

Figure 2.1: Framework of deep SORT [4] CNN-based feature extractor. Deep appearance features extracted from CNN, pre-trained for classification, are used to improve the association between new detections and existing tracklets.

first published, their model outperformed the second-best tracker on the 2DMOT15 [58] challenge by 7% in terms of Multi-Object Tracking Accuracy (MOTA) [1].

Later, Yu *et al.* [62] employed a network similar to GoogLeNet [46], pre-trained on multiple person re-identification datasets, to extract the appearance features from image patches of different detections and utilized the cosine distance between different detection features as a measure of the appearance affinity. Appearance affinities are then combined with straightforward motion and shape affinities to construct the affinity matrix, and finally, the Hungarian algorithm [78] is adopted. The Hungarian algorithm is a combinatorial optimization algorithm that solves the assignment problem based on the computed affinities scores. In their experiments, Yu *et al.*demonstrated that high-performance detection (Faster R-CNN [33] trained on several extra detection datasets) combined with simple CNN-based appearance features for association can lead to state-of-the-art performance in MOT.

Inspired by the work of Yu *et al.* [62], CNN-based appearance features were extensively used in simple MOT algorithms and combined with state-of-the-art detectors and were able to achieve state-of-the-art performance. One of the most widely used frameworks in this category is Deep

SORT [4], which is an extension of Simple Online and Realtime Tracker (SORT) [63] (details about SORT can be found in §2.3.4). Deep SORT combined CNN-based appearance features with Kalman-filter-based tracking and Hungarian algorithm for data association and achieved the best performance of (MOTA score of 61.4) on MOT16 challenge [59] in 2017. Deep SORT employs a wide residual network [79] to extract appearance features of objects and then computes their similarities with cosine distance. The network is pre-trained on a large-scale person re-identification dataset to extract a normalized vector for each box, capturing the appearance features of each. A combination of the cosine distance and motion-based Mahalanobis distance is used as affinity estimation. A diagram of the framework is illustrated in Figure 2.1. This adaptation successfully reduced the number of identity switches by 45% leading to a more stable tracking. Their work demonstrated that incorporating appearance features from pre-trained CNN into the MOT algorithm improves the tracking performance.

2.3.3 Siamese Networks

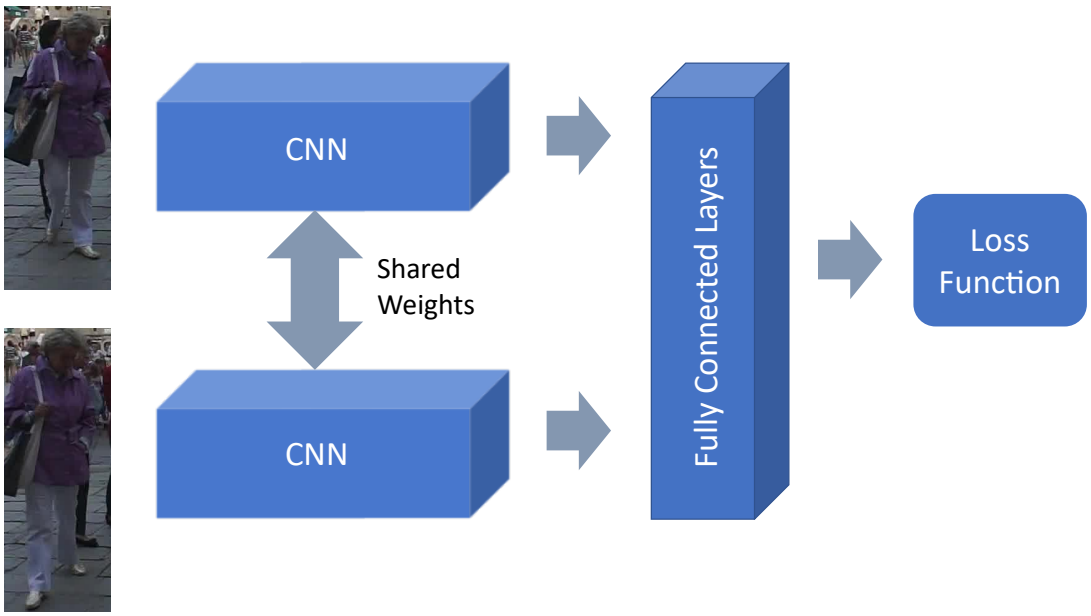

Figure 2.2: The basic architecture of a Siamese network. The weights of the convolutional layers are shared between the two inputs of the network. For feature extraction, the network is trained as a Siamese CNN. For the inference, the last fully connected layer is utilized as the representative feature vector for a given object. Some methods use Siamese architecture during the inference for estimating affinity between pairs of objects as well.

A variation of the classical CNN architecture that has seen enormous use in MOT and is considered the most popular DL architecture in the MOT problem is the Siamese network (Figure 2.2). A Siamese network is basically a CNN network trained with a loss function that combines information from multiple images with shared weights to learn features that best discriminate between objects. Intuitively, the Siamese network is suitable for the task of learning the association likelihood between two inputs.

Contrastive loss function [80] is the traditional loss function that was used to train a Siamese network. This loss function trains the network to minimize the distance between each pair of images that belongs to the same object (lower than a pre-selected margin) while maximizing the distance between each pair that belong to two different objects (larger than the same margin). Leal-Taixé *et al.* [81] proposed a CNN-based Siamese network that directly predicts the likelihood that the two inputs belong to the same object. More specifically, the two input patches are stacked along with the associated optical flow information. The obtained input is then fed to the Siamese network which is trained using the contrastive loss function. Furthermore, contextual features are extracted from the variations of position and size between the two input patches and then combined with the final layer of the Siamese network. The obtained combined representation is then fed to a gradient boosting classifier to produce the final similarity score.

Lee *et al.* [69] proposed Feature Pyramid Siamese Network (FPSN) inspired by the work of Leal-Taixé *et al.* [81], in which they extended the CNN Siamese network by incorporating FPN [35]. In FPSN, the feature maps at multiple levels of the backbone convolutional networks are extracted to use both coarse- and fine-level information. To incorporate motion information into FPSN, a spatio-temporal motion difference feature is added for better affinity estimation. To train FPSN, the contrastive loss function was generalized to capture the multi-level features in FPSN.

Various pairwise loss functions have also been proposed to improve the discriminability of the Siamese network. Bae *et al.* [82] for example, used appearance features extracted from their proposed Siamese CNN network, and for training, they proposed an energy function with high-level feature distances of sample pairs inspired from the work of Alexis *et al.* [83]. In their

experiments, the authors demonstrated the effectiveness of the proposed loss as compared to the contrastive loss. Bae *et al.*formulated the MOT problem based on tracklet confidence measured using its detectability and continuity through frames. For each given new frame, they associated the tracklets and new detections in different ways according to the confidence values of the tracklet. For both high- and low-confidence tracklets, similarity scores based mainly on the appearance features are computed and then passed to the Hungarian algorithm for the tracklet-detection association. The authors also demonstrated that combining online transfer learning, by adapting the pre-trained deep model during online tracking, improves appearance discriminability.

Thanks to the recent advances in the computational power of DL frameworks, many researchers took advantage of tensors formulation to produce a faster way of computing affinities between multiple pairs of detections in a single forward pass. Sun *et al.* [84] used a Siamese network that takes two video frames as input and outputs similarity scores between all pairs of detections. Appearance embeddings are first extracted for all detections in a pair of frames, and then all possible pairs of detections embeddings between the two frames are concatenated to form a tensor. Later, the obtained tensor is passed through convolution layers with a kernel size (1×1) to obtain the final similarity matrix containing all similarity scores between all pairs of detections. Finally, a binary classification loss function is used to train the network to classify each output in the similarity matrix to a matched or mismatched pair. This approach is similar to the classical Siamese network, extended by using tensors formulation, making its implementation faster.

Another widely used loss function for training Siamese networks is the triplet loss [85] where each training input consists of three samples instead of two. Two of the three samples belong to the same object, and the third belongs to a different object. Thus, the loss function trains the Siamese network to maximize the relative distance between the matched pair and the mismatched pair. Chen *et al.* [86] employed a network inspired by GoogLeNet [46] and trained the network with triplet loss on person re-identification task to extract the appearance features of objects. Their MOT algorithm first outputs possible detection candidates using both the output of R-FCN detector [87]

and the predictions of existing tracklets using Kalman filter. The authors explained that extracting redundant candidates is beneficial because high-confidence detections prevent tracking drift in the long term, and the predictions of tracklets help in handling occluded objects. Candidates for data association are then selected based on a unified confidence score and by applying NMS. Finally, existing tracklets are hierarchically associated with selected candidates using Euclidean distance between the appearance features. At the time of publishing, their method achieved state-of-the-art performance on the MOT16 benchmark with a MOTA score of 47.6.

Rather than considering a pair or a triplet of image patches with Siamese networks, Son *et al.* [88] proposed an extensive Siamese network, called Quadruplet Convolutional Neural Network (Quad-CNN), that uses quadruplets of image patches as inputs: three of them belong to the same object, but in ascending temporal order, and the fourth one belongs to a different object. Similar to other Siamese networks, Quad-CNN enforces a matched pair to have a lower distance than a mismatched pair. In addition, it encourages a temporally adjacent matched pair of detections to have a lower distance than a temporally distant pair. Quad-CNN learns the similarity between objects using a quadruplet loss which combines the appearance of detections with their sequence-specific motion-aware position for metric learning. A multi-task loss for learning object association and bounding-box regression (to refine noisy localization of detected objects) is used to jointly optimize Quad-CNN in an end-to-end fashion. Quad-CNN showed competitive results with state-of-the-art methods on the MOT16 benchmark, achieving MOTA score of 44.1.

Siamese networks are typically used to learn affinities between pairs of detections. Wang *et al.* [89] proposed a novel Siamese network, called TrackletNet Tracker (TNT), that learns similarities between pairs of full tracklets. Appearance features extracted from FaceNet [85] are concatenated with bounding box parameters to construct an embedding feature vector for each detection. To estimate affinity between two tracklets, embeddings from all detections of both tracklets within a fixed time window are fused and fed to TNT network. The MOT problem in this work is formulated as a tracklet graph model. First, tracklets are built from consecutive detections using IoU and CNN-based appearance similarity. Then, the TNT network is employed to estimate the similarity

between each pair of tracklets. Finally, a clustering algorithm is adopted to group tracklets that belong to the same object into the same cluster. TNT achieved promising results on MOT16 (49.2 MOTA) and MOT17 (51.9 MOTA) benchmarks compared with state-of-the-art methods.

2.3.4 Motion Features

In real-world scenarios, appearance-only models can be unreliable for discriminating between objects of the same class, especially with similar shapes and textures. In such cases, motion features can provide better representative information useful to discriminate between objects. Motion is an important component in tracking systems that can be combined with appearance models [4, 90] or even employed solely [56, 63].

Motion prediction features are the most common motion features in MOT. In early works, Kalman filter [91] was a very popular motion prediction model in MOT that is used to predict the current object's state based on its previous states. A typical technique in early methods [4, 56, 63, 90] is leveraging Kalman filters to model the object velocity and predicting its bounding box location and size in the current frame, followed by the Hungarian algorithm to complete the association. One popular Kalman filter based MOT algorithm is "Simple Online and Realtime Tracking" (SORT) [63] which has been one of the pioneers to leverage CNNs for object detection and combine it with an off-the-shelf Kalman filter and achieved the best performance, of 33.4 MOTA, among open-source algorithms on 2DMOT15 challenge in 2016. SORT first detects object bounding boxes and classes using Faster R-CNN [33] and uses the Kalman filter to predict the tracklets' location in the current frame. The similarity cost matrix is then computed as the IoU distance between each new detection and all predicted bounding boxes from existing tracklets. Kalman filter has also been employed for 3D tracking (tracking 3D bounding boxes) to assign a unique ID for each 3D detection. AB3DMOT, for example, [56], uses 3D detection from LiDAR point cloud and then combines 3D Kalman filter and the Hungarian algorithm for state estimation and data association.

Kalman filter makes a major assumption on the linearity of the motion model. However, in realistic tracking scenarios, the motion model can be more complex and especially in crowded scenes. To tackle the non-linearity problem, variants of Kalman filter have been proposed to solve MOT problems such as the Extended Kalman Filter (EKF) [92], and Unscented Kalman Filter (UKF) [93].

Recently, some works employed CNN correlation filters to predict existing tracklets locations in the current frame [64, 94]. For instance, Zhao *et al.* [64] incorporated features from multiple layers of the SSD network [39], compressed using PCA, into the correlation filter. The correlation filter computes a response map for each tracklet in the current frame, and the maximum value in the map is then considered as the estimated location of the tracklet. Finally, the similarity score is computed based on the IoU between new detections and predicted locations of the tracklets combined with the appearance similarity score extracted from the response map. Their approach was able to achieve a MOTA score of 32.7 and 71.27 on 2DMOT15 and KITTI [11], respectively.

The recent success of Recurrent Neural Network (RNN) in modeling time-series data motivated researchers to employ RNNs and their variations to solve dynamic and complex motion prediction problems. Milan *et al.* [67] proposed an RNN-based approach to learn complex motion models inspired by the Bayesian filtering idea, where the temporal dynamics of objects learned by RNN are exploited to accomplish state prediction and updating as well as track management. Their RNN network jointly performs motion prediction, data association, state update, and estimation of the initiation and termination of objects within a unified network. However, the entire approach mainly focuses on motion prediction and ignores appearance features, and this was reflected by relatively poor tracking performance (19.0 MOTA on 2DMOT15 benchmark). To mitigate this issue, Sadeghian *et al.* [68] presented a MOT framework based on a structure of RNNs that jointly encode temporal dependencies across multiple features including appearance, motion, and interaction features, over a temporal window. Their framework achieved a high tracking performance with a MOTA score of 37.6 and 47.2 on 2DMOT15 and MOT16

benchmarks, respectively. However, the running time was relatively slow and unsuitable for real-time applications.

It is also shown that the optical flow features are a powerful tool to describe motion between video frames and are useful for MOT. The traditional Lucas–Kanade algorithm [95] for optical flow estimation has been used in MOT [61, 94]. Because DL approaches [96] are able to get a smoother and more robust estimation of optical flow compared to traditional methods, the tracking accuracy is expected to be promoted with deep optical flow features. Zhou *et al.* [97] proposed a CNN that estimates the displacement for each tracklet from the previous frame to the current frame. Then, a deep continuous conditional random field with asymmetric pairwise terms is used to refine the displacements. Spatial similarity score based on the predicted object displacements is combined with appearance similarity score computed using the Siamese network, and then used by the Hungarian algorithm for linking existing tracklets to new detections. Deep flow features have also been used in end-to-end MOT methods (see §2.4); however, they have been only used to associate detections between consecutive frames and have not shown reliable tracking accuracy for long-term frames association.

2.3.5 Interaction Features

Most MOT frameworks use a feature extractor independently for each object without accounting for feature interaction between objects, something that may lead to sub-optimal discriminative feature learning. To tackle this issue, Graph Neural Networks (GNNs) have recently been introduced in MOT. Weng *et al.* [98] for instance, proposed a novel feature interaction mechanism based on GNN. Specifically, appearance and motion features are extracted for each object and then fused through concatenation. A GNN is then constructed where each node represents the features of either past tracked objects or newly detected objects in the current frame. Each node feature is updated by aggregating all features from other nodes, which allows for interaction among object features where each object feature can be adapted with reference to other features. The affinity estimation is achieved via edge regression in the GNN. Finally, a

batch triplet loss is utilized on the node feature of every GNN layer for discriminative feature learning. The authors noticed that the discriminative ability considerably increased within a couple of GNN layers when accounting for feature interaction. Their approach, called GNN3DMOT, achieved 82.24 MOTA on KITTI and 6.21 AMOTA on the nuScenes validation dataset. Multiple other works followed the same process and used GNNs to estimate similarity scores between existing tracklets and new detections [99, 100]. The work of Weng *et al.* [98] and other GNN based methods alike [99, 100] have shown poor tracking performance; however, they pointed out an under-explored topic which is incorporating object interactions during tracking.

Some recent works started to explore transformer-style architectures [8] for tracking while taking into account the interaction between different objects [101–103]. TransTrack [102] performs tracking using attention-based query-key mechanism which relies on heuristic post processing (IoU matching) to generate final tracklets. Meinhardt *et al.* [101], proposed TrackFormer, an extension of the recent DEtection TRansformer (DETR) [104] object detector. TrackFormer uses the track query embeddings to follow object location changing over time in an autoregressive manner and adopts object queries as in DETR [104] to deal with newborn objects. For Trackformer, the MOTA on MOT17 benchmark is 65.0.

While transformer-based trackers [101–103] have shown that transformers can associate objects in a video using only attention operations, the tracking performance of transformer-based trackers is not state-of-the-art for several reasons. First, these methods used only simple features (such as spatial information and velocity) for representing each object, and they do not rely on any additional matching such as appearance or motion modeling. Second, a scene generally contains a large number of objects. Modeling the spatio-temporal relationships of these objects with a general transformer architecture is not efficient because it does not consider the spatial-temporal structure of the objects. Third, most of the transformer-based trackers [101, 102] consider interaction only within two adjacent frames. In contrast, in Chapter 5, we integrate a transformer-style network and apply it with objects' appearance embeddings combined with spatial and temporal information. The experimental results show that the proposed approach

performs favorably against or comparable to state-of-the-art methods while significantly outperforming the competing methods that consider interactions and dependencies between the actors.

2.4 Joint Detection and Tracking

In real-world applications such as self-driving vehicles, system latency is a primary concern. However, existing feature extractors (see §2.3) are often time-consuming. Furthermore, existing MOT frameworks generally treat detection and association separately, which induces accumulated latency and errors. To avoid such problems, the ability to use a shared neural network to provide the “heavy lifting" required for detection and tracking may reduce latency and improve accuracy (via joint learning).

Recently, a common trend in MOT consisted of leveraging off-the-shelf detectors and extending them to perform both detection and tracking in an end-to-end framework. This has shown to be helpful in different ways, including a reduction in system latency and also removal of error accumulation of the cascaded detection-tracking methods.

Bergmann *et al.* [105] presented an efficient tracking framework, Tracktor, which leverages an object detector and converts it into a tracker under the assumption that objects slightly move between consecutive frames. Tracktor adapts the Faster R-CNN detector [33] by adding a regression head to regress the location of a bounding box of an object in the new frame from the previous frame. While Tracktor achieved top performances on MOT benchmarks [58, 59], it suffers from a few drawbacks: (i) it only works well on high frame-rate videos where inter-frame motion is low, (ii) it does not capture the appearance of the tracked object, which makes the model unreliable when multiple objects are in close proximity, and also in the case of temporary occlusion. To overcome the second drawback, Bergmann *et al.*presented an extension to Tracktor, called Tracktor++, by integrating appearance features generated by a separate Siamese network which led to better tracking accuracy, but with the sacrifice of efficiency, partially defeating the

benefits of Tracktor. When Tracktor++ was published, it achieved state-of-the-art performance on MOT16 (54.4 MOTA) and MOT17 (53.7 MOTA) benchmarks.

Multiple end-to-end MOT frameworks have then been constructed based on Tracktor. Liu *et al.* [106] applied a new graph representation to Tracktor. The proposed graph representation uses both the extracted features of individual objects and also the interactions among objects, and this has been shown to enhance the tracking performance of Tracktor by +5.6 MOTA on MOT16 and +2.9 MOTA on MOT17. Xu *et al.* [107] presented an end-to-end MOT training framework, using a differentiable approximation of MOT metrics [1, 108] in the loss functions. They showed improvements for the Tracktor approach on MOT16 and MOT17 benchmarks (+0.4 and +0.2 MOTA, respectively) when extended with their training framework.

To tackle the issue of high inter-frame motion that Tracktor and its extensions suffer from, some end-to-end MOT methods used CNNs to estimate inter-frame offsets between detections of consecutive frames using pairs of frames as input instead of using only one frame. Feichtenhofer *et al.* [109] extended R-FCN [87] detector into the Siamese network to simultaneously perform detection and also compute convolutional cross-correlation between feature responses of consecutive frames in order to estimate inter-frame offsets between bounding boxes. On top of the features, they proposed to employ Region-of-Interest (RoI) pooling layer [87] for tracking which regresses bounding box transformations (translation, aspect ratio, etc.) across frames.

Similarly, Zhou *et al.* [7] proposed CenterTrack, which extends the CenterNet detector [13] to estimate inter-frame offsets of bounding boxes. Each object in CenterTrack is represented by the center location of its bounding box, and all center points are linked across frames. Their proposed extension of CenterNet takes as input the current frame, the previous frame, and a heatmap rendered from the existing tracklets centers and performs detection as well as regression of displacement vectors, from the current object center to its center in the prior frame, for each detected object. For the association stage, a greedy matching algorithm is applied based only on the distance between the detected centers' locations in the prior frame and the predicted displacements. CenterTrack is the current state-of-the-art method in many tracking

benchmarks [11, 58, 59]. It obtained 67.8 MOTA on the MOT17 benchmark and 88.83 MOTA on KITTI. CenterTrack can be extended easily to 3D tracking and even has shown the best performance (4.6 AMOTA) for monocular 3D tracking methods on nuScenes dataset [3]. While Zhou *et al.* [7] and Feichtenhofer *et al.* [109] have solved the issue of large inter-frame motion and achieved good results, they only link objects in adjacent frames and are not able to perform long-range matching, which makes them unable to correctly link detections when the object is temporarily occluded.

For better long-term association, Wang *et al.* [110] proposed the first attempt of simultaneous detection and tracking approach based on appearance features. Their model named JDE, or "Joint Detection and Embedding" employs a unified network to jointly output detections and their corresponding appearance embeddings. JDE extends YOLOv3 [38] with a re-identification branch trained with triplet loss [85] and jointly learnt with detection losses. For each new frame, JDE updates the appearance embedding of the tracklet using a combination of all embeddings for the detections belonging to the tracklet. The similarity score is a combination of cosine distance between appearance embeddings and Mahalanobis distance between the current detection and the predicted bounding box of the tracklet computed using the Kalman filter. The Hungarian algorithm is then employed to find the optimal assignments. JDE achieved 64.4 MOTA on the MOT16 benchmark at 20.2 FPS. JDE showed a speedup in inference time but was unable to achieve a tracking accuracy improvement over non-jointly-learned alternatives.

Recently, Zeng *et al.* [14] proposed PnPNet, an end-to-end model that performs joint detection, prediction, and tracking. All three modules share the same backbone network, and the full model can be trained end-to-end. The model takes the bird's eye view representation of multi-sweep LiDAR point clouds and HD map as input and passes it to a CNN backbone with multi-scale feature fusion to extract intermediate feature representation used for the three tasks. At each frame, the extracted features are combined with estimated motion features to construct a single feature representation for each detection. The constructed feature representations of all detections belonging to a tracklet are passed through LSTM to output feature representation for

each tracklet. Feature representations of each pair of new detection and existing tracklet are passed to MLP to compute the similarity score. Finally, the Hungarian algorithm is applied for tracklets-detections assignments. PnPNet is tested only on the validation set and only on the car class of the nuScenes dataset [3], leaving its generalization ability questionable.

End-to-end methods that perform simultaneous detection and tracking have shown better results than non-jointly-learned alternatives and with less running time. However, existing appearance-based end-to-end methods did not yield significant improvements in tracking accuracy. In contrast, in Chapter 4, we present an efficient joint detection and tracking model that relies on an appearance-based object matching network jointly learned with an underlying object detection network, and which has shown to have significant advantages in robustness, compared to current end-to-end methods, when applied to challenging tracking data.

2.5 Conclusions

Geo-localization of static traffic-control objects is an important component of the autonomy system for self-driving vehicles. Multiple prior works have studied this task. Traditional approaches are based on the triangulation technique, which has yielded poor performance in realistic scenarios like dense areas where many objects are present. Self-driving vehicles are expected to encounter many traffic lights in a single intersection, and triangulation-based methods are more likely to fail. Recently, a few attempts tried to explore DL-based end-to-end methods that jointly detect, associate, and geo-locate static objects in a single model. End-to-end models have shown very promising results even when multiple identical objects are present in the scene. However, they are fundamentally not suitable for online autonomous driving application. Also, one major issue in the static objects geo-localization domain is the absence of a common benchmark where different approaches can have a fair comparison.

Scene understanding is a major component of the autonomy system for self-driving vehicles. It involves a profound video analysis where all static and moving objects are identified and tracked. As a result, both object detection and multi-object tracking tasks have been studied for

decades by computer vision specialists and researchers. Recently, the significant improvements in detection performance thanks to modern DL techniques have served as a strong basis for researchers to tackle, once inevitable, tracking problems. Furthermore, with the success of DL, researchers' focus drifted from developing novel optimization algorithms for matching to learning better discriminative features while using simple matching algorithms and yet achieving better performance.

Studies have shown that appearance features are essential for good-performing trackers, and CNNs are a powerful tool for extracting them. Besides, integrating future motion prediction as well as flow features have been shown to boost tracking performance. These features can be used solely or in conjunction with appearance features. Recently, further attempts to extract better representative features consisted of incorporating features interaction between objects. This has opened a new direction for the problem of tracking, where many researchers are showing great interest.

Recent end-to-end methods of joint detection and tracking have shown solid results in reducing system latency and enhancing the performance of both tasks. However, existing appearance-based end-to-end methods did not yield significant improvements in tracking accuracy.

Chapter 3

Static Objects Geo-localization

Accurately estimating the position of static objects, such as signs and traffic lights, from the moving camera of a self-driving car is a challenging problem. In this chapter, we present a method for estimating 5D pose, tracking, and localizing spatially-compact static objects from a single camera of a self-driving car. Each video frame is assumed to be associated with a reasonable ego-pose of the camera, as is readily available in open-source self-driving data sets. The proposed method consists of neural networks that address each of the main components of the system, combined to allow joint-optimization via learning to improve overall performance. This work was published in WACV2021 [111] and the code is publicly available at: https://github.com/MedChaabane/Static_Objects_Geolocalization .

The top-level model takes a pair of geo-located video frames as input and outputs a set of localized objects (5 Degree-of-Freedom, or “5D" poses). For each input image, a network performs 5D pose regression for each detected object. Detected objects are represented with both appearance and pose information for learning how to associate them between frames. We employ an existing object detector but propose new networks for single-image object pose regression and cross-image object matching. The system applies these networks in a multi-object tracking paradigm to produce robust 5D poses for the set of tracked objects in a video sequence. The performance of the proposed approach is evaluated on traffic lights due to the availability of data. In principle, this method could be applied to other static object types as well.

The proposed object localization method consists of two models. The first is a pose regression network (§ 3.1) used to estimate the 5D pose of objects present in an RGB image. The second is an object matching network (§ 3.2) used to associate objects across a sequence of frames. The proposed approach is an online method, so it uses information derived only from past frames, making it suitable for use in self-driving vehicles and other streaming applications. At each given frame t, the network produces a set of 2D object detections in the image. For each detection, the 5D

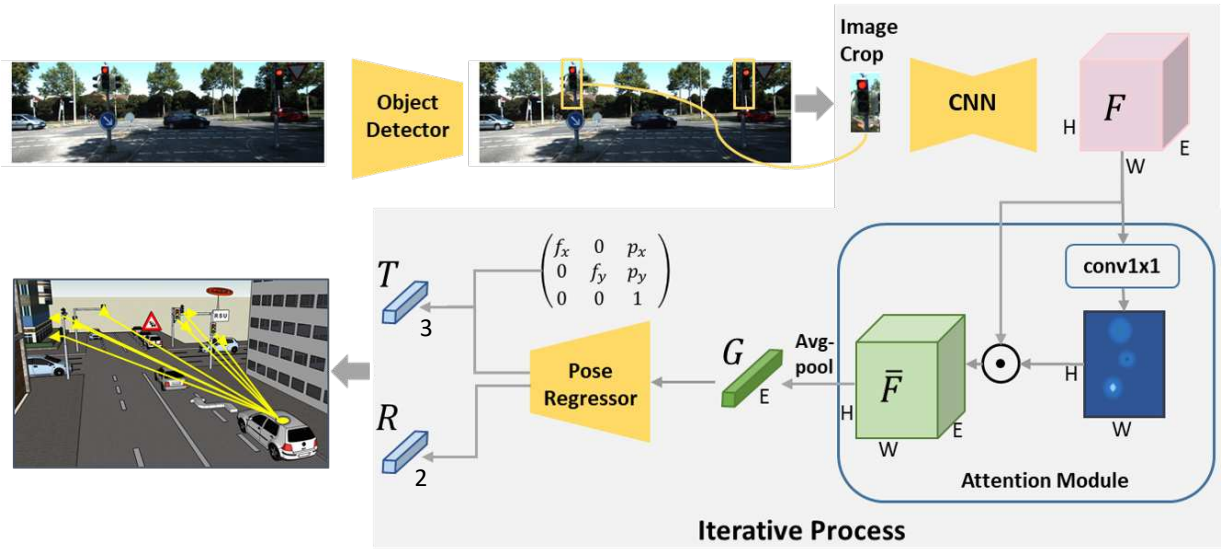

Figure 3.1: Single-image Object Pose Regression The model first computes bounding boxes (crops) of objects of interest from geo-located images. Each image crop is then processed with an encoder-decoder CNN to generate a feature map, F, which is processed by an attention module to yield $\bar{F}$. Using average pooling, a fixed-size geometry embedding G is created, which is then fed to the pose regressor to output the 5D pose.

pose is estimated. The current-frame detections are associated with tracks of previously detected objects using the object matching network. For each tracked object, the estimated 5D poses are aggregated over time to compute the final location and rotation. Object locations are aggregated by applying an LSTM network. In the following, details on the two main components, the pose regression network and the object matching network, are provided.

3.1 Pose Regression Network

Figure 3.1 illustrates the architecture of the proposed object pose regression model. The method is for application to spatially compact static objects, such as traffic lights or signs. As static objects of interest are tracked across frames, the per-frame pose estimates are used not only to refine the final 5D pose of the object, but also to help disambiguate matching objects across frames (see § 3.2.2). The network outputs 5D object pose vectors $p = [T, R]$ where $T = (T_x, T_y, T_z)$ represents the 3D translation vector of the center of the object in the camera coordinate system and $R = (R_x, R_y)$ represents the unit vector orthogonal to the object (the direction in which traffic

light or sign is facing) with respect to the camera coordinate frame. To estimate the pose, the network is trained using the euclidean loss $L_{\text{trans}}(T, \hat{T}) = \|T - \hat{T}\|_2$ for the translation regression, and the log hyperbolic cosine loss $L_{\text{rot}}(R, \hat{R}) = \sum_{a \in \{x,y\}} \log(\cosh(R_a - \hat{R}_a))$ for the rotation regression, where $p = [T, R]$ is the ground truth pose and $\hat{p} = [\hat{T}, \hat{R}]$ is the estimated pose. Instead of regressing the full translation vector T, the pose regression network is trained to regress the T_z component and the object's center position $c = (c_x, c_y)$ in image pixel space. This formulation provides better invariance to camera parameters. Projective geometry is used to recover the full translation vector $T_a = (c_a - p_a)T_z/f_a$ for $a \in \{x, y\}$, where f_x, f_y are the camera focal lengths, and (p_x, p_y) is the camera principal point offset.

The proposed pose regression network is a two-staged network. The first stage is a typical 2D object detection network [37, 112, 113]. The bounding boxes of the detected objects are padded by N_p pixels for each side to include more context and to take into account slight errors coming from the object detector model. Features from within each padded bounding box ("image crop") are used in the second stage to estimate object pose.

3.1.1 Geometry Embedding

The image crop is fed into an encoder-decoder network that maps an image of size $H \times W \times 3$ into a feature map $F \in \mathbb{R}^{H \times W \times E}$. Each pixel of the feature map is an E-dimensional vector representing the appearance information of the input image crop at each pixel location. From the feature map F, the embedding of the image crop is derived as follows. A spatial attention mechanism is employed to focus the embedding on the most salient parts of the image crop. The spatial attention distribution $a \in \mathbb{R}^{H \times W}$ is learned using 1×1 convolutions from the extracted feature maps F. The spatial attention map a is then normalized using softmax of the responses:

$$\bar{a} = \frac{\exp(a)}{\sum_{i=1}^{H} \sum_{j=1}^{W} \exp(a_{i,j})} \tag{3.1}$$

The normalized spatial attention map $\bar{a}$ is applied to weight the feature map F to generate the attention-weighted feature map $\bar{F} = \text{rep}(\bar{a}) \odot F$ ($\bar{a}$ is replicated for E times to match the size of F). Average pooling is then applied to $\bar{F}$ to obtain the geometry embedding $G \in \mathbb{R}^E$.

3.1.2 Pose Regressor

The pose regressor transforms the geometry embedding G into 5D pose estimates for each object crop in the input image. The pose regressor is composed of a rotation and a translation branch, each composed of fully connected layers. The rotation branch estimates the rotation vector R and is normalized before computing the loss. The translation branch estimates the T_z component of the translation vector and the object's center position $c = (c_x, c_y)$. The network is trained by minimizing the loss $L_{\text{pose}} = L_{\text{rot}} + \beta L_{\text{trans}}$.

3.2 Object Matching Network

The object matching network is responsible for associating objects between pairs of frames, allowing the system to track objects through the video sequence. At a high level, the architecture follows the Deep Affinity Network (DAN) of [84], but modifications are made to add pose features into the embeddings. With these changes, the object matching network supports joint learning of pose, appearance, and similarity to improve static object tracking. A summary of the architecture is presented in Figure 3.2. The entire architecture is divided into two sub-networks: a feature extractor and a matching head. Both are described in more detail below, following a description of data preparation and encoding, which is important for understanding the network design.

3.2.1 Data Preparation and Encoding

A pair of images n frames apart, I_t and I_{t-n}, are input to the object matching network along with the sets of bounding boxes of the detected objects, $B_t = [b_1^t, b_2^t, ..., b_{N_t}^t]$ and $B_{t-n} = [b_1^{t-n}, b_2^{t-n}, ..., b_{N_{t-n}}^{t-n}]$ respectively, with $1 \leq N_t, N_{t-n} \leq N_{max}$ where N_{max} is the maximum number of allowed detected objects in any frame. In order to provide more robustness

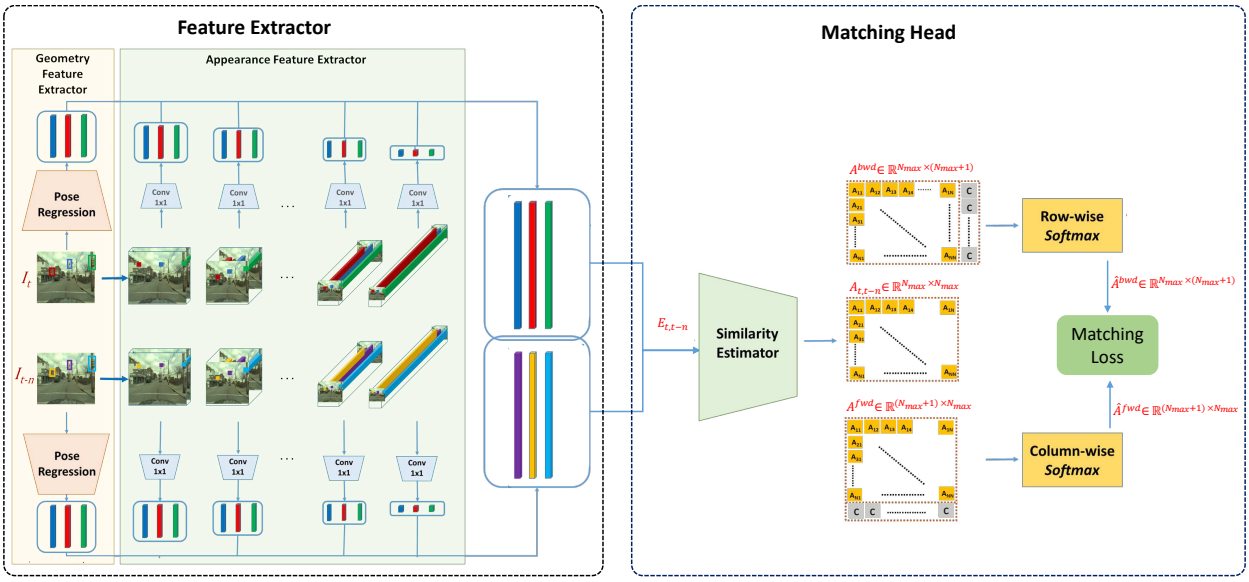

Figure 3.2: Object Matching Network. A pair of images n frames apart, I_t and I_{t-n}, along with the detected 2D bounding boxes, are input to the network. The feature extractor extracts a d-dimensional vector encoding pose and appearance information for each detected object in each frame. The matching head uses these to produce affinity estimations, matching objects across the two frames.

during inference, the matching network is trained using image pairs separated by a variable amount of time. The lower bound of the interval is a single frame of separation. The upper bound is a number of frames representing a few seconds of time, to allow capturing the situation where the camera has moved significantly between two observations of the same object. When generating the training data, we sample uniformly from the range between the lower and upper time intervals (in number of frames between image pairs), and the training data is expressed as $X_{train} = \{(I_t, I_{t-n}) \mid n \in [1, n_{gap}]\}$, where n_{gap} is the maximum frames of separation between image pairs. Each image in a pair is resized to a fixed-size.

For each training pair, two ground truth matching matrices M^{fwd} $(t - n \rightarrow t)$ and $M^{bwd}(t \rightarrow t - n)$ are created, representing the forward and backward associations, respectively. The ground truth matching matrices consist of entries $[i, j] \in \{0, 1\}$, and have dimensions $N_{max} \times (N_{max} + 1)$ to allow for unassociated objects. The elements $M^{fwd}[i, j]$ and $M^{bwd}[j, i]$ encode the association between the object observations b_i^{t-n} and b_j^t. A value of 1 encodes an association, meaning that the observations pertain to the same physical object. Entities entering and leaving the scene are

encoded with $M^{fwd}[i, N_{max}+1] = 1$ and $M^{bwd}[j, N_{max}+1] = 1$, respectively. Everywhere else, the value is 0.

3.2.2 Feature Extractor

The feature extractor network extracts the compact features used to associate objects between image pairs. The pair of frames (I_{t-n}, I_t) are fed in parallel to the feature extractor where the two branches share the same set of weights. This network is composed of a geometry feature extractor (yellow box in Figure 3.2) and an appearance feature extractor (green box in Figure 3.2). The underlying idea of the feature extractor is that the affinity scores between objects can be computed based on visual and geometric cues.

The proposed approach focuses on autonomous driving scenes, where the video frames are from a monocular camera mounted on a car moving on the road plane, and the tracked targets are static objects near the road. Thus, geometry features that describe the location and rotation of objects can be helpful to discriminate between objects. Benefiting from reliable pose estimation, the same physical object in the 2 frames I_{t-n} and I_t is expected to have similar estimations of location and rotation in a *common reference frame*.

Thus, from any frame I_t, the proposed pose regression network is used to output the estimated location and rotation of the detected object. The estimated pose is then transformed into the camera coordinates system of a common reference frame I_{ref}; in the implementation, we chose the reference frame to be the first frame for each video. The pose is transformed into the common reference frame I_0 from any other frame I_t as follow:

$$\left[\hat{T}^0 \mid 1\right] = \zeta_t^{-1} \cdot \zeta_0 \cdot \left[\hat{T}^t \mid 1\right], \tag{3.2}$$

$$\left[\hat{R}^0 \mid 0\right] = \zeta_t^{-1} \cdot \zeta_0 \cdot \left[\hat{R}^t \mid 0\right], \tag{3.3}$$

where ζ_0, ζ_t are the extrinsic matrices of I_0 and I_t, respectively.

The geometry embedding G used in the pose regression network contains information about the geometry of the objects as well. Thus, the features of G are concatenated with the 6 pose values described above to construct $f_{g,i} \in \mathbb{R}^{6+E}$ geometry feature descriptor for the i^{th} detected object.

Table 3.1: Details on the architecture of the appearance feature extractor network used in the object matching network. The layers used in the final embedding are denoted in the column "Label" as $f_{R_n} | n \in [1, 10]$.

	Layer	Output size	Kernel size	Stride	Receptive field	Label
1	$3\times$ conv	$H \times W \times 3$	3×3	1	7	-
	Max Pool	${}^{H}/_{2} \times {}^{W}/_{2} \times 64$	3×3	2	9	-
4	$2\times$ conv	${}^{H}/_{2} \times {}^{W}/_{2} \times 64$	3×3	1	17	-
	Max Pool	${}^{H}/_{4} \times {}^{W}/_{4} \times 128$	3×3	2	21	f_{R_1}
6	$3\times$ conv	${}^{H}/_{4} \times {}^{W}/_{4} \times 128$	3×3	1	45	-
	Max Pool	${}^{H}/_{8} \times {}^{W}/_{8} \times 256$	3×3	2	53	f_{R_2}
9	$2\times$ conv	${}^{H}/_{8} \times {}^{W}/_{8} \times 256$	3×3	1	85	f_{R_3}
11	$1\times$ conv	${}^{H}/_{8} \times {}^{W}/_{8} \times 512$	3×3	1	101	-
	Max Pool	${}^{H}/_{16} \times {}^{W}/_{16} \times 512$	3×3	2	117	f_{R_4}
12	$3\times$ conv	${}^{H}/_{16} \times {}^{W}/_{16} \times 512$	3×3	1	213	f_{R_5}
15	$2\times$ conv	${}^{H}/_{16} \times {}^{W}/_{16} \times 512$	3×3	1	277	f_{R_6}
17	$2\times$ conv	${}^{H}/_{16} \times {}^{W}/_{16} \times 512$	3×3	1	341	-
	Max Pool	${}^{H}/_{32} \times {}^{W}/_{32} \times 512$	3×3	2	373	f_{R_7}
19	$3\times$ conv	${}^{H}/_{32} \times {}^{W}/_{32} \times 512$	3×3	1	565	f_{R_8}
22	$3\times$ conv	${}^{H}/_{32} \times {}^{W}/_{32} \times 1024$	3×3	1	757	-
	Max Pool	${}^{H}/_{64} \times {}^{W}/_{64} \times 1024$	3×3	2	821	f_{R_9}
25	$2\times$ conv	${}^{H}/_{64} \times {}^{W}/_{64} \times 1024$	3×3	1	1077	-
	Max Pool	${}^{H}/_{112} \times {}^{W}/_{112} \times 1024$	3×3	2	1205	$f_{R_{10}}$

Given a monocular imaging system, the objects and close surroundings are expected to maintain their visual appearance over short time spans. To extract appearance features, a convnet is employed, inspired by the increased performance of CNNs with smaller filter size (3×3) and deeper architectures such as VGG [114]. We conducted a model search to identify the best performing architecture for the appearance feature extractor. The architecture that achieved the highest performance is presented in Table 3.1. It consists of 26 convolutional layers and 7 max-pooling layers. Each convolutional layer is followed by batch normalization [115] and a ReLu activation function.

For each detected object, feature vectors are extracted from the object's center location as regressed from the pose regression network. If the center of the i^{th} object is at position $(X_{i,r}, X_{i,c})$ in the input frame of size $W \times H$, then for a feature map $\mathcal{F}_j$ of size $W_j \times H_j \times C$, C-dimensional feature vector is extracted at position $(\frac{X_{i,r}}{H} H_j, \frac{X_{i,c}}{W} W_j)$ as the corresponding feature vector for the i^{th} object.

Appearance features $[f_{Ri} \mid i \in [1, 10]]$ from ten layers matching varying receptive fields are extracted, where the corresponding receptive field for each extracted feature vector is shown in Table 3.1. This multi-resolution architecture helps to simultaneously capture fine geometric details as well as higher-level semantics of the surroundings. Using a multi-resolution feature vector has been shown to outperform those using a single receptive field (see § 3.3.4).

After extracting appearance and geometry features for each detected object, both are concatenated to obtain $f_i = f_{g,i} \bigoplus f_{a,i} \in \mathbb{R}^d$ $(d = A + 6 + E)$ which is a fused feature descriptor for the i^{th} detected object.

3.2.3 Matching Head

Using the extracted feature descriptors, the tensor $E_{t,t-n} \in \mathbb{R}^{N_{max} \times N_{max} \times 2d}$ is built, where $E_{t,t-n}[i, j, :] = f_{i,t-n} \bigoplus f_{j,t}$ is the concatenation of the feature vectors of the i^{th} object of I_{t-n} and the j^{th} object of I_t. To construct fixed size tensor $E_{t,t-n}$, the rest of the tensor is padded with zeros. This formulation allows the model to compute object affinities in a single forward pass. $E_{t,t-n}$ is fed to a similarity estimator network composed of 6 layers of 1×1 convolutions. The output of the matching head is affinity matrix $A_{t,t-n} \in [0, 1]^{N_{max} \times N_{max}}$. Note that 1×1 convolutions are used so that the computation of $A_{t,t-n}[i, j]$ is computed using only the feature vectors $f_{i,t-n}$ and $f_{j,t}$ and will not be affected by other feature descriptors.

Since we learn similarity between feature descriptors, there is no guarantee that the resulting scores are symmetric when matching backwards or forwards across frames, i.e., when matching from $(t \rightarrow t-n)$ versus $(t-n \rightarrow t)$. As a result, the affinities in both directions are computed using a separate affinity matrix for each, denoted with superscripts "bwd" and "fwd" in the following.

To allow for objects that should not be associated between the frames (objects new to the scene, or departed), a column is added to $A_{t,t-n}$ filled with constant value c to obtain the matrix A^{bwd}. Softmax is applied to each row of A^{bwd} to obtain matrix $\hat{A}^{bwd}$, representing the final affinity including non-matched scores. The choice of c is not overly sensitive – the network will learn to assign affinities greater than c for true matches.

Each $\hat{A}^{bwd}[i,j]$, represents the estimated probability of associating b_i^t to b_j^{t-n}. $\hat{A}^{bwd}[i, N_{max}+1]$ represents the estimated probability that b_i^t is an object not present in frame $t-n$. Similarly, the forward affinity matrix is constructed, using the transpose matrix $A_{t,t-n}^T$ to which we add a column filled with constant value c and then softmax is applied to each row to obtain matrix $\hat{A}^{fwd}$. During the inference, the similarity score between b_i^t and b_j^{t-n} is given as the average of $\hat{A}^{bwd}[i,j]$ and $\hat{A}^{fwd}[j,i]$.

3.2.4 Joint Loss Function

The object matching network is trained with the loss function $\mathcal{L}_{\text{match}}$ defined as the average of the two losses $\mathcal{L}_{\text{match}}^{fwd}$ and $\mathcal{L}_{\text{match}}^{bwd}$, where $\mathcal{L}_{\text{match}}^{bwd}$ is the error of matching bounding boxes in frame t to those in frame $t-n$ and $\mathcal{L}_{\text{match}}^{fwd}$ is the error of matching bounding boxes in frame $t-n$ to those in frame t. The expression of the matching loss is given as follows, where "*" represents "fwd" or "bwd" as appropriate:

$$\mathcal{L}_{\text{match}}^{*} = \sum_{i=1}^{N_{max}} \sum_{j=1}^{N_{max}+1} M^{*}[i,j] \log(\hat{A}^{*}[i,j]), \tag{3.4}$$

$$\mathcal{L}_{\text{match}} = \frac{\mathcal{L}_{\text{match}}^{fwd} + \mathcal{L}_{\text{match}}^{bwd}}{2(N_t + N_{t-n})}, \tag{3.5}$$

Training optimizes the joint affinity and pose estimation losses as defined in Eq. (3.6). The loss of the pose estimation task is computed as the average of the pose losses of all object detected in both frames. Pose and affinity losses are traded-off with a scalar λ.

$$L_{\text{joint}} = \mathcal{L}_{\text{match}} + \lambda(\frac{1}{N_t + N_{t-n}} \sum_{i=1}^{N_t+N_{t-n}} L^i_{\text{pose}}) \tag{3.6}$$

3.2.5 Multi-Object Tracking

The proposed Multi-Object Tracking (MOT) approach follows the tracking-by-detection paradigm. Given a new frame with the bounding boxes of the detected objects, the tracker computes the similarity scores between the already tracked m targets (each target consists of multiple instances from different frames) and the n newly detected objects using the object matching network. The score matrix is defined as $S = [s_i^j \mid 1 \leq i \leq m \text{ and } 1 \leq j \leq n + m]$, where s_i^j represents the similarity between the i^{th} target and j^{th} detection and it is computed as the maximum over the similarity between the instances of the i^{th} target before frame $t-1$ and the j^{th} detection at current frame t, s_i^{i+n} for $1 \leq i \leq m$ represents the likelihood of i^{th} target to not being matched to any of the new detected objects at frame t and is computed as the average of the values at last column in $\tilde{S}^1_{t-n,t}$ for the instances of i^{th} target and $s_i^j = -\infty$ for $j > n$ and $j \neq i$. Finally, the widely-used Hungarian algorithm [78] is adopted to derive the optimal assignments.

3.3 Experiments and Results

3.3.1 Datasets

We constructed the Traffic Lights Geo-localization (TLG) data set. TLG is derived from nuScenes [116], a popular open-source data set for autonomous driving. The nuScenes data contains 1000 scenes of 20 seconds (at 12Hz video rate), filmed in two cities (Boston and Singapore), in both night and day, and with three weather conditions (rain, sun, and clouds). Each scene comes with data from six cameras placed at different angles on the car.

Those scenes within road intersections containing Traffic Lights (TLs) are selected. For each scene in the nuScenes data set, and for each video clip from one of the six cameras, we iterated through key frames (2Hz), selecting TLs within 100 meters of the camera location. Each TL location was transformed from world coordinates to camera coordinates and then into 2D

homogeneous image coordinates, using the provided extrinsic and intrinsic camera calibration parameters. TL locations not visible to the camera are filtered. Finally, scenes are selected only if at least one TL is visible in 5 different key frames in one of the six cameras. With this process, we ended up with 348 scenes for training and 56 scenes for testing.

In the TLG data, each video clip (from different cameras) in each scene contains 240 RGB images (including 40 key frames) with a resolution of 1600×900. Images are augmented with camera pose information and camera metadata, including information about each visible TL: unique ID, 5D pose in world coordinates, 5D pose in the camera coordinates of the first frame, and TL type (horizontal or vertical).

Three sub-datasets are created for the main tasks, one each for pose, matching, and tracking. The “Traffic Lights 5D Pose" data contains around 66,000 snippets of TLs (60,000 for training and 6,000 for testing) along with their 5D poses. The “Traffic Lights Matching" data contains 200,000 pairs of images (170,000 for training and 30,000 for testing) along with bounding boxes of TLs and ground truth matching matrices between the two images. The average elapsed time between image pairs in the Traffic Lights Matching data set is 1.4 seconds (the maximum frames of separation between image pairs n_{gap} is set to 35), and on average, four traffic lights appear per image. The “Multi-Traffic Lights Tracking" (MTLT) data provides a detection and annotation file for each video following the format of [59].

We evaluated several other potential sources of data that we hoped could be used to evaluate the proposed static object localization approach. Unfortunately, beyond nuScenes, we were unable to find other useful data sets.

3.3.2 Implementation Details

The proposed approach was implemented using PyTorch [117]. All experiments were run on an Ubuntu server with an Nvidia TitanX GPU with 12GB of memory. The performance comparison of contemporary methods for all tasks evaluated in this work was produced using the original authors' publicly available code.

In the pose regression network, the 2D object detector is the same as used in PoseCNN [47]. It is pre-trained on COCO [36] and Mapillary [53] datasets. The bounding box padding, N_p, is set to be between 5-25 pixels, scaled based on the bounding box. The architecture used to extract feature map F is composed of a Resnet-18 encoder followed by 4 up-sampling layers as decoder. The geometry embedding dimension E is set to 128. The weight factor β is set to 0.1. The pose regression network is trained using SGD for 40 epochs with a momentum of 0.9, and a weight decay of 0.0005. The resultant weights are used for the initialization when training the object matching network.

For the object matching network, the maximum number of tracked objects, N, is set to 30, and c is set to 8. The frames were resized to 896 × 896. By experimental evaluation, the optimal dimensions of the appearance features vectors $f_{R1}, f_{R2}, \ldots, f_{R10}$ are set to 100, 80, 70, 60, 50, 40, 30, 30, 20 and 20 respectively, which results in a 634-dimensional (500 + 6 + 128) feature descriptor for each detected object. The object matching and pose regression networks are jointly trained for 130 epochs with a momentum of 0.9, a weight decay of 0.0008, and λ is 0.005. The pose network is initialized to pre-trained weights.

3.3.3 5D Pose Estimation

We compared the proposed pose regression model to those which take RGB images as input and regress directly 5D poses such as PoseNet [45], and PoseCNN [47]. To make the comparison fair, all methods use the same object detector [118] as in PoseCNN, and both PoseNet and PoseCNN are finetuned on the training data with the same loss function used to train the proposed pose regression network. We made the following modifications to the PoseNet and PoseCNN architectures in order to adapt them to the problem domain:

- PoseNet: The RGB input image is fed through the object detector to output bounding boxes of detected objects, and then for each detected object, an image crop of size 224×224 centered in the center of the bounding box is extracted and fed to PoseNet architecture. The final fully connected layer in PoseNet has also been modified to output a 5-dimensional pose

Table 3.2: Pose regression ablation study. In "w/o Attention" the attention module of the pose regression ($\bar{F} = F$) is removed. In "Joint Training", the regression model is trained jointly with the object matching model to minimize loss function L_{joint} in Eq. (3.6). "Baseline" indicates training the model as described, as a stand-alone network

Model	5D Pose Errors (mean/median)				Run time
	All objects		Near ($\leq$ 20m) objects		sec/frame
	Translation (m)	Rotation (°)	Translation (m)	Rotation (°)	
Ours (w/o Attention)	4.95 / 3.93	17.68 / 10.51	3.02 / 2.24	16.26 / 7.64	0.05
Ours (Baseline)	4.67 / 3.61	17.00 / 9.70	2.64 / 1.83	14.74 / 6.24	0.05
Ours (Joint Training)	**4.43 / 3.39**	**15.97 / 9.16**	**2.51 / 1.70**	**14.21 / 6.08**	0.05
PoseNet [45]	7.25 / 5.83	28.47 / 21.82	5.36 / 4.48	24.31 / 18.23	**0.04**
PoseCNN [47]	5.54 / 4.47	19.63 / 11.35	3.68 / 2.91	18.04 / 8.86	0.11

vector instead of a 7-dimensional pose vector. The weights of PoseNet were initialized with those for the pretrained model on Cambridge Landmarks dataset [45], as this has shown to give better performance than randomly initialized weights.

- PoseCNN: The output of the 3D rotation regression branch in PoseCNN is modified to output a 2-dimensional vector instead of a 4-dimensional vector. The weights of PoseCNN were initialized with those for the pretrained model on the YCB-Video dataset [47].

Table 3.2 presents a comparison of the proposed pose regression model against PoseNet and PoseCNN on the Traffic Lights 5D Pose data. The proposed pose regression outperforms both PoseNet and PoseCNN. As expected, TLs far away from the camera can be challenging to locate accurately. All methods have considerably lower pose errors when evaluating only on TLs within 20 meters. In the full data set, TLs can be up to 100 meters away from the camera. We show in a later discussion of end-to-end performance (see Table 3.6) that most of the translation error is concentrated in the depth axis, T_z.

To understand the effects of the attention module and joint training strategy, the performance of three variants of the pose regression network is compared as shown in Table 3.2. The inclusion of the attention module reduces the rotation and translation errors. This shows how focusing on some regions in the image crop helps the model to extract a better representation for 5D pose

regression. Training the pose regression and object matching networks jointly improves further the pose regression performance.

To see how the proposed pose regression model performs on different views of traffic lights, we performed an analysis of 5D pose errors on four views of traffic lights: Front, Back, Right, and Left. Each type is defined by the relative orientation of traffic light with respect to the camera coordinates. We found out that the controlling traffic lights (Front) have lower translation errors and higher rotation errors compared to other types.

Table 3.3: Results analysis of 5D pose errors of the proposed pose regression network on four views of traffic lights: Front, Back, Right and Left.

Model	5D Pose Errors (mean/median)			
	All objects		Near ($\leq$ 20m) objects	
	Translation (m)	Rotation (°)	Translation (m)	Rotation (°)
Front ($R_z \leq -0.96$)	4.16 / 3.14	18.12 / 11.81	2.28 / 1.53	16.15 / 8.05
Back ($R_z \geq 0.96$)	4.41 / 3.34	15.70 / 8.53	2.51 / 1.71	13.74 / 5.47
Right ($R_x \geq 0.96$)	4.58 / 3.52	14.95 / 8.08	2.66 / 1.81	13.36 / 5.16
Left ($R_x \leq -0.96$)	4.53 / 3.49	15.12 / 8.11	2.60 / 1.78	13.42 / 5.24

3.3.4 Object Matching

To highlight the impact of the feature extractor of the object matching network, we report matching accuracy after changing the feature extractor component in Table 3.4. In this ablation study, we measure the impact of using only appearance features, only geometric features, using both appearance and geometric features, and joint training. Additionally, we measure variants of the appearance features when larger or smaller receptive fields are used, and we show variants of the geometric features when using only the 5D values or when combining the 5D values with the vector G from the pose regression network.

In Table 3.4 and the following text, “AFE" will indicate using only appearance features, and “GFE" will indicate using only geometric features in the object matching network. AFE outperformed single RF based architectures (Resnet-50 and VGG-16) by more than 4.9

Table 3.4: Object matching network ablation study. AFE uses only the appearance features. GFE uses only the geometry features. For AFE, also shown is the impact on receptive field (RF) sizes. For GFE, we show with and without including the pose regression feature vector G.

Object Matching Feature Extractor	mAP	Runtime
Resnet-50 [119]	0.744	0.1
VGG-16 [114]	0.824	**0.08**
AFE (RFs $\leq$ 213 only)	0.857	0.11
AFE (RFs $>$ 213 only)	0.839	0.12
AFE	0.873	0.12
GFE (5D only)	0.825	**0.08**
GFE (5D + G)	0.831	**0.08**
AFE + GFE	0.912	0.14
AFE + GFE (Joint Training)	**0.928**	0.14

percentage points, demonstrating the benefit of multi-resolution networks for our application. We found that appearance features extracted from small RFs perform better than those extracted from larger RFs, as illustrated when comparing AFE (RFs $>$ 213) and AFE (RFs $\leq$ 213). This fact is supported by comparing Resnet-50 (RF size = 483) and VGG-16 (RF size = 212), where VGG outperforms Resnet-50. Combining features from both small and large RFs (AFE) results in mAP gain of 1.6 percentage points. This can be explained by the fact that features from small RFs will focus on low level information such as color, texture, and shape, while features from large RFs will have richer contextual information that can be beneficial in some challenging cases.

By comparing the performance of AFE and GFE, we can conclude that appearance is more important than geometry for the object matching network. However, including the geometry cues helps to increase the mAP by 3.9 percentage points over appearance alone. We argue that the advantage gained from geometry features comes when TLs look similar and are close in image space. In those cases, TLs will also have similar backgrounds and thus produce similar appearance embeddings. The joint training strategy provides the remaining improvements, increasing the object matching network's mAP by 1.6 percentage points when compared to stand-alone training.

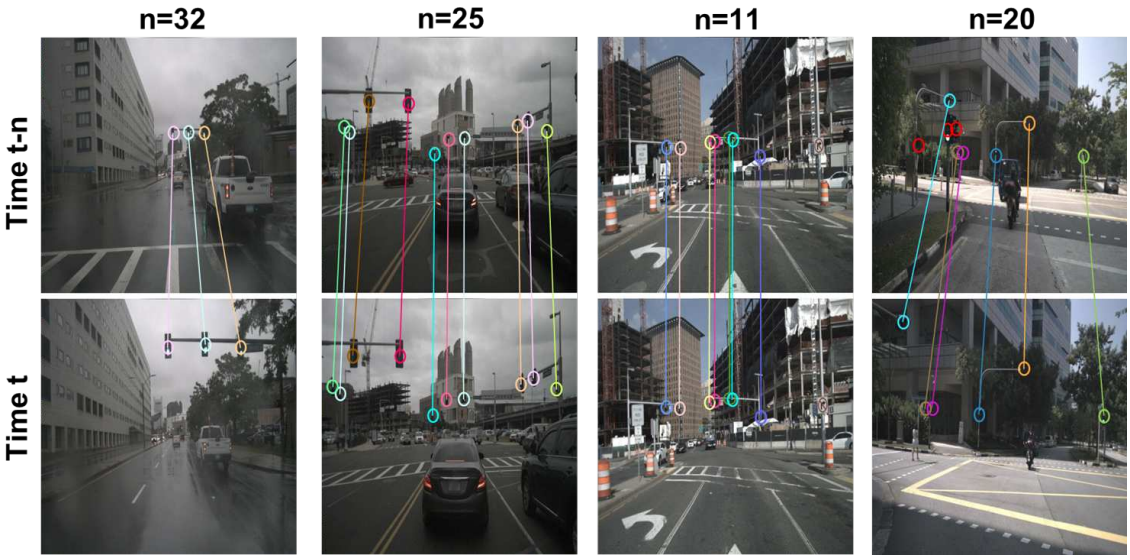

Figure 3.3: Object matching examples. Each column of the figure shows a pair of frames separated by n frames. Object matching remains robust to illumination and weather conditions and existence of multiple similar TLs in the frames.

Figure 3.3 shows examples of the object matching network's output from the Traffic Lights Matching data. We observe that the association appears robust to illumination and weather conditions. Also, even with the existence of multiple similar looking TLs at very close locations in the image space, the network is able to correctly associate the TLs. The chosen examples in Figure 3.3 are random. We noted similar level of performance by the object matching network for all the tested examples.

3.3.5 Multi-Object Tracking

We evaluate the performance of our tracker using MOT metrics [1] and compare its performance with three contemporary online MOT algorithms that are known to have reproducible results with publicly available code (Table 3.5). By only using appearance features (AFE), the proposed tracker achieves 81.29 in terms of MOTA which is higher than appearance-based trackers (DMAN and DeepSORT), demonstrating the strength of the multi-resolution appearance features. By using only geometry features (GFE), the proposed tracker achieves 74.12 in MOTA. By using both appearance and geometry features, the tracking accuracy is increased to 85.52 in MOTA, out-performing the other methods. The proposed tracker

Table 3.5: Comparison of the proposed method and contemporary MOT trackers on the MTLT test sequences. The standard CLEAR-MOT metrics [1] are utilized: MOTA (multi-object tracking accuracy), MOTP (multi-object tracking precision), MT (number of mostly tracked trajectories), ML (number of mostly lost trajectories), IDS (number of identity switches) and FPS (frame per second). ↑ and ↓ indicate higher or lower values are preferred

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDS ↓	FPS ↑
DMAN [120]	80.79	82.40	61.12	12.91	103	3.3
DeepSORT [4]	77.69	77.81	56.34	**9.41**	69	**17.2**
Tracktor++ [105]	83.31	**86.73**	66.54	9.71	82	2.64
Ours (GFE)	74.12	75.32	51.17	20.64	162	9.7
Ours (AFE)	81.29	82.18	62.37	12.66	96	6.1
Ours (GFE + AFE)	**85.52**	85.14	**69.57**	10.79	**61**	5.3

is twice as fast as Tracktor++, which has somewhat similar performance for many of the metrics other than IDS, where the proposed method is much better.

The proposed tracker's performance as captured by the MT metric is significantly better, suggesting the tracker generates more integrated trajectories by combining geometry and appearance cues. Similarly, the tracker's identity switches (IDS) value of 61 is best. Both MT and IDS are critical metrics when the output of the tracker is used to generate an aggregated pose estimate, as in our application.

3.3.6 Object Geo-localization

The end goal for our application is geo-locating static objects for HD Maps. The performance is evaluated in this regard by comparing predicted and ground truth geo-locations of traffic lights in the TLG data set. The proposed approach is compared with MRF-triangulation [5], and SSD-ReID-Geo [6], given that they are the only two approaches that have publicly available codes. By analyzing the errors of different methods (Table 3.6), it is worth noting that errors along Z-axis (depth) are considerably higher than errors along X and Y axes, which is typical for monocular vision-based systems. When localizing traffic lights, errors along Z-axis are less troubling than lateral or vertical errors. This is because the perception of whether or not a traffic light pertains to the self-driving car (i.e., the lane the car is in) is more affected by its horizontal position above the

road than the depth along the roadway. A lateral error of 2m could cause confusion about which lane the light controls. On the other hand, a depth error of a few meters is unlikely to cause such confusion. The proposed method shows a median error in the X and Y axes of less than 20cm, and a mean error within 25cm. The median depth error (Z axis) of about 1.5m is well within the accuracy bounds of the problem domain.

Table 3.6: Translation Error (TE) along X, Y and Z axes

Model	TE along X-axis (m)			TE along Y-axis (m)			TE along Z-axis (m)		
	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std
Ours	**0.25**	**0.16**	0.15	**0.23**	**0.15**	0.14	**2.24**	**1.47**	1.28
MRF-triangulation	0.31	0.24	0.12	0.35	0.27	0.15	4.75	3.89	1.92
SSD-ReID-Geo	0.64	0.51	0.37	0.51	0.45	0.33	3.77	2.85	1.68

Given the uneven error contribution, we decided to evaluate the different approaches using the Mahalanobis distance (see Fig. 3.4, B). In this case, positive predictions are defined as those within an elliptical shape from the ground truth location, defined by 3 units of Mahalanobis distance. This corresponds to an ellipse defined with semi-axes a=0.4, b=0.39, and c=3.84, along the X, Y, and Z axes, respectively. This distance threshold requires that the vertical and horizontal location of the prediction be within 40cm, while the depth error is tolerated to about 4m. The advantage of the Mahalanobis distance is that it provides much tighter thresholds in the X and Y axes while allowing more tolerance in depth, making it more suitable for our application. Figure 3.4 compares the precision/recall of the proposed approach against MRF-triangulation and SSD-ReID-Geo using two distance thresholds: 2m Euclidean distance and 3 units of Mahalanobis distance.

The proposed approach leads to more accurate geo-localizations than the other methods. The proposed approach outperforms MRF-triangulation thanks to the efficiency of the proposed pose regression model over the depth estimation in [5], and the joint learning employed by the proposed approach. SSD-ReID-Geo uses only pairs of frames when estimating object poses. For a fair comparison, the proposed approach is also tested using only frame pairs (Figure 3.4, denoted with †). The proposed approach outperforms SSD-ReID-Geo, even with this restriction. We also

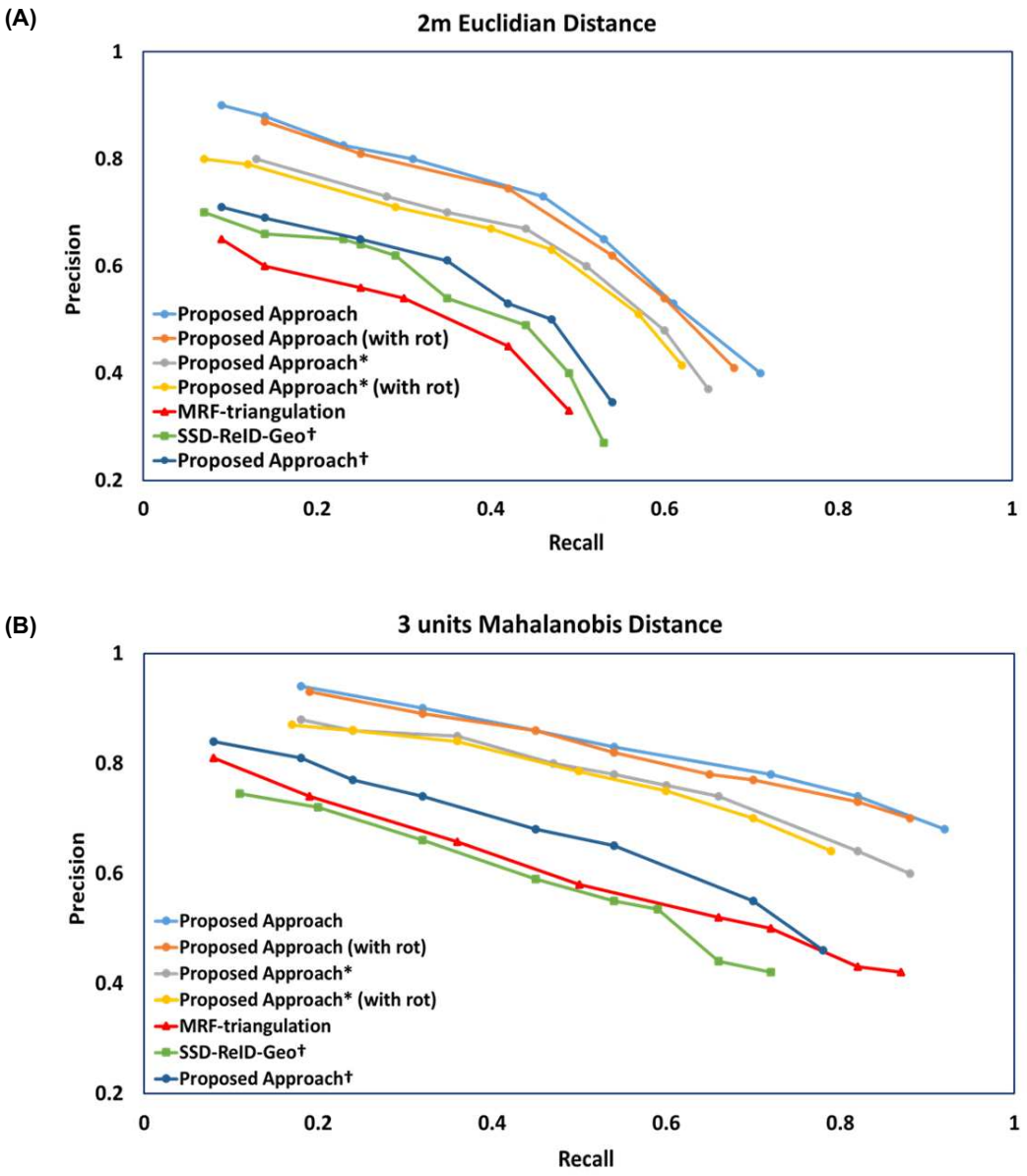

Figure 3.4: Comparison of the performance of the proposed approach for static object geo-localization against MRF-triangulation [5] and SSD-ReID-Geo [6]. An estimated geo-location is a true positive if it is within a threshold distance of a ground truth point. Methods marked with * use only key frames (2fps) for testing, methods marked with † are tested with only frame pairs, and "with rot" means that true positives must also be within 20° of the true orientation.

observe that using the Mahalanobis distance, the PR curve of SSD-ReID-Geo becomes lower than MRF-triangulation due to the added restrictions along X and Y axes.

The performance of the proposed approach improves by aggregating more information from more frames. The proposed approach can perform anytime prediction (even using only 1 frame) by

rapidly producing an initial estimate using the single-frame pose regression, and then improving the estimate by tracking across frames. This improvement is shown in Figure 3.4, when comparing the performance of the proposed approach under various settings: using pairs of frames, all key frames (2fps), and all available frames (10fps).

Another advantage of the proposed approach over the other methods is that it is able to estimate rotation as well as position. When adding a rotation error component to the definition of a true positive (i.e., within the matching distance threshold and within the rotation angular threshold of 20°), there is only a slight lowering of performance. In essence, the proposed method makes a stronger prediction (5D) even when compared to other methods that only output 3D locations.

3.4 Conclusion

In this chapter, we covered a novel end-to-end method for 5D pose estimation, tracking, and localization of spatially-compact static objects from a single camera of a self-driving car. Specifically, the proposed end-to-end approach is mainly composed of two networks: 1) a pose regression network for estimating 5D poses of static objects from geo-located RGB inputs, shown to outperform contemporary methods, and 2) objects matching network for matching objects between pairs of video frames combining multi-resolution appearance features and geometric features from the proposed pose regression network. Jointly optimizing the pose regression and object matching models improves 5D pose estimation, tracking, and geo-localization simultaneously. Adding geometric cues explicitly for matching static objects has been shown to increase the matching accuracy over appearance alone. Moreover, static objects geo-localization data set was created, which can help to accelerate static objects geo-localization domain and provide a common benchmark for the evaluation of different approaches and, to the best of our knowledge, is the first publicly available data set for this application.

Chapter 4

Multi-Object Tracking

Visual Multi-Object Tracking has made significant progress in recent years, motivated in part by high-profile mobile robotics and autonomous driving applications. Continued improvements in the accuracy and efficiency of CNN-based object detectors have driven the dominance of the “tracking by detection" paradigm. There are many ways to associate detections across frames, but those featuring learned associations are interesting because they have the promise of addressing edge-cases where modeling and heuristics-based approaches fail. Even with learned associations, the two-stage approach can lead to sub-optimal results in terms of accuracy and efficiency. A recent trend of jointly learning detection and tracking tasks in a single neural network has led to increases in performance on tracking benchmarks and related applications. However, existing end-to-end methods that combine appearance and motion cues can be complex and slow (see §2.3, Related Work).

We posit that a learned object matching module can be added to most contemporary CNN-based object detectors to yield a high performing multi-object tracker, and further, that by jointly training the detection and tracking (association) modules, both modules adapt to each other and together perform better. Using the same backbone for object detection and inter-frame association increases efficiency and accuracy when compared to methods that use detection as a black-box feeding input to the association logic.

Given the tracking-by-detection paradigm, and inspired by the proposed approach for static objects association (see §3.2), we propose to exploit the representational power of the intermediate feature maps of the object detector (“detector backbone") to extract embeddings to be used in an object matching network that associates objects across frames. The proposed approach is named “Detection Embeddings for Tracking" (DEFT). The detector is jointly trained with the object matching network. During training, errors in object association propagate back through the detection backbone such that the appearance features are optimized for both detection

and matching. DEFT also employs a low-dimensional LSTM module to provide geometric constraints to the object matching network. This work was published in CVPR 2021 Workshop of Autonomous Driving:Perception, Prediction and Planning [121] and the code is publicly available at: https://github.com/MedChaabane/DEFT .

DEFT implemented with a CenterNet [13] backbone achieves state-of-the-art performance on several tracking benchmarks, while being faster than most similarly-scoring alternatives. The speedup is in part due to the fact that in DEFT, object association is a small additional module within the detection network, thus adding only a few milliseconds of latency beyond detection. DEFT performance can be upgraded as stronger and faster detection backbones are developed. DEFT is also easily extensible to monocular 3D tracking. DEFT was applied to the monocular 3D tracking by using the detection form of CenterNet [13]. Specifically, the 3D detection head of CenterNet is used in order to predict object depth, rotation (encoded as an 8-dimensional vector [122]), and 3D extent.

During inference (see Figure 4.1), the embedding extractor head uses features maps and bounding boxes from the detector as input and extracts appearance embeddings for each detected object. The matching head uses the embeddings to compute a similarity between the objects in the current frame and those remembered from previous frames (current tracks). A motion forecasting module (LSTM) prevents matches that lead to physically implausible trajectories. The Hungarian Algorithm is used to make the final online association of objects to tracks. Details for each module are provided below, followed by the training procedure.

4.1 Object Embeddings

The object embeddings used in the matching network are extracted from the detection backbone, as described below. This is labeled as the “embedding extractor" in Figure 4.1 and subsequent text. The embedding extractor constructs representative embeddings from the intermediate feature maps of the detector backbone to help associate (or, “re-identify") objects during tracking. Feature maps at different layers are used to extract appearance from multiple

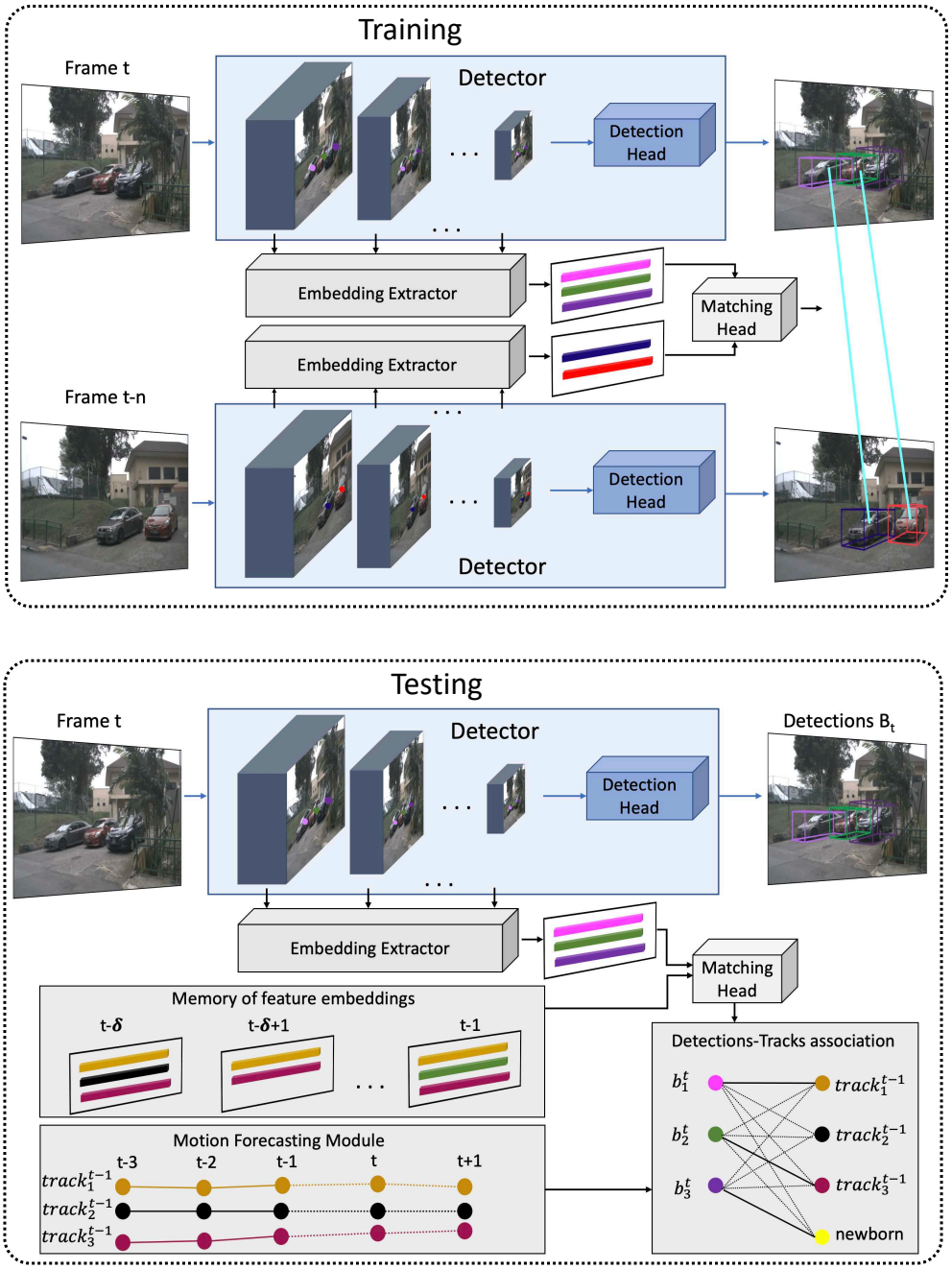

Figure 4.1: DEFT Training and Inference. DEFT adds an embedding extractor and a matching head to an object detection backbone to jointly train appearance features for both detection and association tasks. During inference, the matching head matches the current detections to the embeddings remembered for all active tracks and uses a motion forecasting module to eliminate implausible trajectories.

receptive fields (RFs), which provides additional robustness over single-RF embeddings. DEFT takes a video frame t as input and outputs a set of of N_t bounding boxes $B_t = [b_1^t, b_2^t, ..., b_{N_t}^t]$.

For 2D detection, a bounding box is defined $b_i^t = (x, y, w, h)$, where (x, y) represents the 2D center and (w, h) represents the width and height. For 3D detection, a bounding box is defined as $b_i^t = (x, y, z, w, h, l, q)$, where (x, y, z) represents the 3D center and (w, h, l, q) represents the width, height, length and orientation.

For each detected object, feature embeddings are extracted from the estimated 2D object's center location. For 3D bounding boxes, projection of 3D center location into the image space is used as its estimated 2D center location. If the center of the i^{th} object is at position (x, y) in the input frame of size $W \times H$, then for a feature map of size $W_m \times H_m \times C_m$, C_m-dimensional vector is extracted at position $(\frac{y}{H}H_m, \frac{x}{W}W_m)$ as the feature vector f_i^m for the i^{th} object in feature map m. Features from M feature maps are concatenated to construct the resulting e-dimensional feature embedding $f_i = f_i^1 \cdot f_i^2 \dots f_i^M$ for the i^{th} object.

The dimension of the feature vector f_i^m affects its contribution to the resulting feature embedding. To change the contribution of some feature maps and to control the dimension of the embedding, a single convolution layer is added to some feature maps to increase/decrease the dimension from C_m to C'_m before extracting the feature vectors. In practice, this serves to increase the dimension of features contributed by earlier maps while reducing those from later maps.

4.2 Matching Head

Given N_t bounding boxes in frame t, and N_{t-n} in frame $t - n$, the matching head uses the object embeddings to estimate the affinity scores between all pairs of detections across the two frames. With N_{max} maximum number of allowed objects in each frame, the tensor $E_{t,t-n} \in \mathbb{R}^{N_{max} \times N_{max} \times 2e}$ is constructed such that the feature embedding of each object in frame t is concatenated along depth dimension with all feature embeddings of objects in frame $t - n$ and vice versa. $E_{t,t-n}$ is fed to the a similarity estimator network which is composed of a few (4-6) layers of 1×1 convolutions. Following the same process in § 3.2.3, the affinity matrices $\hat{A}^{bwd}$ and $\hat{A}^{fwd}$ are computed.

4.3 Online Data Association

In DEFT, tracks remember the object embeddings from each observation comprising the track in the last δ frames. The association between a new detection and the existing tracks requires computing the similarity of the new object to the set of observations from each track in memory. To allow for occlusions and missed detections, track memory is maintained for a few seconds so that inactive tracks can be revived if a new observation is strongly associated to the previous observations. Tracks with no observations after N_{age} frames are discarded from memory.

A track T is defined as the set of associated detections from frame $t-\delta$ to $t-1$, noting that tracks may not have a detection for each frame. The size of the track is given as $|\mathrm{T}|$, representing the number of bounding boxes and associated embeddings it comprises. The distance between the i^{th} new detection b_i^t and T_j is defined as:

$$d(b_i^t, \mathrm{T}_j) = \frac{1}{|\mathrm{T}_j|} \sum_{b_k^{t-n} \in \mathrm{T}_j} \frac{\hat{A}_{t,t-n}^{fwd}[k,i] + \hat{A}_{t,t-n}^{bwd}[i,k]}{2} \tag{4.1}$$

The detections-to-tracks association problem is formulated as a bipartite matching problem so that exclusive correspondence is guaranteed. Let $\mathrm{K} = \{\mathrm{T}_j\}$ be the set of current tracks. The detections-to-tracks similarity matrix $D \in \mathbb{R}^{|\mathrm{K}| \times (N_t + |\mathrm{K}|)}$ is constructed by appending the all-pairs detections-to-tracks distance (Eq. (4.1)) of size $|\mathrm{K}| \times N_t$ with a matrix X of size $|\mathrm{K}| \times |\mathrm{K}|$ used to represent when a track is associated with no detections in the current frame. The entries along the diagonal of X are computed as the average non-match score of the detections in the track; the off-diagonal entries are set to $-\infty$. Specifically, D is constructed as follows:

$$D = [S|X] \tag{4.2}$$

$$S[j,i] = d(b_i^t, \mathrm{T}_j) \tag{4.3}$$

$$X[j,k] = \begin{cases} \frac{1}{|\mathrm{T}_j|} \sum_{b_k^{t-n} \in \mathrm{T}_j} \hat{A}_{t,t-n}^{fwd}[k, N_{max}+1] & j = k \\ -\infty, & j \neq k \end{cases} \tag{4.4}$$

Finally, the bipartite matching problem defined by D is solved with the Hungarian algorithm [78]. Only likely associations are included, if the affinity is larger than a specified threshold γ_1. Unmatched detections will start newborn tracks. Tracks that have not been associated for more than a predefined maximum age N_{age} are considered to have left the scene and are deleted from the track set.

4.4 Motion Forecasting

When learning to associate detections across frames using the appearance features from a detection backbone, there is some chance that two objects look similar enough in the embedding space to cause confusion. It is common practice to add additional geometric or temporal constraints to help resolve such ambiguities. This often takes the form of a Kalman Filter (e.g. [4, 123]) or an LSTM module (e.g. [68]).

DEFT uses an LSTM as the motion forecasting module. This module predicts future locations of each track in the next ΔT_{pred} frames given its information in ΔT_{past} past frames. It uses features from past bounding boxes coordinates for each track. For 2D tracking, features for each detection at time t are represented by 8-dimensional vector $(x_t, y_t, w_t, h_t, v_t^x, v_t^y, \frac{\Delta_{w_t}}{\Delta_t}, \frac{\Delta_{h_t}}{\Delta_t})$ containing the 2D center location, height,width, velocity in the x and y directions and change in width and height divided by time difference between consecutive detections . For 3D tracking, features for each detection at time t are represented by 11-dimensional vector $(x_t, y_t, z_t, w_t, h_t, l_t, r_t, v_t^x, v_t^y, v_t^z, v_t^r)$ containing 3D the center location, height,width,length,orientation about the z-axis, velocity in the x, y and z directions and the rotational velocity v_t^r.

The motion forecasting module is used to constrain the associations between frames to those that are physically plausible. It sets affinity scores in Eq. (4.1) to $-\infty$ for detections that are too distant from the track's predicted location. In §4.6.6, an ablation study is provided to show the impact of the LSTM motion forecasting module, and it has shown that it modestly outperforms a Kalman Filter in our application.

In practice, it was also found that a second stage of association using IoU between unmatched detections and predicted locations of unmatched tracks with age less than ΔT_{IoU} frames helps improve the tracking accuracy of DEFT. 2D and 3D IoU are adopted for 2D and 3D tracking, respectively. This helps to account for objects with weak detections (e.g., caused by partial occlusion). Only associations that have sufficiently large IoU scores higher than a specified threshold γ_2 are considered.

4.5 Training

During training, a pair of frames n frames apart is input to DEFT as shown in Figure 4.1. The image pairs are separated by a random number of frames $1 \leq n \leq n_{gap}$ to encourages the network to learn to be robust to temporary occlusions or lost detections. For each training pair, two ground truth matching matrices M^{fwd} and M^{bwd} are created, representing the forward and backward associations, respectively.

To train DEFT for matching estimation, the loss function $\mathcal{L}_{\text{match}}$ is used, which is defined as the average of the two losses $\mathcal{L}_{\text{match}}^{fwd}$ and $\mathcal{L}_{\text{match}}^{bwd}$, where $\mathcal{L}_{\text{match}}^{bwd}$ is the error of matching bounding boxes in frame t to those in frame $t-n$ and $\mathcal{L}_{\text{match}}^{fwd}$ is the error of matching bounding boxes in frame $t-n$ to those in frame t. The expression of the matching loss is given as follows, where "*" represents "fwd" or "bwd" as appropriate:

$$\mathcal{L}_{\text{match}}^{*} = \sum_{i=1}^{N_k} \sum_{j=1}^{N_{max}+1} M^{*}[i,j] \log(\hat{A}^{*}[i,j]), \tag{4.5}$$

$$\mathcal{L}_{\text{match}} = \frac{\mathcal{L}_{\text{match}}^{fwd} + \mathcal{L}_{\text{match}}^{bwd}}{2(N_t + N_{t-n})}, \tag{4.6}$$

Training optimizes the joint affinity and detection losses as defined in Eq. (4.7). For a better optimization of the proposed proposed dual-task network, the strategy proposed in [124] is used for automatic loss balancing the two tasks.

$$\mathcal{L}_{\text{joint}} = \frac{1}{e^{\lambda_1}}(\frac{\mathcal{L}^{t}_{\text{detect}} + \mathcal{L}^{t-n}_{\text{detect}}}{2}) + \frac{1}{e^{\lambda_2}}\mathcal{L}_{\text{match}} + \lambda_1 + \lambda_2 \tag{4.7}$$

where $\mathcal{L}^{t}_{\text{detect}}$ is the detection loss for frame t and λ_1 and λ_2 are the balancing weights to the two tasks. Note that the balancing weights are modeled as learnable parameters.

4.6 Experiments and Results

4.6.1 Datasets and Metrics

The tracking performance of DEFT is evaluated on a set of popular benchmarks: MOT Challenge (MOT16/MOT17) [59], KITTI tracking [11], and the nuScenes Vision Tracking benchmark [3]. The MOT Challenge and KITTI benchmarks are used to assess 2D visual tracking, while nuScenes is used for monocular 3D visual tracking.

MOT16/MOT17. The MOT16 and MOT17 tracking challenges are part of the multi-object tracking benchmark MOT Challenge [59]. They are composed of indoor and outdoor pedestrian tracking sequences. The videos have frame rates between 14 and 30 FPS. Both challenges contain the same seven training sequences and seven test sequences. The proposed tracker is tested using public detections provided by the benchmark protocol, following [105], as well as with private detections output by DEFT. With public detections, the jointly-trained DEFT model is used, but the provided bounding boxes are employed for embedding extraction – all else is the same. With private detections, the bounding boxes output from the model are used.

MOT Challenge benchmarks employ the following metrics: MOTA - multi-object tracking accuracy, MOTP - multi-object tracking precision, IDF1 - identity F1 Score, MT - mostly tracked, ML - mostly lost, FP - false positives, FN - false negatives, IDS - identity switches.

KITTI. The KITTI tracking benchmark is composed of 21 training sequences and 29 test sequences that were collected using cameras mounted on top of a moving vehicle. The videos are recorded at 10 FPS. The performance is evaluated using the “Car" class because it is the only class

with enough examples to allow effective training without external data sources.[1] Public detections are not provided with KITTI. KITTI uses the HOTA metrics [57]: HOTA - higher order tracking accuracy and AssA - association accuracy, in addition to the tracking metrics of the MOT Challenge benchmarks.

nuScenes. nuScenes is a large-scale data set for autonomous driving. nuScenes is composed of 1000 sequences, with 700, 150, 150 sequences for train, validation, and testing, respectively. Sequences were collected in Boston and Singapore, in both day and night, and with different weather conditions. Each sequence length is roughly 20 seconds with a camera frequency of 12 FPS. Each sequence contains data from six cameras forming a full 360° field of view, but box annotations are provided only for key frames (2 FPS). Given the ground truth format, only key frames are used for training and evaluation. The effectively low frame rate of 2FPS makes this data set challenging for visual tracking as the inter-frame motion of objects can be large. DEFT is evaluated on the 7 annotated classes: Car, Truck, Trailer, Pedestrian, Bicycle, Motorcycle, Bus. nuScenes uses tracking metrics aggregated over the curve of operating thresholds [56]. They are as follows: AMOTA - average MOTA, AMOTP - average MOTP, MOTAR - recall-normalized MOTA score.

4.6.2 Implementation Details

DEFT is implemented using PyTorch with an open source license. In all experiments, an Ubuntu server with a TITAN X GPU with 12 GB of memory is used. All hyper-parameters were chosen based on the best MOTA score for 3-fold cross validation on the training sets for 2D tracking and best AMOTA score on the validation set for 3D tracking. The implementation runs at approximately 12.5Hz on all data sets.

2D Tracking. DEFT was trained and evaluated with four object detectors including CenterNet [13], YOLO v3 [38], FPN [35] and Faster R-CNN [33]. Same implementation details and hyper-parameters settings from their official public codes is followed.

[1]This shouldn't be interpreted as a lack of generality, as the results on MOT16/MOT17 and nuScenes show.

Table 4.1: 3-fold cross-validation results of implementing DEFT with different object detector networks on KITTI.

DEFT Variant	MOTA↑	FP ↓	FN ↓	IDS ↓
DEFT + YOLO v3 [38]	85.6	6.5%	6.8%	1.1%
DEFT + FPN [35]	87.0	6.0%	6.3%	0.7%
DEFT + Faster R-CNN [33]	87.7	5.8%	**5.8**%	0.7%
DEFT + CenterNet [13]	**88.1**	**5.7**%	5.9%	**0.3**%

CenterNet is used with the modified DLA-34 [125] backbone, and feature embeddings are extracted from all 13 feature maps of the modified DLA-34 backbone. YOLO v3 is used with the Darknet-53 backbone, and feature embeddings are extracted from 12 feature maps which are the output of layers 4, 10, 14, 19, 23, 27, 31, 35, 39, 44, 48, and 52. For FPN and Faster R-CNN, ResNet101 [119] backbone is used and feature embeddings are extracted from 11 features maps which are the output of layers 7, 10, 16, 22, 34,46, 58,70, 82, 91, and 100.

All object detectors were pre-trained on the COCO dataset [36]. The data is augmented with random cropping, flipping, scaling, and photometric distortions. Table 4.1 shows the relative performance from 3-fold cross-validation on KITTI. This evaluation is also performed using MOT17, which yielded the same ranking, not shown here for brevity. Since the DEFT+CenterNet variant was the strongest performing, it is used as the detection backbone for the remaining results in this chapter.

While DEFT achieves the best performance with CenterNet, these results demonstrate that DEFT can achieve good tracking performance using different detector backbones. Interestingly, Faster R-CNN and CenterNet have similar detection performance on KITTI, however, the association performance is better with CenterNet (fewer ID switches). This might be due to the nature of CenterNet being an anchor-free detector and matches well to DEFT's design of using feature embeddings from the object center locations – allowing the features at center locations to benefit from the supervised signals of both tasks.

When training DEFT with CenterNet for the various experiments in this chapter, DEFT is trained for 80 epochs with a starting learning rate of e^{-4} using the Adam [126] optimizer and batch size 8. The learning rate is reduced by 5 at epochs 30, 60, and 70.

The nuScenes evaluation was performed on the full 360° panorama and not with each camera separately. Following [7], outputs from all cameras are fused naively without any additional post-processing for handling duplicate detections between cameras or for associating objects across cameras. This ensures a fair comparison between all monocular 3D trackers. Additional analyses are performed on the validation set, which allows observing how performance varies when controlling for certain variables, including the amount of occlusion in the track and a measure of inter-frame displacement.

4.6.3 Parameter Settings

MOT16/MOT17. For MOT16 and MOT17, frames are resized to 960×544. The embedding extractor head outputs a feature embedding of $e = 416$ features. The hyper-parameters $N_{max}, n_{gap}, \delta, \Delta T_{\text{IoU}}, \gamma_1, \gamma_2, N_{age}, c$ were set to 100, 60, 50, 5, 0.1, 0.4, 50, 10 respectively. For the motion forecasting module, $\Delta T_{\text{past}}, \Delta T_{\text{pred}}$ were set to 15 and 10.

KITTI. For KITTI, the embedding extractor head outputs a feature embedding of $e = 672$ features. The hyper-parameters $N_{max}, n_{gap}, \delta, \Delta T_{\text{IoU}}, \gamma_1, \gamma_2, N_{age}, c$ were set to 100, 30, 25, 3, 0.1, 0.6, 30, 10 respectively. For the motion forecasting module, $\Delta T_{\text{past}}, \Delta T_{\text{pred}}$ were set to 10 and 5.

nuScenes For 3D monocular tracking in nuScenes, DEFT was trained and evaluated with CenterNet as 3D object detector backbone. The embedding extractor head outputs a feature embedding of $e = 704$ features. The frames are resized to 800×448. The hyper-parameters $N_{max}, n_{gap}, \delta, \Delta T_{\text{IoU}}, \gamma_1, \gamma_2, N_{age}, c$ were set to 100, 6, 5, 1, 0.1, 0.2, 6, 10 respectively. For motion forecasting module, $\Delta T_{\text{past}}, \Delta T_{\text{pred}}$ were set to 10 and 4.

4.6.4 Comparative Evaluation

The proposed approach is compared against other online tracking methods using the protocol appropriate for each benchmark. The common practice of comparing against

Table 4.2: MOT16. Results of the proposed approach with using public (provided) and private detections. Performance comparison with published online methods on the leaderboard for MOT16. JDE is not present on the public leaderboard, results are from their paper.

Method	MOTA↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	IDS ↓
Tracktor17 [105]	54.4	78.2	52.5	19.0	36.9	682
DeepMOT-Tracktor [107]	54.8	77.5	53.4	19.1	37.0	645
Tracktor v2 [105]	56.2	**79.2**	54.9	20.7	35.8	617
GSM Tracktor [106]	57.0	78.1	58.2	22.0	34.5	**475**
Ours (Public)	**61.7**	78.3	**60.2**	**27.0**	**31.8**	768
JDE [110] (Private)	64.4	-	55.8	35.4	20.0	1544
Ours (Private)	68.03	78.71	66.39	33.06	22.92	925

Table 4.3: MOT17. Results of the proposed approach with using public detections (provided bounding boxes) and private detections (those from the network). Compared methods are the published online methods from the MOT17 leaderboard, except CenterTrack where the results are from their paper.

Method	MOTA↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	IDS ↓
Tracktor17 [105]	53.5	78.0	52.3	19.5	36.6	2072
DeepMOT-Tracktor [107]	53.7	77.2	53.8	19.4	36.6	1947
Tracktor v2 [105]	56.3	**78.8**	55.1	21.1	35.3	1987
GSM Tracktor [106]	56.4	77.9	57.8	22.2	35.3	**1485**
CenterTrack [7]	**61.5**	–	59.6	**26.4**	**31.9**	2583
Ours (Public)	60.4	78.1	**59.7**	26.3	32.1	2581
CenterTrack [7] (Private)	67.8	–	64.7	34.6	24.6	3039
Ours (Private)	66.6	78.83	65.42	31.7	24.5	2823

published/peer-reviewed methods listed on the leaderboard is followed. Tracking results for MOT (Table 4.2 and Table 4.3), KITTI (Table 4.4), and nuScenes (Table 4.5) benchmarks are computed by host test servers with hidden labels on the test set.

DEFT is able to achieve a new state-of-the-art in terms of MOTA, IDF1, MT, and ML scores in MOT16 benchmark among all trackers on public detections. In terms of the overall tracking accuracy, DEFT significantly outperforms the best performing trackers GSM Tracktor [106] and Tracktor v2 [105] by 4.7% and 5.5% respectively. DEFT also improves IDF1 score by 2.0%, which demonstrates its effectiveness in identity preservation.

Both CenterTrack and DEFT use a CenterNet detection backbone. We see in Table 4.3 and Table 4.4 that both are top-scoring among published online trackers on the leaderboards. This

Table 4.4: KITTI car tracking. Performance comparison with published online entries on the leaderboard.

Method	HOTA↑	MOTA↑	AssA↑	MOTP ↑	MT ↑	ML ↓	IDS ↓
mmMOT [127]	62.05	84.77	54.02	85.21	73.23	2.77	284
MASS [128]	68.25	85.04	64.46	85.53	74.31	2.77	301
TuSimple [129]	71.55	86.62	71.11	83.97	72.46	6.77	293
SMAT [130]	71.88	84.27	72.13	**86.09**	63.08	5.38	341
CenterTrack [7]	73.02	**88.83**	71.20	85.84	82.31	2.31	**116**
mono3DT [122]	73.16	84.52	**74.18**	85.64	73.38	2.77	377
Ours	**74.23**	88.38	73.79	84.55	**84.77**	**1.85**	343

Table 4.5: nuScenes Vision Tracking. Performance comparison with published monocular 3D tracking entries on the leaderboard.

Method	AMOTA	AMOTP	MOTAR	MOTA
Mapillary [131]+ AB3D [56]	1.8	1.8	9.1	2.0
PointPillars [132]+ AB3D [56]	2.9	1.7	24.3	4.5
CenterTrack [7]	4.6	1.5	23.1	4.3
Ours	**17.7**	1.5	**48.4**	**15.6**

shows the power of the CenterNet detection backbone and provides support that jointly optimized detection and tracking methods can outperform those that have detection as a distinct step. DEFT achieves the highest HOTA score and best trajectory coverage on KITTI, providing evidence that DEFT maintains longer tracks better, possibly due to remembering embeddings for several observations in a track.

We observe a big performance advantage with DEFT on nuScenes (Table 4.5), which is attributed in large part due to differences in how well DEFT tolerates long occlusions and large inter-frame displacements of tracked objects. This hypothesis is explored in §4.6.5. Compared with CenterTrack, DEFT achieves a gain of 13.1 percentage points in AMOTA, 25.3 in MOTAR, and 11.3 in MOTA. As others have noted [2, 3, 133], nuScenes is significantly more difficult and closer to real-world scenarios than KITTI.

Table 4.6 provides a per-class breakdown of performance when evaluating tracking performance on the nuScenes validation data, front-camera only imagery. The Trailer class is particularly difficult when using private (vision-based) detections. When using the provided

Table 4.6: nuScenes 3D monocular Tracking results in term of AMOTA on the validation set. We present the results of our approach with private detections (those from our network) and public detections (from the lidar-based MEGVII [2])

	CenterTrack [7]	Ours	Ours (Public Det)
Bicycle	16.7	20.6	27.3
Bus	39.7	46.4	66.1
Car	49.6	54.6	71.9
Motorcycle	23.8	28.4	50.5
Pedestrian	31.4	38.9	69.5
Trailer	0.0	0.3	47.2
Truck	15.1	20.4	48.0
Overall	25.2	30.6	54.3

lidar-based detections, tracking performance becomes more consistent with the other classes. This points out that some classes may not have enough training samples to train a robust visual detector, lacking the lidar signal.

4.6.5 Performance Analysis

As previously stated, DEFT and CenterTrack perform similarly on 2D tracking benchmarks of MOT17 and KITTI, but DEFT outscores CenterTrack and all other methods on the nuScenes vision tracking leaderboard by a sizable margin. In this section, we investigate what factors may explain the performance difference.

Our intuition is that the major performance gains on KITTI and MOT benchmarks are driven by improved detectors. For MOT and KITTI, the tracking/association logic can be weak and still produce top numbers. Others in the community seem to agree. The authors of Tracktor promote “tracking without bells and whistles" [105], while the creators of CenterTrack state that it “...trades the ability to reconnect long-range tracks for simplicity, speed, and high accuracy in the local regime...this trade-off is well worth it." [7] The first row in Table 4.7 of the ablation study (see §4.6.6) provides additional support for this point of view. We observe that a simple baseline, using nothing other than a motion model and IOU associations, when applied to CenterNet detections, yields a MOTA score of 86.7 on KITTI (validation) and 63.5 on MOT17 (validation). While one

cannot compare validation scores directly to test results, this is suggestive that many of the top leaderboard methods are only marginally better than a naive baseline coupled to a SOTA detector.

However, is tracking without bells and whistles sufficient for harder tasks? We divided the nuScenes validation data into partitions based on two factors, an occlusion score and a displacement score. The occlusion score for a track is the number of frames for which the object was temporarily occluded. We sum this over all tracks to score a video. The displacement score for a track is the mean of the 2D center displacements in consecutive frames. We use the mean of the top ten tracks as the video's score. Scores are linearly rescaled to the range $[0, 1]$. The distribution of scores has led to dividing the occlusion factor into easy and hard categories (below/above the median, respectively) and the displacement factor into approximately equal-sized easy/moderate/hard partitions. We also looked at a combined difficulty factor, which is simply the maximum of the two normalized difficulty scores. Figure 4.2 shows the scores distribution.

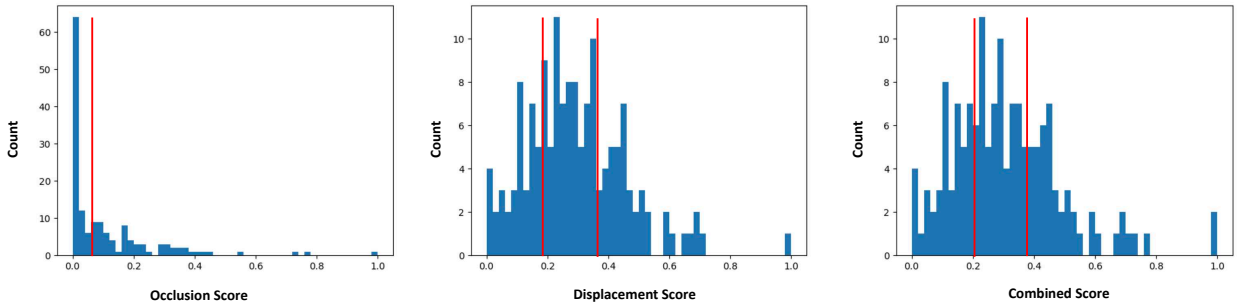

Figure 4.2: Occlusion, Displacement, and Combined scores distribution. Red lines represent thresholds used for the split. The occlusion factor is divided into 76 easy and 74 hard videos (below/above the median respectively). The displacement factor is divided based on two thresholds of half standard deviation from the median score to obtain 44 easy, 55 moderate and 51 hard videos. Similarly, the combined score is divided into 43 easy,60 moderate and 47 hard videos.

Figure 4.3 shows that the difference in performance between DEFT and CenterTrack is marginal on easier videos with respect to the displacement factor or the overall difficulty factor. CenterTrack outperforms DEFT on the easiest videos by less than one percentage point, which is

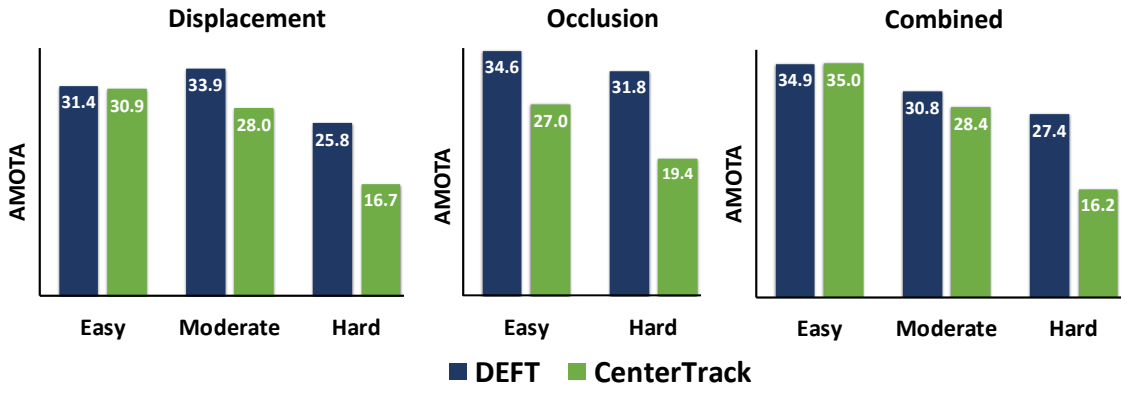

Figure 4.3: DEFT compared with CenterTrack on nuScenes validation front camera videos, according to difficulty factors of occlusion and inter-frame displacement.

consistent with the difference in performance we observe on KITTI and MOT benchmarks. However, when looking at moderate and hard videos, we observe that DEFT provides superior performance. For occlusions, CenterTrack drops 7.6 percentage points from easy to hard, whereas DEFT drops only 2.8, indicating that CenterTrack is more sensitive to occlusions than DEFT. Figure 4.4 provides example frames from videos that feature occlusions and large displacements, where DEFT is more robust.

4.6.6 Ablation Studies

Table 4.7: Ablation study of DEFT on MOT17, KITTI and nuScenes datasets. Results are obtained with 3-fold cross-validation on the training sets for MOT17 and KITTI, for nuScenes the results are on the validation set.

			MOT17				KITTI				NuScenes	
Feature Embeddings	Motion Model	2D/3D IOU	MOTA↑	MT ↑	ML ↓	IDS ↓	MOTA↑	MT ↑	ML ↓	IDS ↓	AMOTA↑	MOTA↑
None	LSTM	✓	63.5	19.6%	38.1%	2.8%	86.7	51.3%	27.9%	1.7%	4.2	5.0
Single-Scale			64.0	28.8%	28.5%	2.3%	87.4	81.46%	3.15%	1.0%	17.1	15.0
Multi-Scale			64.4	29.8%	27.5%	1.9%	87.7	82.55%	2.71%	0.7%	18.4	16.2
Multi-Scale	Kalman		65.2	30.0%	27.3%	1.1%	87.8	82.60%	2.66%	0.6%	18.9	16.4
Multi-Scale	LSTM		65.2	30.0%	27.3%	1.1%	88.0	82.81%	2.57%	0.4%	20.0	17.2
Multi-Scale	LSTM	✓	**65.4**	**30.3%**	**27.0%**	**0.9%**	**88.1**	**83.05%**	**2.36%**	**0.3%**	**20.9**	**17.8**

Figure 4.4: Qualitative results comparison between DEFT and CenterTrack [7] on nuScenes. Each pair of rows shows the results comparison for one sequence. The color of the boxes represents the identity of the tracks. Red arrows point at tracking errors (identity switches). Notice that DEFT is more robust to occlusions and large inter-frame displacements.

In Table 4.7, results of an ablation study on MOT17, KITTI, and nuScenes benchmarks are presented to investigate the importance of various aspects of DEFT. The first row of the table shows baseline performance for tracking when using CenterNet detections followed by a simple non-learned motion+IOU association step. We observe that the baseline tracker performs

reasonably well in MOT17 and KITTI validation data but fails to perform well on the more challenging nuScenes data.

The next two rows compare the benefit of using feature embeddings extracted from different resolutions in the detector backbone versus using only the features from the final feature map. The dimensionality is controlled so that the embeddings are the same size for both the single-scale and multi-scale variants. Using multi-scale embeddings leads to modest improvements across the board.

The table also shows the effect of the motion forecasting module – having one improves results modestly on MOT17 and KITTI, with a stronger effect on nuScenes. An alternative to the learned LSTM motion model would be to use a standard Kalman filter. The performance of the LSTM over the Kalman filter is more pronounced in nuScenes. Finally, there are a few situations in which having an IOU-based association layer as a second-stage to object matching can provide a small performance gain. This is shown in the final row of the table.

On MOT and KITTI, the naive baseline performs well enough that additional gains from DEFT result in a cumulative benefit of only a couple percentage points across most metrics, with the exception of MT and ML (mostly tracked, mostly lost) scores. There we see a big jump in performance from the baseline to multi-scale DEFT. This suggests that learned appearance-based detection-to-track associations help maintain longer tracks. On nuScenes, the value of DEFT is obvious. The gains in AMOTA from baseline to multi-scale DEFT is over 14 percentage points, with an additional 1.5 points gained by adding the motion forecasting LSTM module. This confirms that the learned matching-based association step is critical to overall performance and that the motion model is a helpful addition, but relatively minor in terms of the overall performance.

To show the benefit of joint training, DEFT is compared with joint and separate training strategies. We can see from Table 4.8 that when jointly training detection and tracking tasks, tracking performance is improved on both KITTI and nuScenes datasets without hurting the detection performance.

Table 4.8: Tracking and detection results of implementing DEFT with two training strategies (jointly vs separately optimized) on KITTI and nuScenes. Results are obtained with 3-fold cross-validation for KITTI where detection is evaluated with 2D bounding box AP for three different difficulty levels: easy (AP_E), moderate (AP_M) and hard (AP_H). Results are obtained on the validation set for nuScenes where detection is evaluated with mean Average Precision (mAP) over all 7 classes.

	KITTI					nuScenes		
Method	MOTA	IDs	AP_E	AP_M	AP_H	AMOTA	MOTA	mAP
DEFT (separate training)	87.9	0.5%	92.8	83.6	**74.3**	20.1	17.1	24.5
DEFT (joint training)	**88.1**	**0.3**%	92.8	**83.8**	74.2	**20.9**	**17.8**	24.5

4.7 Conclusion

Most state-of-the-art trackers on popular public benchmarks follow the tracking-by-detection paradigm, with substantial boosts in performance attributable in large part to improved object detectors. This has allowed top-scoring algorithms to use limited matching strategies while achieving high tracking performance and efficiency. The concept of tracking in the "local regime," that is, constraining the association logic to relatively short temporal and spatial extents, has been shown to be effective on at least two popular 2D tracking benchmarks (MOT, KITTI). However, not all tracking challenges are ones where the assumption holds true that the local regime dominates performance. In self-driving car applications, objects tracked in side-mounted cameras experience large inter-frame displacements, and occlusions lasting a few seconds are not uncommon. Additionally, there are use-cases for tracking with lower frame-rate videos in bandwidth constrained domains.

We have shown that detection embeddings used with learned similarity scores provide an effective signal for tracking objects, and are more robust to occlusions and high inter-frame displacements. On KITTI and MOT tracking benchmarks, DEFT is comparable in both accuracy and speed to leading methods. On the more challenging nuScenes visual tracking benchmark, tracking performance more than doubles compared to the previous state of the art, CenterTrack (3.8x on AMOTA, 2.1x on MOTAR). Further, DEFT and CenterTrack perform near parity when occlusions and inter-frame displacements are low. However, when either factor becomes more challenging, DEFT performs better. Importantly, these are not corner cases – the moderate and

hard difficulty samples represent the majority in nuScenes, not the minority. DEFT's significant improvement in these cases is of considerable practical significance.

Chapter 5

Multi-Object Tracking and Sequence Modeling

5.1 Motivation

In Chapter 4, a joint detection and tracking model, DEFT, was presented. DEFT relies on an appearance-based object matching network jointly learned with an underlying object detection network. DEFT has shown to have significant advantages in robustness, compared to current end-to-end methods, when applied to challenging tracking data. We have shown that detection embeddings used with learned similarity scores provide an effective signal for tracking objects and are robust to occlusions and high inter-frame displacement. On KITTI and MOT tracking benchmarks, DEFT is comparable in both accuracy and speed to leading methods. On the more challenging nuScenes visual tracking benchmark, DEFT raises the bar, more than doubling the performance of the previous top method. However, despite the success of DEFT on existing tracking benchmarks, it fails to address some aspects that are important for the effectiveness of autonomous driving systems. These aspects can be summarized as follows:

1. **Sequence Modeling Suitability:** The multi-object tracker outputs the objects' trajectories that are essential for many other sequence modeling tasks. Modern perception systems [134, 135] often perform multi-object tracking and sequence modeling separately in a cascaded order, where tracking is performed first to obtain trajectories in the past, followed by sequence modeling to estimate state of future position. However, this cascaded pipeline with separately trained modules can lead to sub-optimal performance, as information is not shared during training. Since many sequence modeling tasks are dependent upon tracking, it would be beneficial to optimize them jointly. Also, it is not convenient for autonomous driving systems to have separate models for multi-object tracking and the subsequent sequence modeling tasks, as this will result in making compromises in each module in order to meet the computing budget. Thus, for autonomous

driving applications, it is not just the accuracy of the tracker that is important but also the design that makes it suitable for the subsequent sequence modeling tasks. Just as DEFT jointly learns detection and tracking, we could also jointly learn tracking and sequence modeling with the benefit of using common backbone for all the tasks.

2. **Interaction and Dependencies between objects:** DEFT uses a feature extractor independently for each object without accounting for feature interaction or dependencies between objects, something that may lead to sub-optimal discriminative feature learning. Taking into account the interaction and dependencies between objects is beneficial for the tracking and for sequence modeling tasks that follow the tracking process, such as the velocity estimation task, where each object's velocity depends on its surrounding objects. For example, drivers control the vehicle velocity to keep a safe distance to the vehicle ahead. DEFT uses only features from the current detection and the observations belonging to the existing tracklet when estimating the similarity score between them without considering other detections and existing tracklets. This can be insufficient in highly crowded scenes where multiple similar-looking objects move close to each other in the scene. In such cases, the association approach needs to account for the features of other adjacent objects to help address the ambiguities. Thus, we propose extending the detection/tracklet similarity scoring process to consider all the tracked objects and all other new detections, not only the objects of concern.

3. **Effective use of Temporal Information:** DEFT uses temporal information only for the motion forecasting module to constrain the associations between frames to those that are physically plausible, but it ignores such information for the similarity estimation process. Some recent papers [67, 68] have highlighted the importance of aggregating the temporal information and combining it with the context information. They have shown that utilizing higher-order information along with the affinities between pairs of detections helps improve the tracker's performance. Effective use of the temporal information in DEFT can be helpful for the tracking and the following sequence modeling tasks.

5.2 Attention-based DEFT Network

In this chapter, we propose an end-to-end model to solve the tasks of detection, tracking, and sequence modeling from raw sensor data. The proposed model extends the original DEFT model to address the aspects mentioned above. We refer the reader to Figure 5.1 for the overall model architecture, called Attention-based DEFT. The proposed model uses attention to compute tracklet embeddings that 1) account for object interactions and 2) capture the context and temporal information of the tracklet's past observations. The tracklet representation is used in tracking and any subsequent sequence modeling task. Importantly, all detection, tracking, and sequence modeling modules share computation as there is a single backbone network, and the whole model can be trained end-to-end.

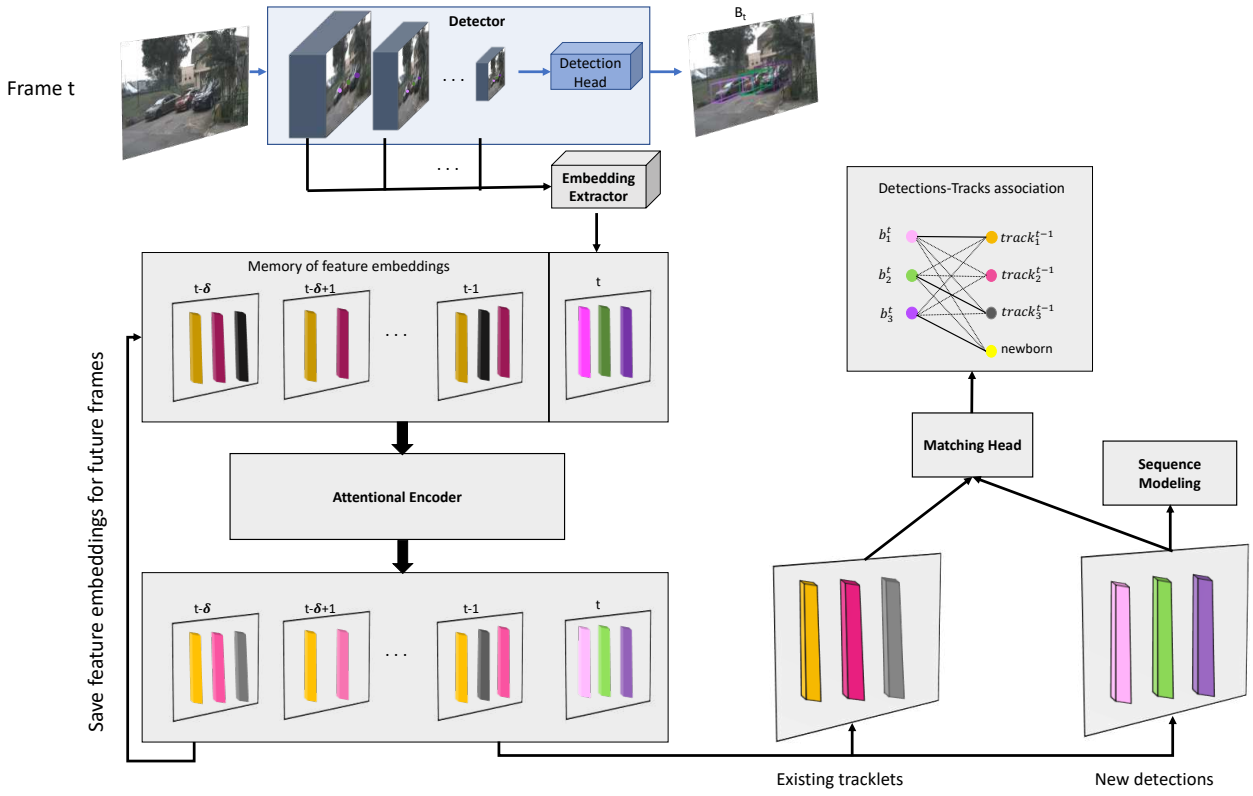

Figure 5.1: Attention-based DEFT. Attention-based DEFT adds an attentional encoder module inspired by the transformer networks. Object embeddings from the last $(\delta + 1)$ frames are processed by the attentional encoder module. The processed embeddings of both the existing tracklets, represented by their last detection embeddings, and the new detections are then fed to the matching head, which estimates the similarity scores between them. The processed new detections embeddings are passed to the sequence modeling network.

At a high level, as shown in Figure 5.1, feature embeddings of the new detections of the current frame t are first extracted using the embedding extractor and then passed along with the embeddings of all detections in the memory (from the last δ frames) to the attentional encoder module. The attentional encoder module processes the input embeddings and generates new feature embeddings for all detections. It applies self-attention layers to the detections embeddings, and this way, the model is able to learn the spatio-temporal dependencies between all detections across all δ frames. The processed embeddings of both the existing tracklets, represented by their last detection embedding, and the new detections are then fed to the matching head, which estimates similarity scores between the new detections and the existing tracklets. The matching head follows the same architecture as in the original DEFT network (see §4.2). Finally, the encoded attentional feature embeddings are stored in the memory for future frames. This process is repeated for each new frame.

It is worth noting that in DEFT (see §4.3) and other existing MOT frameworks [7, 56], the similarity scores between new detections and existing tracklets are estimated based on the hard past data associations. This can be suboptimal as it: 1) ignores several data association hypotheses, 2) does not take into account the contextual information from unmatched detections, and 3) neglects possible errors in the past data associations which can propagate to future associations. In contrast, in the formulation of Attention-based DEFT, no information is provided about the hard past data associations made by the model when estimating similarity scores.

5.2.1 Attentional Encoder Module

An overview of the attentional encoder module is illustrated in Figure 5.2. This module is mainly inspired by the encoder network of the original transformer architecture proposed by Vaswani *et al.* [8]. The transformer network was first proposed to solve sequence-to-sequence tasks in natural language processing (NLP), such as word sequence modeling. Specifically, transformer networks discard the sequential nature of language and instead utilize the potent self-attention mechanism to model temporal dependencies, differentially weighing the importance

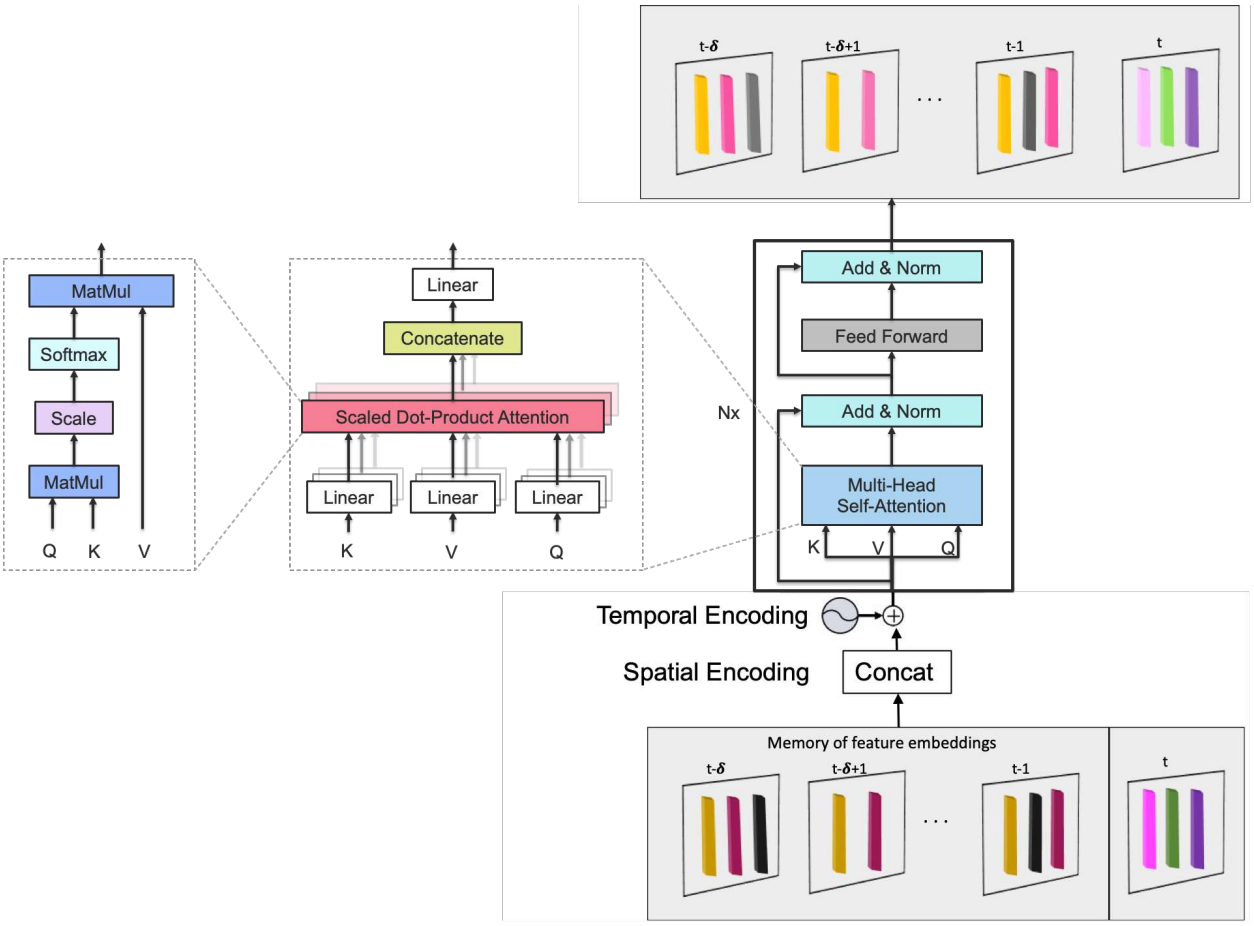

Figure 5.2: Attentional Encoder Module. This module adds spatial and temporal information to the detections embeddings and applies self-attention layers to the obtained embeddings. Parts of this figure are from the original transformers paper [8].

of each part of the input data. The primary gain of the Transformer architecture is that self-attention significantly improves temporal modeling compared to RNNs [8].

In the attentional encoder module, the input is the appearance feature embeddings of the detections obtained in the last $(\delta + 1)$ frames $\{f_i^\eta \mid 1 \leq i \leq N_{max} \text{ and } \eta \in [t - \delta, t]\}$. Each detection embedding is first concatenated with a spatial embedding. For 2D tracking, the spatial embedding is a 2-dimensional vector containing the relative position of the center location of the object in the image space. For 3D tracking, the spatial embedding is a 3-dimensional vector containing the object's 3D location projected to a common reference frame. As shown in precedent work [98, 136], adding the spatial information explicitly to the model is an effective

way for better describing the scene to the model; it enhances its capability to capture the spatial dependencies between different objects. We denote the obtained embeddings by $\{\tilde{f}_i^{\eta} \in \mathbb{R}^{d_{\text{model}}} \mid 1 \leq i \leq N_{max} \text{ and } \eta \in [t-\delta, t]\}$

Self-attention networks are invariant to sequence ordering in their structure, and that is why it requires explicit incorporation of the temporal information. Thus, each obtained embedding is summed with temporal embedding, $T_\eta \in \mathbb{R}^{d_{\text{model}}}$, representing the temporal difference between the current frame t and the actual frame of the detection. In our work, we use learned temporal embedding that is learned during the model training. We have also experimented with the sinusoidal positional encoding proposed by the original formulation of the transformer architecture [8], but we have encountered convergence difficulties and inferior tracking performance.

Finally, the resultant embeddings $\{x_{i,\eta} = \tilde{f}_i^{\eta} + T_\eta \mid 1 \leq i \leq N_{max} \text{ and } \eta \in [t-\delta, t]\}$ are passed through $N_{encoding}$ Transformer-like layers (see Figure 5.2). Each layer is composed of two components: a multi-head self-attention network and a point-wise feed forward network.

In each head of the multi-head self-attention network, an attention function is applied to each embedding $x_{i,\eta} \in \mathbb{R}^{d_{\text{model}}}$. We can define an attention function as mapping a query vector and a set of key-value pairs of vectors to an output. The output is calculated as a weighted sum of the values and the weights are determined for each value based on the compatibility function of the query with the considered key. In our work, each embedding $x_{i,\eta} \in \mathbb{R}^{d_{\text{model}}}$ is updated with scaled dot-product attention, leading to

$$\tilde{x}_{i,\eta} = \sum_j \text{softmax}(\frac{Q_i K_j^T}{\sqrt{d_k}}) V_i \tag{5.1}$$

where $Q_i = W_q x_{i,\eta}, K_i = W_k x_{i,\eta}, V_i = W_v x_{i,\eta}$ are the query, key, value features obtained by applying a linear transformation on a feature embedding $x_{i,\eta}$, where $W_q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_k \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_v \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and j denotes the index of all detections in the last $(\delta+1)$ frames. Thus, each detection embedding will be updated by using a weighted sum of all detections embeddings.

Similarly to the work of [8], we use multi-head attention where embeddings from M attention heads are concatenated to construct $z_{i,\eta} = \tilde{x}^1_{i,\eta} \cdot \tilde{x}^2_{i,\eta} \ldots \tilde{x}^M_{i,\eta}$. In this work, we use $d_k = d_v = \frac{d_{\text{model}}}{M}$.

The point-wise feed forward network comprises fully connected feed-forward layers applied to each position separately and identically. It consists of two linear transformations with a ReLU activation in between. While the linear transformations are the same across different positions, they use different parameters from layer to layer.

Each of these components (multi-head self-attention network and a point-wise feed forward network) is then encapsulated within a residual connection and a layer norm operation. The complete definition of an attentional encoder module layer can be finally written as:

$$\tilde{z}_{i,\eta} = \text{AddNorm}(F(z_{i,\eta}) + x_{i,\eta}) \tag{5.2}$$

where AddNorm indicates the composition of a residual connection and a layer normalization, and $F(.)$ represents the point-wise feed-forward network.

Given the structure above, $N_{encoding}$ encoding layers are stacked sequentially where the output set of layer $l-1$ feeds into the following layer l. This allows creating multi-level encodings encapsulating the relationships between object embeddings, in which higher encoding layers can leverage relationships already identified by previous layers.

When Attention-based DEFT is trained for associating new detections to the existing tracklets (loss function $\mathcal{L}_{\text{match}}$ in §5.2.3), the attentional encoder module computes the objects embeddings that encode the spatio-temporal features that account for object interactions, and thus it learns to output embeddings with higher similarity for detections that belong to the same tracklet and embeddings with lower similarity for those that do not. This improves association with multiple similar-looking objects. Unlike DEFT, where only the appearance features of the existing tracklet and the new detection are considered when estimating the similarity, the proposed attentional encoder module also accounts for other objects in the scene. We show that this extension improves the discriminative power of the appearance features for the detections/tracks association process and improves overall tracking performance in highly crowded scenes (see §5.3.4).

5.2.2 Sequence Modeling

The attentional encoder module addresses aspects that are essential to make DEFT more effective for autonomous driving systems. It tackles aspects that can improve the tracking performance in real-world scenarios, and it also serves better suitability for sequence modeling.

With the self-attention layers formulation, the detection embedding encapsulates data about the tracklet of interest as well as information about other objects encoded with interactions data. This formulation is important for the sequence modeling task, as the new detections embeddings will essentially attend to the embeddings that belong to the same tracklet observations and also to some other detections embeddings that are useful for the sequence modeling task (nearby objects when estimating tracklet's velocity for example). This helps the model to take into account the dependencies and interaction between objects for the sequence modeling task.

As shown in Figure 5.2, the new detections embeddings output of the attentional encoder module are fed to the sequence modeling task network. For each new detection obtained at time t comprising feature embedding $\tilde{z}_{i,\eta}$ output of the last encoding layer, a feed-forward neural network computes a measurement embedding $\tilde{z}^{seq}_{i,\eta}$. This network consists of two fully-connected layers. The two fully connected layers use ReLU and linear activation functions, respectively. To assess the effectiveness of the sequence modeling capability for Attention-based DEFT, we experiment with the velocity estimation task, which is a key requirement for collision avoidance in several scenarios. Thus, we estimate V_x and V_y components of the object's actual velocity in the vehicle coordinate system with the nuScenes [3] dataset. Therefore, $\tilde{z}^{seq}_{i,\eta}$ is a 2-dimensional vector in this case.

5.2.3 Training

In order to generate training data, the ground truth track labels are processed sequentially, where track proposals are generated in an online manner for each frame. Specifically, given N_{Tr} existing tracklets in the previous δ frames and N_t new detections, $N_{Tr} \times N_t$ track-detection pairs are generated in the current frame by considering all possible pairs between the existing tracklets

and the new detections. These $N_{Tr} \times N_t$ track proposals are then used to calculate the matching loss function. These steps are repeated until all frames in each training video are processed, and all videos of the training dataset are treated.

To train Attention-based DEFT for matching estimation, the loss function $\mathcal{L}_{\text{match}}$ (same loss used with DEFT) is used, which is defined as the average of the two losses $\mathcal{L}_{\text{match}}^{fwd}$ and $\mathcal{L}_{\text{match}}^{bwd}$, where $\mathcal{L}_{\text{match}}^{bwd}$ is the error of matching new detections in frame t to the existing tracklets present in the last δ frames and $\mathcal{L}_{\text{match}}^{fwd}$ is the error of matching the existing tracklets to the new detections. The expression of the matching loss was given in §4.5, with $N^{bwd} = N_t$ and $N^{fwd} = N_{Tr}$.

Similarly to DEFT, training optimizes the joint affinity and detection losses as defined in Eq. (5.3).

$$\mathcal{L}_{\text{joint}} = \frac{1}{e^{\lambda_1}}\left(\frac{\mathcal{L}_{\text{detect}}^{t} + \mathcal{L}_{\text{detect}}^{t-n}}{2}\right) + \frac{1}{e^{\lambda_2}}\mathcal{L}_{\text{match}} + \lambda_1 + \lambda_2 \tag{5.3}$$

where $\mathcal{L}_{\text{detect}}^{t}$ is the detection loss for frame t and λ_1 and λ_2 are the balancing weights to the two tasks.

Attention-based DEFT is fully differentiable and can be trained end-to-end for detection, tracking, and sequence modeling. When trained for sequence modeling, the model is first trained for few epochs for detection and tracking tasks and then jointly trained with the sequence modeling task by minimizing the joint loss as shown in Eq. (5.4).

$$\mathcal{L}_{\text{total}} = \frac{1}{e^{\lambda_3}}\mathcal{L}_{\text{joint}} + \frac{1}{e^{\lambda_4}}\mathcal{L}_{\text{seq}} + \lambda_3 + \lambda_4, \tag{5.4}$$

$$\mathcal{L}_{\text{seq}} = \frac{1}{N^{bwd}} \sum_{j=1}^{N^{bwd}} \left\| V_j - \tilde{V}_j \right\|_1, \tag{5.5}$$

where $\mathcal{L}_{\text{seq}}$ is the sequence modeling loss or velocity estimation loss in our experiments, and V_j and $\tilde{V}_j$ are the predicted and ground truth velocity vectors.

During the training experiments with the Attention-based DEFT, we realized that the current training strategy affects the robustness of the model to occlusion. This is mainly due to the high

imbalance in the training data where the rate of occluded tracks (occlusion can last for few frames in real scenarios) is very small. To solve this issue, we uniformly augment the data by simulating occlusions: We randomly remove bounding boxes in the tracks to augment the data with missing detections. For each ground truth track, we randomly choose the missing detection rate from a probability between 0.1 and 0.5. After selecting the missing detection rate, we randomly remove bounding boxes in the selected track according to the selected rate. In §5.3.4, we show a comparison of the model performance with and without the "simulated occlusions".

5.3 Experiments and Results

The performance of Attention-based DEFT is evaluated on KITTI tracking [11], and the nuScenes Vision Tracking benchmark [3]. The KITTI benchmark is used to assess 2D visual tracking, while nuScenes is used for monocular 3D visual tracking and velocity estimation.

5.3.1 Implementation Details

Attention-based DEFT is implemented using PyTorch with an open-source license. In all experiments, an Ubuntu server with a TITAN X GPU with 12 GB of memory is used. All hyper-parameters were chosen based on the best MOTA score for 3-fold cross-validation on the training set for 2D tracking and the best AMOTA score on the validation set for 3D tracking. The implementation runs at approximately 12Hz on all data sets.

Same as DEFT, Attention-based DEFT was trained and evaluated with CenterNet detector [13]. CenterNet is used with the modified DLA-34 [125] backbone, and feature embeddings are extracted from all 13 feature maps of the modified DLA-34 backbone.

For the KITTI dataset, Attention-based DEFT is trained for 80 epochs with a starting learning rate of $1.25e^{-4}$ using the Adam [126] optimizer and batch size 8. The learning rate is reduced by 5 at epochs 30, 60, and 70.

For the nuScenes dataset, Attention-based DEFT is first trained for detection and tracking tasks for 40 epochs using the Adam [126] optimizer and batch size 8. Then, it is jointly trained with the

velocity estimation task for 45 epochs. The learning rate is started at $2e^{-4}$ and then reduced by 5 at epochs 30,50, 60, and 70.

5.3.2 Parameter Settings

KITTI. For KITTI, the embedding extractor head outputs a feature embedding of $e = 670$ features. The hyper-parameters $N_{max}, \delta, N_{age}, d_{\text{model}}, d_k, d_v, M, N_{encoding}$ were set to 100, 25, 30, 672, 84,84, 8, 2 respectively.

nuScenes For 3D monocular tracking in nuScenes, the embedding extractor head outputs a feature embedding of $e = 701$ features. The frames are resized to 800×448. The hyper-parameters $N_{max}, \delta, N_{age}, d_{\text{model}}, d_k, d_v, M, N_{encoding}$ were set to 100, 6, 6, 704, 88,88, 8, 4 respectively.

5.3.3 Tracking Comparative Evaluation

Attention-based DEFT is compared against DEFT and other online tracking methods using the protocol appropriate for each benchmark. Tracking results for KITTI (Table 5.1), and nuScenes (Table 5.2) benchmarks are computed by host test servers with hidden labels on the test set.

Table 5.1: KITTI car tracking. Performance comparison with published online entries on the leaderboard.

Method	HOTA↑	MOTA↑	AssA↑	MOTP ↑	MT ↑	ML ↓	IDS ↓
SoDA† [103]	-	84.3	-	85.3	70.3	3.5	408
mmMOT [127]	62.05	84.77	54.02	85.21	73.23	2.77	284
MASS [128]	68.25	85.04	64.46	85.53	74.31	2.77	301
TuSimple [129]	71.55	86.62	71.11	83.97	72.46	6.77	293
SMAT [130]	71.88	84.27	72.13	**86.09**	63.08	5.38	341
TrackMPNN† [137]	72.30	87.33	70.63	84.49	84.46	2.15	481
CenterTrack [7]	73.02	**88.83**	71.20	85.84	82.31	2.31	**116**
mono3DT [122]	73.16	84.52	**74.18**	85.64	73.38	2.77	377
DEFT	**74.23**	88.38	73.79	84.55	**84.77**	**1.85**	343
Attention-based DEFT	74.02	88.61	73.50	84.59	84.0	2.31	344

† denotes methods that account for the objects interactions and dependencies.

Table 5.2: nuScenes Vision Tracking. Performance comparison with published monocular 3D tracking entries on the leaderboard. IDS_A represents the average of IDS over all classes.

Method	AMOTA ↑	AMOTP ↓	MOTAR ↑	MOTA ↑	IDS_A ↓
Mapillary [131]+ AB3D [56]	1.8	1.8	9.1	2.0	-
PointPillars [132]+ AB3D [56]	2.9	1.7	24.3	4.5	-
CenterTrack [7]	4.6	1.5	23.1	4.3	-
DEFT	17.7	1.5	48.4	15.6	985
Attention-based DEFT	**17.8**	1.5	**52.2**	15.6	**930**

On the KITTI dataset, Attention-based DEFT achieves a similar performance to the state-of-the-art. While Attention-based DEFT falls short of DEFT and CenterTrack on a few metrics, it beats them in some others. Attention-based DEFT performs favorably compared to the methods that take into account interaction and dependencies between different objects, such as the transformer-based SoDA and the GNN-based TrackMPNN. Specifically, it achieves a 4.31% higher MOTA, a 13.7% higher MT, a 1.19% lower ML, and a 15.6% lower IDS when compared to SoDA, and a 1.72% higher HOTA, a 2.84% higher AssA, 1.28% higher MOTA, and a 28.4% lower IDS when compared to TrackMPNN.

On the nuScenes dataset, both DEFT and Attention-based DEFT are the two top-scoring methods among published online trackers on the leaderboard. This shows the power of the detection embeddings used with learned similarity scores. The addition of the attentional encoder module to DEFT appears to have a positive effect on AMOTA, MOTAR and IDS_A. It leads to a MOTAR improvement of 7.5% and an IDS_A reduction of 5.5%.

Table 5.3 shows that the results comparison on the nuScenes validation data set are consistent with the results obtained on the test data set. Compared with DEFT, Attention-based DEFT

Table 5.3: nuScenes Vision Tracking. Performance comparison with the state-of-the-art monocular 3D tracking methods on the nuScenes validation data set.

Method	AMOTA ↑	AMOTP ↓	MOTAR ↑	MOTA ↑	IDS_A ↓
CenterTrack [7]	6.8	1.5	-	-	813
DEFT	20.9	1.5	47.9	17.8	812
Attention-based DEFT	**21.3**	1.5	**50.3**	**17.9**	**791**

achieves a gain of 0.4 percentage points in AMOTA and 2.4 in MOTAR. The improvements of Attention-based DEFT over DEFT on nuScenes dataset (both validation and test datasets) are attributed in large part due to differences in how Attention-based DEFT takes into account the interactions and dependencies among the actors, which results in superior performance in highly crowded scenes. This hypothesis is explored in §5.3.4.

5.3.4 Tracking Performance Analysis

As previously stated, Attention-based DEFT reasons about the interaction and dependencies between different objects when associating new detections to existing tracklets, which we hypothesized will improve the discriminative power of the model in much denser scenes, where several similar-looking objects are present close to each other. To validate that, we created a crowdedness score to estimate how crowded a scene is, in order to evaluate the performance of Attention-based DEFT under different crowdedness conditions.

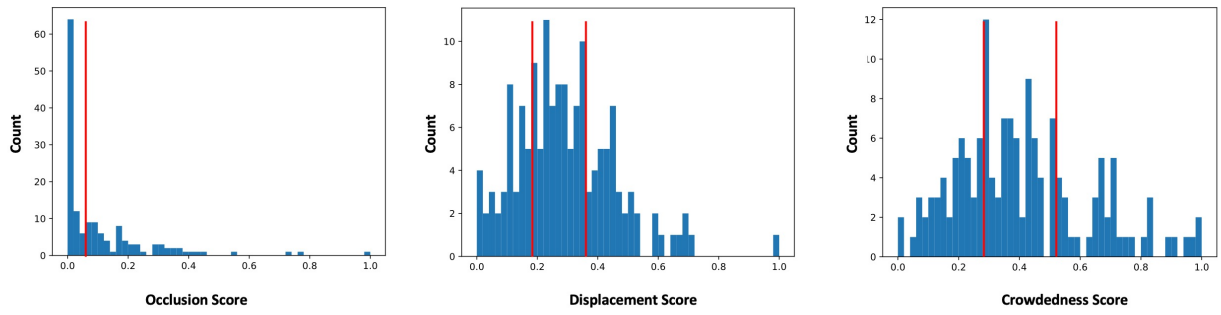

Figure 5.3: Occlusion, displacement, and crowdedness scores distribution. Red lines represent thresholds used for the split. The occlusion factor is divided into 76 easy and 74 hard videos (below/above the median, respectively). The displacement factor is divided based on two thresholds of half standard deviation from the median score to obtain 44 easy, 55 moderate, and 51 hard videos. Similarly, the crowdedness score is divided into 45 easy,65 moderate, and 40 hard videos.

The crowdedness score for a video frame is computed as the total number of objects present from all the classes. We use the mean of the top five video frames as the video's score. Scores are linearly rescaled to the range $[0, 1]$. The distribution of the crowdedness score has led to dividing

the crowdedness factor into easy/moderate/hard partitions. Figure 5.3 shows the scores distribution.

Figure 5.4 shows an interesting variation in the performance of different methods on nuScenes validation front camera videos, according to difficulty factors of crowdedness, occlusion, and inter-frame displacement. DEFT and Attention-based DEFT outperform CenterTrack on both simple and complex scenes. With respect to the displacement and crowdedness factors, the difference in performance between Attention-based DEFT and CenterTrack is marginal on easier videos, and it becomes more prominent as the videos get more challenging. For displacement, for example, CenterTrack drops 14.2 percentage points from easy to hard, whereas Attention-based DEFT drops only 6.8, indicating that Attention-based DEFT is more robust to large inter-frame displacements than CenterTrack.

Interestingly, DEFT outperforms Attention-based DEFT on easier videos with respect to the crowdedness factor; however, when looking at hard videos, we observe that Attention-based DEFT provides superior performance. DEFT drops 6.1 percentage points from easy to hard, whereas Attention-based DEFT drops only 3.8, indicating that DEFT is more sensitive to crowdedness than Attention-based DEFT. It is also worth noting that in terms of the number of identity switches, DEFT IDS_A increases by 137 (from 182 to 319) from easy to hard, while Attention-based DEFT IDS_A increases only by 101 (from 194 to 295) from easy to hard. The results suggest that appearance only based matching may not be discriminative enough in crowded and complex interactive scenes, where multiple objects can share similar appearances in adjacent locations. Taking into account additional factors, mainly the interaction and dependencies between different actors in the scene, and considering all possible interactions among present actors as well as between present and past actors for the detection/tracklet similarity scoring process, has shown to improve the discriminative power of the tracker for each track.

To quantify the effects of data augmentation during training for Attention-based DEFT, we train a model without the "simulated occlusions". The resulting tracking metrics of the two variants

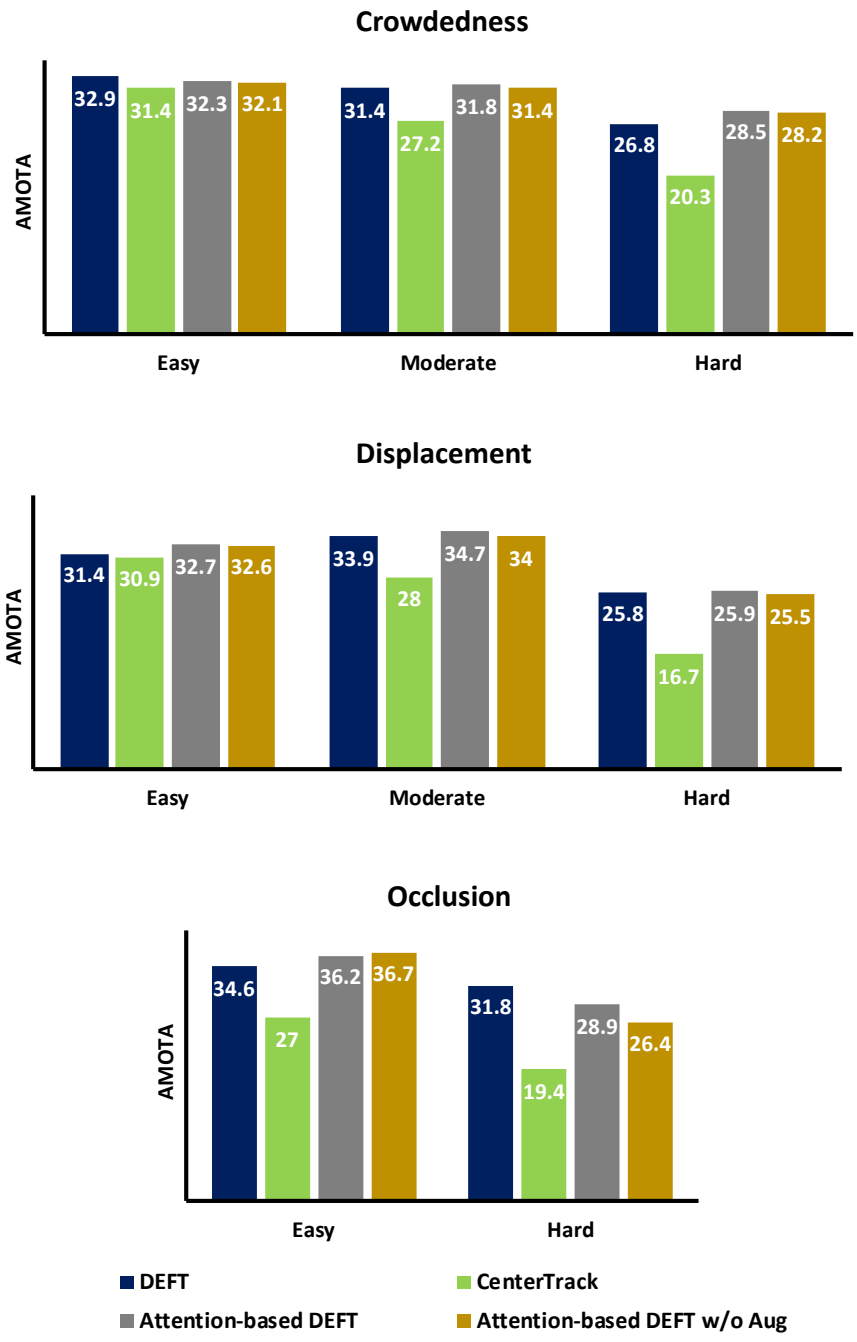

Figure 5.4: Attention-based DEFT compared with DEFT and CenterTrack on nuScenes validation front camera videos, according to difficulty factors of crowdedness, occlusion, and inter-frame displacement. We also present the results of a variant of Attention-based DEFT trained without the "simulated occlusions". Attention-based DEFT is more robust to crowdedness factor than any other compared method, while DEFT is more robust to occlusions.

Table 5.4: Ablation study on the "simulated occlusions" during training on the nuScenes validation data set.

Method	AMOTA ↑	AMOTP ↓	MOTAR ↑	MOTA ↑	IDS_A ↓
Attention-based DEFT w/o Aug	20.7	1.5	49.6	16.9	826
Attention-based DEFT	**21.3**	1.5	**50.3**	**17.9**	**791**

(with and without augmentation) are presented in Table 5.4. The data augmentation improves most metrics across the board. Figure 5.4 indicates that the proposed augmentation technique improves the model performance on hard videos with respect to all factors - and especially for the occlusion factor: AMOTA is improved by 2.5 percentage points. This can be explained by the fact that most of the ground truth tracks in the training data are not occluded, and thus Attention-based DEFT was overfitting to associate objects between consecutive frames, which is why the data augmentation was essential.

It is worth noting that even with the "simulated occlusions", DEFT is more robust to occlusions than Attention-based DEFT: Attention-based DEFT drops 7.3 percentage points from easy to hard, whereas DEFT drops only 2.8. This shows that explicitly using track embeddings memory and considering only the objects of concern with simple learned pairwise detection/tracklet similarity estimation is more beneficial when multiple objects are occluded in the scene. However, as mentioned above, considering all the tracked objects and all other new detection for the detection/tracklet similarity scoring process is more helpful when the crowd density is high.

5.3.5 Tracking Ablation Studies

In Table 5.5 and Table 5.6 we present the results of an ablation study on KITTI and nuScenes benchmarks investigating how 5 architectural variants (V1 - V5) affected Attention-based DEFT's performance.

The first variant (V1) shows baseline performance for tracking by removing the attentional encoder module completely and passing directly the existing tracklets last embeddings and the new detections embeddings to the matching head. We observe that the baseline tracker performs

Table 5.5: Ablation study of tracking performance of Attention-based DEFT on KITTI dataset. Results are obtained with 3-fold cross-validation on the training sets for KITTI.

Model	Feature Interaction	Temporal Information	Spatial Information	Motion Model	KITTI MOTA↑	KITTI MT ↑	KITTI ML ↓	KITTI IDS ↓
V1	None	None	None	None	87.73 ± 0.025	81.37 ± 0.076	3.29 ± 0.035	0.87 ± 0.041
V2	✓	None	None	None	87.98 ± 0.042	82.10 ± 0.230	2.73 ± 0.061	0.52 ± 0.062
V3	✓	✓	None	None	88.21 ± 0.039	82.65 ± 0.068	2.58 ± 0.038	0.29 ± 0.028
V4	✓	✓	✓	None	**88.25 ± 0.036**	**82.85 ± 0.078**	**2.51 ± 0.042**	**0.25 ± 0.024**
V5	✓	✓	✓	✓	88.20 ± 0.054	82.72 ± 0.084	2.61 ± 0.045	0.30 ± 0.029
DEFT					88.10 ± 0.046	83.05 ± 0.108	2.36 ± 0.053	0.30 ± 0.036
CenterTrack					-	-	-	-

Table 5.6: Ablation study of tracking performance of Attention-based DEFT on nuScenes dataset. the results are on the validation set.

Model	Feature Interaction	Temporal Information	Spatial Information	Motion Model	NuScenes AMOTA↑	NuScenes MOTAR↑	NuScenes MOTA↑	NuScenes IDS_A ↓
V1	None	None	None	None	17.3	43.7	15.2	971
V2	✓	None	None	None	19.5	47.5	16.9	858
V3	✓	✓	None	None	21.0	49.7	17.8	804
V4	✓	✓	✓	None	**21.3**	**50.3**	**17.9**	**791**
V5	✓	✓	✓	✓	21.2	49.9	17.8	807
DEFT					20.9	47.9	17.8	812
CenterTrack					6.8	-	-	813

reasonably well in KITTI and nuScenes validation data. With such a simple design, our model has achieved a strong baseline compared with state-of-the-art methods on the nuScenes validation dataset (Table 5.3). It outperforms CenterTrack by 10.5 percentage points in terms of AMOTA.

The second variant (V2) evaluates the effectiveness of considering interaction and dependencies between new detections and existing tracklets. It adds the attentional encoder module to V1, but with removing the spatial information and also with passing as input only the last embedding of each existing tracklet (instead of all detections embeddings in the memory) along with the new detections embeddings. This leads to a modest results improvement on KITTI, with a more substantial effect on nuScenes: a gain of 2.2 percentage points in AMOTA, 3.8 in

MOTAR, and an IDS_A reduction of 11.6% on nuScenes. This shows the benefit of considering interaction and dependencies between past tracks and new detections for objects association.

The third variant (V3) evaluates the effect of the temporal information aggregation on the tracking performance of Attention-based DEFT. Instead of passing only the last embedding of each existing tracklet as input (V2), all detections embeddings in the memory and the new detections embeddings are processed by the attentional encoder module. The variant V3 further improves AMOTA by 1.5 percentage points and MOTAR by 2.2 on nuScenes, which validates that the attentional encoder module effectively exploits the rich spatio-temporal context information to enhance data association. The improvements are more pronounced in nuScenes, which suggests that aggregating temporal information, as well as context information, is more beneficial in the more complex and real-world scenarios.

The fourth variant (V4) shows the modest additional benefit of adding the spatial information (embedding), which helps the model to capture both temporal and spatial dependencies between different actors in the scene.

Interestingly, unlike DEFT, where adding a motion forecasting module (LSTM) to constrain the associations between frames to those that are physically plausible was greatly beneficial to the tracking performance, we can notice a slight drop in performance when adding such component to Attention-based DEFT (variant V5). This suggests that the attentional encoder module is able to learn motion modeling and implicitly associate only physically plausible trajectories.

5.3.6 Sequence Modeling Comparative Evaluation

We evaluate the velocity estimation performance of Attention-based DEFT on nuScenes validation data set and compare it with several baseline methods. We summarize the results in Figure 5.5.

A large number of the objects in nuScenes data set are static objects (zero velocity): 74% for Car, 78% for bicycle, 43% for Bus, 62% for Motorcycle, 84% for Trailer and 76% for Truck . Therefore, we tested Zero-Velocity baseline which predicts zero velocity for all objects. The

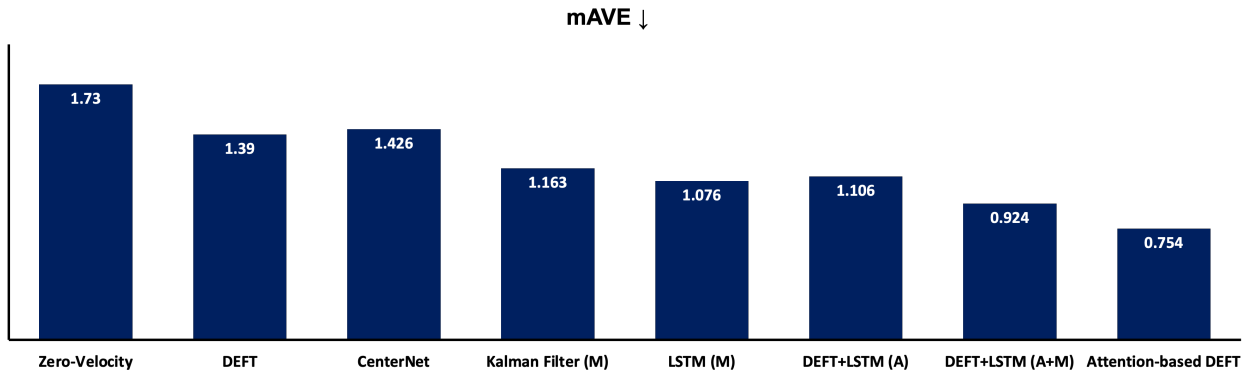

Figure 5.5: Velocity estimation performance comparison on nuScenes validation data set in terms of mean Absolute Velocity Error (mAVE). The mAVE is defined as the L2 norm of the velocity differences in 2D in meters per second (m/s). ↓ indicates lower values are preferred. We compare with Zero-velocity baseline which predicts zero velocity for all objects, single-frame-based baselines (DEFT and CenterNet) and temporal modeling baselines (Kalman Filter, LSTM, DEFT + LSTM). (M) stands for motion features, and (A) stands for appearance features.

Zero-velocity baseline shows higher velocity estimation error than all other methods. DEFT and CenterNet are single-frame-based baselines. Both estimate the velocity of the objects using only static appearance features extracted from single video frames. To adapt DEFT for the single frame velocity estimation task, we pass each object's appearance embedding (output of the feature extractor network) through a feed-forward neural network to estimate the object velocity. We train DEFT jointly for multi-object detection, tracking, and velocity estimation tasks. For CenterNet, we evaluate the performance of the published pretrained model. Details about the CenterNet velocity estimation head can be found in the original paper [13].

The rest of the baseline methods consider temporal relation between consecutive frames. Kalman Filter and LSTM use the trajectories output of Attention-based DEFT and predict the velocity for each predicted tracklet using motion features (location, orientation and velocity). The motion features are the same features used by the motion forecasting module used with DEFT to predict tracks' future locations (see §4.4). We also built a variant of DEFT (DEFT + LSTM) adapted for modeling temporal sequence data by stacking an LSTM on top of the frame-wise appearance features. For each tracklet, The LSTM recursively updates the sequence information stored in its hidden state with each new appearance input. Then, a feed-forward neural network

takes as input the hidden state to produce the estimated velocity. We also tested out a variant (A + M) where motion features (same features as in §4.4) are concatenated with appearance features before going through the LSTM network.

The results show that Kalman Filter and LSTM baselines which use simple motion features, achieve superior performance over single-frame baselines DEFT and CenterNet: mAVE is decreased by more than 16%. The temporal information is not considered in DEFT and CenterNet, and thus their performance is quite limited. This indicates that objects velocities can be inferred from semantic contexts but with limited capability. This is expected since predicting the absolute velocity of an object from a single video frame can be challenging even for a human.

Adding an LSTM on top of the frame-wise appearance features extracted from DEFT (DEFT + LSTM (A)) leads to mAVE drop of 20%. This indicates that though visual appearances (and their context) are important for velocity estimation, it is rather more important to model the temporal structure. We also show that by adding the motion features to DEFT + LSTM model, we achieve an additional 16.4% mAVE relative reduction.

Attention-based DEFT outperforms DEFT + LSTM (A+M) baseline by achieving 18% lower mAVE. The superiority of Attention-based DEFT for velocity estimation over all other methods is quite impressive. It confirms the ability of the attentional encoder module to model the temporal structure with significant variations and complexities effectively.

5.3.7 Sequence Modeling Performance Analysis

Table 5.7 provides a per-class breakdown of performance when evaluating velocity estimation performance on the nuScenes validation data. Attention-based DEFT consistently outperforms all other baselines on most classes (except for Trailer). The most improved classes are Car and Pedestrian, where Attention-based DEFT achieves 21% and 27% less mAVE, respectively, compared to DEFT + LSTM (A+ M). Objects in these two classes represent the majority of the total annotations (56% for Car and 21% for Pedestrian), and multiple objects are usually present simultaneously in the scene, making them tend to interact with each other and change their

motion because of that. Our intuition is that the major performance gains of Attention-based DEFT over DEFT + LSTM (A+ M) come from its capability of learning complicated dependencies and interaction between different objects when estimating the velocities of the objects, and this is what explains the more pronounced performance gap on the Car and Pedestrian classes.

Table 5.7: Per-class performance comparison for velocity estimation (mAVE) on nuScenes validation dataset.

	DEFT	DEFT + LSTM (A)	DEFT + LSTM (A+ M)	Attention-based DEFT
Bicycle	1.02	1.12	0.96	**0.91**
Bus	2.48	1.62	1.51	**1.38**
Car	1.51	1.17	0.95	**0.75**
Motorcycle	2.87	1.80	1.62	**1.59**
Pedestrian	0.84	0.78	0.69	**0.50**
Trailer	0.80	0.75	**0.56**	0.61
Truck	1.47.	1.24	1.03	**0.91**
Overall	1.39	1.10	0.92	**0.75**

We analyze the effect of different difficulty factors on the velocity estimation performance of Attention-based DEFT and DEFT + LSTM (A+ M) in Figure 5.6. With respect to crowdedness, we observe that while Attention-based DEFT achieves lower velocity estimation error than DEFT + LSTM (A+ M) on easy, moderate, and hard videos, the difference in performance is marginal on easier videos and substantial on hard videos. Attention-based DEFT mAVE is increased by 0.37 from easy to hard, whereas Attention-based DEFT mAVE is increased only by 0.11. This validates the effectiveness of Attention-based DEFT for capturing spatio-temporal interaction of the crowd, which helps it to be more robust to handle complexities in heavily crowded scenes.

As shown in Figure 5.6, Attention-based DEFT consistently achieves lower velocity estimation error than DEFT + LSTM (A+ M) on both simple and complex scenes. Despite that DEFT achieves better tracking performance (see Figure 5.4) on hard videos with respect to occlusions, Attention-based DEFT velocity estimation error is lower than DEFT + LSTM (A+ M). This shows the power of the proposed spatio-temporal context information encoding to

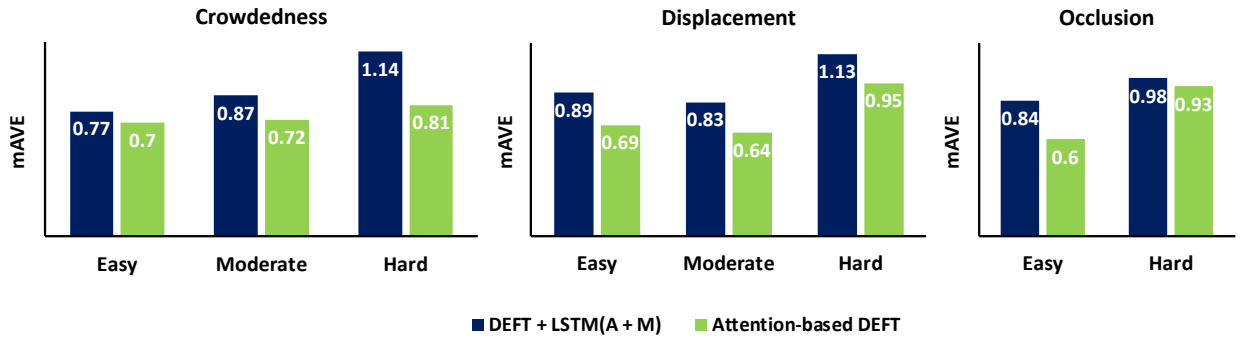

Figure 5.6: Attention-based DEFT compared with DEFT + LSTM (A+ M) on nuScenes validation dataset, according to difficulty factors of crowdedness, occlusion, and inter-frame displacement.

accurately predict objects velocities from inputs with missing observation data (multiple occluded objects), thanks to its attention mechanism, something in which LSTMs have shown to be ineffective [138]. Notably, thanks to the temporal embedding, which time-stamps the input, the attentional encoder module learns to effectively neglect missing observations while being aware of the relative time-stamps of the presented observations.

Furthermore, Attention-based DEFT aggregates information from all detections in the past frames with no information about the past hard data associations. This prevents the past association errors from negatively affecting the current tracklet representation the velocity estimation network uses. This is depicted in hard videos with respect to the occlusion factor. On the other hand, in DEFT + LSTM, the information is gathered from the detections that are matched with each tracklet via hard data associations, which badly impact the velocity estimation when the model's tracking performance is poor, such as in hard videos with respect to crowdedness.

5.3.8 Sequence Modeling Ablation Studies

We validate the effectiveness of our sequence modeling approach by conducting an ablation study on the nuScenes validation set. We present the results of 5 architectural variants (V1 - V5) in Table 5.8. For a fair comparison, all variants are jointly trained for detection, tracking and velocity estimation.

Table 5.8: Ablation study of velocity estimation performance of Attention-based DEFT on nuScenes validation dataset. AEM stands for attentional encoder module.

Model	Temporal Information	Feature Interaction	Spatial Information	Bicycle	Bus	Car	Motorcycle	Pedestrian	Trailer	Truck	Overall
V1	None	None	None	1.08	2.40	1.55	3.10	0.89	0.92	1.59	1.44
V2	LSTM	None	None	1.02	1.83	1.16	2.21	0.81	0.76	1.04	1.09
V3	AEM	None	None	0.95	1.64	1.10	1.80	0.77	0.75	1.05	1.03
V4	AEM	✓	None	**0.89**	1.45	0.83	1.63	0.52	0.61	0.93	0.81
V5	AEM	✓	✓	0.91	**1.38**	**0.75**	**1.59**	**0.50**	**0.61**	**0.91**	**0.75**

The first variant is single frame baseline where the new detections appearance embeddings $\{f_i^t \mid 1 \leq i \leq N_{max}\}$ are passed through a feed-forward network to estimate the objects velocity. The variant V1 achieves overall poor performance, which is similar to other single-frame-based baselines DEFT and CenterNet (Figure 5.5). This validates our conclusions on the limitations of estimating the velocity of moving objects based only on their static visual information.

In order to evaluate the importance of the temporal aggregation for velocity estimation performance of Attention-based DEFT, we built two variants V2 and V3, that take each tracklet's frame-wise appearance embeddings and pass them through a sequence modeling network, which are an LSTM and attentional encoder module (without the spatial information) for V2 and V3, respectively. Each object is modeled separately without any complex interaction consideration. The LSTM sequentially processes the embeddings before predicting the velocity, while the attentional encoder module scans all available embeddings, weighting them according to an attention mechanism. In these two variants, the sequence modeling network is separate from the actual tracking architecture, but all modules are jointly trained. Both variants outperform the V1 by a large gap, indicating the importance of temporal context for velocity estimation. We observe that the attentional encoder module outperforms the LSTM, which suggests that only-attention-based memory mechanisms of the attentional encoder module provide a better temporal modeling ability compared to RNNs.

The variant V3 can model the motion dynamics of each object separately but fails to incorporate interactions and dependencies between different objects. The variant V4 analyzes the

contribution of considering crowd interaction for velocity estimation performance of Attention-based DEFT. It consists of the same architecture proposed in §5.2 (without the spatial information). This further pushes the velocity estimation performance because each object's velocity depends on its surroundings. For example, when a driver of a vehicle would decide on the velocity, he would first predict the future motions of his neighbors and choose a velocity that avoids collision with others in the next short time interval. This explains the main advantage of Attention-based DEFT over DEFT+LSTM, especially when the crowd density is high (Figure 5.6). The large improvements of V4 over V3 are on the Car and Pedestrian classes, which is consistent also with the per class improvements we saw of Attention-based DEFT over DEFT+LSTM in Table 5.7.

The last variant (V5) shows that adding the spatial information (embedding) helps to further decrease the velocity estimation error. Inferring motion from the features extracted from the raw sensor data is important, but also exploiting motion from explicit objects locations in the scene is also helpful to the overall velocity estimation performance.

5.3.9 Additional analysis

Effect of the encoding memory length: We compare Attention-based DEFT using different sizes of the encoding memory scope δ in Figure 5.7. From the results, we can see that as the memory scope increases, the tracking performance of the model improves. This indicates that Attention-based DEFT leverages all available information from context, history, and interactions. Increasing the memory for more than 6 frames (3 seconds) results in small tracking performance improvements while adding more computational running time and memory usage.

Longer memory leads to lower velocity estimation error, which indicates that history is helpful to velocity estimation. We observe that the model performance saturates at a spanning scope of around 4 frames (2 seconds) in the past. We believe that this has to do with the usual time for real-world traffic changes. Therefore, increasing the memory window beyond 2 seconds does not further improve the performance.

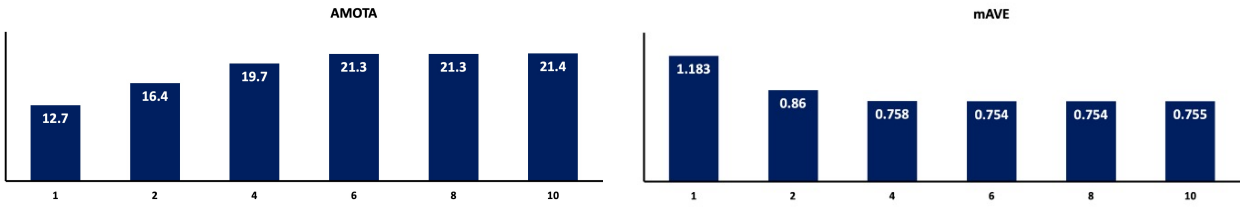

Figure 5.7: Attention-based DEFT's tracking and velocity estimation performance for different sizes of the encoding memory scope δ .

Effect of Joint Training: Attention-based DEFT is compared with joint and separate training strategies. We can see from Table 5.9 that when jointly training detection, tracking, and sequence modeling tasks, tracking performance is improved on both KITTI and nuScenes datasets without hurting the detection and velocity estimation performance.

Table 5.9: Detection, tracking, and velocity estimation results of implementing Attention-based DEFT with two training strategies (jointly vs separately optimized) on KITTI and nuScenes. Results are obtained with 3-fold cross-validation for KITTI where detection is evaluated with 2D bounding box AP for three different difficulty levels: easy (AP_E), moderate (AP_M), and hard (AP_H). Results are obtained on the validation set for nuScenes, where detection is evaluated with mean Average Precision (mAP) over all 7 classes.

	KITTI					nuScenes			
Training Strategy	MOTA	IDs	AP_E	AP_M	AP_H	AMOTA	MOTA	mAP	mAVE
Separate training	88.0	0.4%	**92.8**	83.6	74.3	20.2	17.3	24.5	0.78
Joint training	**88.2**	**0.2**%	92.7	**83.7**	**74.5**	**21.3**	**17.9**	**24.6**	**0.75**

5.4 Conclusion

In this chapter, we have introduced Attention-based DEFT, an end-to-end model to solve the tasks of detection, tracking, and sequence modeling from raw sensor data. Instead of treating each task separately like the traditional cascaded methods, we follow the recent advances of multi-task learning, and we train a model to solve multiple tasks jointly while taking advantage of the shared feature computation. We further improve upon the paradigm with a novel objects representation encoded by the proposed attentional encoder module that 1) jointly reasons about the object

dependencies and interaction with other objects present in the scene and 2) captures the context and temporal information of the tracklet's past observations. We conducted experiments and ablations on two popular benchmarks (KITTI and nuScenes) to validate the effectiveness of Attention-based DEFT. In terms of the tracking performance, the results suggest that the proposed approach performs favorably against or comparable to state-of-the-art methods. We have shown that reasoning about the interactions between the actors in the scene improves the model tracking performance in highly crowded scenes while worsening its performance in hard scenes with respect to occlusions. We have also shown the superiority of Attention-based DEFT for velocity estimation over other baseline methods on simple and complex scenes. This confirms the remarkable ability of the attentional encoder module to model the temporal structure with significant variations and complexities with taking into account the interactions and dependencies among the actors.

Attention-based DEFT makes velocity estimation only by using objects' context, history, and interactions, which might fail in extreme scenarios such as unpredictable sharp turns or sudden stops. Thus, additional information such as map configuration could help solve such issues. Furthermore, Attention-based DEFT was initially designed to adapt to any sequence modeling, and velocity estimation is one possible application. Future work can include its application to other sequence modeling tasks.

Chapter 6

Conclusion and Future Work

6.1 Summary

This dissertation aims to improve the capability of the perception module, an essential module for safe and reliable autonomous driving. Specifically, it focuses on two perception topics: 1) Geo-localization (mapping) of spatially-compact static objects and 2) Multi-target object detection and tracking of moving objects in the scene.

In Chapter 3, we presented a novel system that improves the localization of static objects by jointly optimizing the components of the system via learning. Our system is comprised of networks that perform: 1) 5DoF object pose estimation from a single image, 2) association of objects between pairs of frames, and 3) multi-object tracking to produce the final geo-localization of the static objects within the scene. The proposed approach was evaluated using a publicly available data set, focusing on traffic lights due to data availability. For each component, we compared against contemporary alternatives and showed significantly improved performance. We also showed that the end-to-end system performance is further improved via joint training of the constituent models. The results suggest that the proposed explicit addition of the geometric cues for matching static objects increases the matching accuracy over appearance alone, something that can be considered for future works in this field. Furthermore, a static objects geo-localization data set was created to help accelerate the static objects geo-localization domain and provide a common benchmark for evaluating different approaches.

Chapter 4 described a simple, online, and efficient joint model of detection and tracking named DEFT. DEFT relies on an appearance-based object matching network that is jointly learned with an underlying object detection network. DEFT can be applied effectively to several popular detection backbones. DEFT's design and data association approach yields top-tier performance on KITTI and MOT 2D tracking benchmarks and sets a new standard on the more

challenging nuScenes monocular 3D vision tracking challenge. On the nuScenes benchmark, DEFT performance more than doubles compared to the previous SOTA, CenterTrack (3.8x on AMOTA, 2.1x on MOTAR). Further, partitioning the data using the created difficulty scores helped analyze the difference in performance between DEFT and CenterTrack. DEFT and CenterTrack perform near parity when occlusions and inter-frame displacements are low. However, when either factor becomes more challenging, DEFT performs better. Importantly, in the autonomous driving application, objects tracked, especially in side-mounted cameras, experience large inter-frame displacements, and occlusions lasting a few seconds are not uncommon. These are not corner cases – the moderate and hard difficulty samples represent the majority in nuScenes, not the minority. DEFT's significant improvement in these cases is of considerable practical significance. DEFT is also extensible to new detectors as they arise and adds only modest latency to the underlying detection network.

In chapter 5, an end-to-end model to solve the tasks of detection, tracking, and sequence modeling from raw sensor data, called Attention-based DEFT, was proposed. Attention-based DEFT extends the original DEFT by adding an attentional encoder module that uses attention to compute tracklet embedding that 1) jointly reasons about the tracklet dependencies and interaction with other objects present in the scene and 2) captures the context and temporal information of the tracklet's past observations. We conducted experiments and ablations on KITTI and nuScenes to validate the effectiveness of Attention-based DEFT. In terms of the tracking performance, the results suggest that Attention-based DEFT performs favorably against or comparable to SOTA methods. Reasoning about the interactions between the actors in the scene allowed Attention-based DEFT to improve significantly the model tracking performance in highly crowded scenes. We also showed the superiority of Attention-based DEFT for sequence modeling tasks (velocity estimation) over other baseline methods on both simple and complex scenes. The experimental analysis confirmed the effectiveness of Attention-based DEFT for capturing spatio-temporal interaction of the crowd for velocity estimation task, which helps it to be more robust to handle complexities in densely crowded scenes.

6.2 Findings

The work of this dissertation revealed some interesting findings that we summarize in this section. In Chapter 3, we studied the static objects geo-localization topic for the onboard perception of autonomous vehicles. We have shown that we can benefit from the additional objects attribute of being - static objects - and use geometric cues explicitly when associating the objects across frames. Using the geometric cues alone has shown to achieve a reasonable association accuracy of 83% mAP and when combined with appearance cues they helped to increase the mAP by 3.9 percentage points over appearance alone.

In Chapter 4, We have shown that the major performance gains on some datasets such as KITTI and MOT are driven by improved detectors. Others in the community seem to agree [7, 105]. The first row in Table 4.7 of the ablation study (see §4.6.6) provides additional support for this point of view. We observe that a simple baseline, using nothing other than a motion model and IOU associations, when applied to CenterNet detections, yields a MOTA score of 86.7 on KITTI (validation) and 63.5 on MOT17 (validation). While one cannot compare validation scores directly to test results, this is suggestive that many of the top leaderboard methods are only marginally better than a naive baseline coupled to a SOTA detector. The analysis we did on the more challenging nuScenes dataset in §4.6.5 revealed that the top method in MOT and KITTI, CenterTrack, still performs well on easy videos with respect to different difficulty scores that we created, but its performance drops significantly when evaluating on hard videos. This shows that a failure to look at performance evaluation based on the levels of difficulty within public datasets hide important aspects of performance. Dividing the datasets into partitions based on the difficulty factors reveals where there is a strong contribution for the proposed models. This suggests that it would be beneficial for the tracking community to have this kind of performance comparison based on different difficulty factors on the tracking leaderboards.

In Chapter 5, we have shown that adding the attention encoder module to DEFT (Attention-based DEFT) maintained comparable tracking performance while significantly improved the velocity estimation performance: the mean absolute velocity error is reduced from

0.924 to 0.754. The results also confirms that the end-to-end learning strategy that reframes sequential tasks into multi-task networks, allowing for joint optimization, is effective. The velocity estimation is important task for autonomous vehicles because determining the instantaneous direction and speed of the surrounding actors is a crucial requirement for collision avoidance in many scenarios.

6.3 Methodology

In this dissertation, we follow the common practice of SOTA methods to evaluate the performance of different proposed models. Each of the public datasets used in this work provide public evaluation tools to compute a set of specific metrics, and we used these tools throughout the comparative evaluation and performance analysis of this dissertation.

The models trained and evaluated in this dissertation are complex and take a significant amount of time and resources to train. Thus, with the limited amount of resources available, it was not feasible to have multiple training runs for each experiment to see the variations in the model performance over different runs. Thus, the variation for model instantiation is not as well understood as one might hope. However, we want to highlight that we have analyzed and tested the proposed tracking models on multiple independant benchmarks : MOTA dataset for indoor and outdoor pedestrian tracking, KITTI dataset for vehicles tracking on videos collected on a car in high traffic areas and nuScenes as a large-scale geo-splitted dataset for autonomous driving for tracking 7 object classes. So, if the performance were a one-time fluke of the training distribution, we don't expect to have consistent performance across different benchmarks. Also, by looking on the 2 datasets used for 2D visual tracking , MOTA and KITTI, we can see that the relative ranking of methods that are common to the two are fairly consistent. nuScenes saw a dramatic difference in the performance of CenterTrack and DEFT, and that's why more analysis was done to further investigate the differences in performance.

We have shown that the proposed models in this dissertation were able to achieve SOTA performance. But as we look in the future of AV, a couple of aspects of methodology seem to be

needed. While multiple metrics are used to evaluate different models, these metrics are not enough to analyze the differences in performance of different models and to understand the contribution and the failure of each model. While the proposed difficulty scores (occlusion, displacement and crowdedness) helped us to overcome some of this limitation, it is only a small step in the right direction. Also, not all the metrics are equally severe/consequential, and this thesis makes no attempt to judge the proposed methods in that light. Furthermore, the metrics used in this dissertation are used to evaluate the models performance and not the system performance – in order to deploy the models to a self-driving vehicle, there's a distinct challenge of validation testing that isn't being addressed in this thesis.

6.4 Future Work

The proposed objects geo-localization approach has shown superior performance compared to SOTA methods for static object geo-localization and better suitability for autonomous driving applications. The running time for the proposed approach is 5.3 frames per second, which is less than the frame rate of the cameras for some self-driving data sets such as KITTI [11] and nuScenes [3], on the order of 10Hz. This is mainly because the proposed approach follows a cascaded pipeline that uses an off-the-shelf 2D object detector to obtain the 2D detections, which are then passed to the objects matching network to perform the pose estimation and association the objects across frames. The followed approach essentially processes the image crops of the detected objects twice without sharing any information, inducing accumulated latency to the proposed system. The research we did on DEFT showed that sharing features between the 2D object detector and the object matching network provides opportunities for further joint optimization and inference speed-up. Thus, future plans include extending DEFT to perform joint detection, tracking, and geo-localization in an end-to-end approach. Also, we were limited to evaluating the performance of the proposed approach on traffic lights; application to other static compact objects (signs, etc.) requires creating or identifying new data sets.

Attention-based DEFT achieved superior velocity estimation performance compared to other baseline methods on both simple and complex scenes. It makes velocity estimation only by using objects' context, history, and interactions, which might fail in extreme scenarios such as unpredictable sharp turns or sudden stops. Thus, additional information such as map configuration (e.g., lanes, traffic lights, signs) could help solve such issues. One possible idea would be by rendering HD map into RGB rasterized representation (Bird's Eye View scene color-coded image) [135], passing it through CNNs, extracting local scene context information for each actor using its spatial location in the rasterized image, and then fusing the obtained features with appearance features. This would help the model to account for road context information while modeling the object's velocity. Furthermore, Attention-based DEFT was initially designed to adapt to any sequence modeling; velocity estimation is one possible application. Future work can include its application to other sequence modeling tasks.

All models proposed in this dissertation use the Hungarian algorithm to find the optimal assignment between the new detections and existing tracklets based on the predicted similarity scores. A natural extension to this work is replacing the only non-differentiable module of our models – the Hungarian algorithm – with an equivalent differentiable network to allow complete end-to-end learning. For example, Yihong *et al.* [107] proposed a Deep Hungarian Net (DHN) module that is equivalent to the Hungarian algorithm. DHN estimates the correspondence between object tracklets and ground truth objects to determine differentiable proxies of MOTA and MOTP, which are then used as loss functions to directly optimize the trackers. Their experiments showed improvements in the existing trackers performance when using their proposed fully differentiable framework.

While I think that all the mentioned above directions are exciting and have promising implications for the future of achieving a fully autonomous vehicle, it is indeed a very complex problem that requires enormous research and engineering work. I am grateful to have the opportunity to be present and participate a little in the process of building a self-driving vehicle. The work of this dissertation only focuses on some of the essential components of the autonomy

system. I have and always will be keen on keeping up with the updates and breakthroughs in this field, and I cannot express my excitement and eagerness to witness the achievement of Level 5 autonomy.

Bibliography

[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008.

[2] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection. *arXiv preprint arXiv:1908.09492*, 2019.

[3] 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 (CVPR)*, 2020.

[4] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric. In *IEEE International Conference on Image Processing (ICIP)*, 2017.

[5] Vladimir A Krylov, Eamonn Kenny, and Rozenn Dahyot. Automatic Discovery and Geotagging of Objects from Street View Imagery. *Remote Sensing*, 2018.

[6] Ahmed Samy Nassar, Sébastien Lefèvre, and Jan Dirk Wegner. Simultaneous multi-view instance detection with learned geometric soft-constraints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[7] Xingyi Zhou, V. Koltun, and Philipp Krähenbühl. Tracking Objects as Points. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[9] *Global status report on road safety 2018*. Geneva: World Health Organization, 2018.

[10] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? the KITTI Vision Benchmark Suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3D Object Detection from Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as Points. *arXiv preprint arXiv:1904.07850*, 2019.

[14] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. PnPNet: End-to-End Perception and Prediction with Tracking in the Loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[15] Vladimir A Krylov and Rozenn Dahyot. Object Geolocation Using MRF Based Multi-Sensor Fusion. In *IEEE International Conference on Image Processing (ICIP)*, 2018.

[16] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1999.

[17] Zhe Lin, Larry S Davis, David Doermann, and Daniel DeMenthon. Hierarchical Part-Template Matching for Human Detection and Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2007.

[18] Alex Pentland, Baback Moghaddam, Thad Starner, et al. View-based and modular eigenspaces for face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[19] Joseph L Mundy. Object recognition in the geometric era: A retrospective. *Toward category-level object recognition*, pages 3–28, 2006.

[20] Edgar Osuna, Robert Freund, and Federico Girosit. Training support vector machines: an application to face detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997.

[21] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.

[22] Paul Viola and Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[23] Hiroshi Murase and Shree K Nayar. Visual learning and recognition of 3-d objects from appearance. *International journal of computer vision*, 14(1):5–24, 1995.

[24] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[25] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[26] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, IEEE International Conference on*, volume 3, pages 1470–1470. IEEE Computer Society, 2003.

[27] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[28] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60:84 – 90, 2012.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[31] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88:303–338, 06 2010.

[32] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[34] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[35] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[38] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[40] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[41] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2018.

[42] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2018.

[43] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[44] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155, 2009.

[45] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A Convolutional Network for Real-time 6-DoF Camera Relocalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[47] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *Robotics: Science and Systems (RSS)*, 2018.

[48] Radu Timofte and Luc Van Gool. Multi-view manhole detection, recognition, and 3D localisation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops)*, 2011.

[49] Bahman Soheilian, Nicolas Paparoditis, and Bruno Vallet. Detection and 3D reconstruction of traffic signs from multiple view color images. *ISPRS journal of photogrammetry and remote sensing*, 2013.

[50] Ramya Hebbalaguppe, Gaurav Garg, Ehtesham Hassan, Hiranmay Ghosh, and Ankit Verma. Telecom Inventory Management via Object Recognition and Localisation on Google Street View Images. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017.

[51] Weixing Zhang, Chandi Witharana, Weidong Li, Chuanrong Zhang, Xiaojiang Li, and Jason Parent. Using Deep Learning to Identify Utility Poles with Crossarms and Estimate Their Locations from Google Street View Images. *Sensors*, 2018.

[52] Steve Branson, Jan Dirk Wegner, David Hall, Nico Lang, Konrad Schindler, and Pietro Perona. From Google Maps to a Fine-Grained Catalog of Street trees. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018.

[53] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[54] Ahmed Samy Nassar, Sébastien Lefèvre, and Jan D Wegner. GeoGraph: Learning graph-based multi-view object detection with geometric cues end-to-end. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[55] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[56] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-Object Tracking. *arXiv preprint arXiv:1907.03961*, 2019.

[57] 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*, pages 1–31, 2020.

[58] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[59] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[60] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[61] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[62] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. POI: Multiple Object Tracking with High Performance Detection and Appearance Feature. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[63] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple Online and Realtime Tracking. In *IEEE International Conference on Image Processing (ICIP)*, 2016.

[64] Dawei Zhao, Hao Fu, Liang Xiao, Tao Wu, and Bin Dai. Multi-Object Tracking with Correlation Filter for Autonomous Vehicle. *Sensors*, 2018.

[65] Minyoung Kim, Stefano Alletto, and Luca Rigazio. Similarity Mapping with Enhanced Siamese Network for Multi-Object Tracking. *arXiv preprint arXiv:1609.09156*, 2016.

[66] Bing Wang, Li Wang, Bing Shuai, Zhen Zuo, Ting Liu, Kap Luk Chan, and Gang Wang. Joint Learning of Siamese CNNs and Temporally Constrained Metrics for Tracklet Association. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2016.

[67] Anton Milan, Seyed Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online Multi-Target Tracking Using Recurrent Neural Networks. *arXiv preprint arXiv:1604.03635*, 2016.

[68] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking The Untrackable: Learning To Track Multiple Cues with Long-Term Dependencies. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[69] Sangyun Lee and Euntai Kim. Multiple Object Tracking via Feature Pyramid Siamese Networks. *IEEE Access*, 7:8181–8194, 2018.

[70] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-Speed Tracking-by-Detection Without Using Image Information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[71] Kota Yamaguchi, Alexander C Berg, Luis E Ortiz, and Tamara L Berg. Who are you with and where are you going? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[72] Bi Song, Ting-Yueh Jeng, Elliot Staudt, and Amit K Roy-Chowdhury. A Stochastic Graph Evolution Framework for Robust Multi-Target Tracking. In *European Conference on Computer Vision (ECCV)*. Springer, 2010.

[73] Mikel Rodriguez, Saad Ali, and Takeo Kanade. Tracking in Unstructured Crowded Scenes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009.

[74] Hamid Izadinia, Imran Saleemi, Wenhui Li, and Mubarak Shah. (MP)2T: Multiple People Multiple Parts Tracker. In *European Conference on Computer Vision (ECCV)*. Springer, 2012.

[75] Bo Yang and Ramakant Nevatia. Multi-Target Tracking by Online Learning a CRF Model of Appearance and Motion Patterns. *International Journal of Computer Vision*, 107(2):203–217, 2014.

[76] Longyin Wen, Zhen Lei, Siwei Lyu, Stan Z Li, and Ming-Hsuan Yang. Exploiting Hierarchical Dense Structures on Hypergraphs for Multi-Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[77] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple Hypothesis Tracking Revisited. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[78] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[79] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[80] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[81] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by Tracking: Siamese CNN for Robust Target Association. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2016.

[82] Seung-Hwan Bae and Kuk-Jin Yoon. Confidence-Based Data Association and Discriminative Deep Appearance Learning for Robust Online Multi-Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[83] Alexis Mignon and Frédéric Jurie. PCCA: A New Approach for Distance Learning from Sparse Pairwise Constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[84] ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal S Mian, and Mubarak Shah. Deep Affinity Network for Multiple Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[85] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[86] L. Chen, H. Ai, Z. Zhuang, and C. Shang. Real-Time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[87] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[88] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-Object Tracking with Quadruplet Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[89] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the Connectivity: Multi-Object Tracking with TrackletNet. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 2019.

[90] Keng-Chi Liu, Yi-Ting Shen, and Liang-Gee Chen. Simple online and realtime tracking with spherical panoramic camera. In *IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2018.

[91] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. In *ASME Journal of Basic Engineering*, 1960.

[92] R. Zhang and S. Cao. Extending Reliability of mmWave Radar Tracking and Detection via Fusion With Camera. *IEEE Access*, 7:137065–137079, 2019.

[93] Xi Chen, Xiao Wang, and Jianhua Xuan. Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques. In *International Conference on Engineering and Applied Science (ICEAS)*, 2012.

[94] Lu Wang, Lisheng Xu, Min Young Kim, Luca Rigazico, and Ming-Hsuan Yang. Online multiple object tracking via flow and convolutional features. In *IEEE International Conference on Image Processing (ICIP)*, 2017.

[95] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[96] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[97] Hui Zhou, Wanli Ouyang, Jian Cheng, Xiaogang Wang, and Hongsheng Li. Deep Continuous Conditional Random Fields with Asymmetric Inter-object Constraints for Online Multi-object Tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[98] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking With 2D-3D Multi-Feature Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[99] Jiahe Li, Xu Gao, and Tingting Jiang. Graph Networks for Multiple Object Tracking. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 719–728, 2020.

[100] Guillem Brasó and Laura Leal-Taixé. Learning a Neural Solver for Multiple Object Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[101] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-Object Tracking with Transformers. *arXiv preprint arXiv:2101.02702*, 2021.

[102] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020.

[103] Wei-Chih Hung, Henrik Kretzschmar, Tsung-Yi Lin, Yuning Chai, Ruichi Yu, Ming-Hsuan Yang, and Drago Anguelov. SoDA: Multi-Object Tracking with Soft Data Association. *arXiv preprint arXiv:2008.07725*, 2020.

[104] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[105] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[106] Qiankun Liu, Qi Chu, Bin Liu, and Nenghai Yu. GSM: Graph Similarity Model for Multi-Object Tracking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[107] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How To Train Your Deep Multi-Object Tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[108] 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 (ECCV)*. Springer, 2016.

[109] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to Track and Track to Detect. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[110] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards Real-Time Multi-Object Tracking. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[111] Mohamed Chaabane, Lionel Gueguen, Ameni Trabelsi, Ross Beveridge, and Stephen O'Hara. End-to-end Learning Improves Static Object Geo-localization from Video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2063–2072, 2021.

[112] Min Bai and Raquel Urtasun. Deep Watershed Transform for Instance Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[113] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. UPSNet: A Unified Panoptic Segmentation Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[114] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[115] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[116] 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. *arXiv preprint arXiv:1903.11027*, 2019.

[117] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[118] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[119] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[120] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online Multi-Object Tracking with Dual Matching Attention Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2018.

[121] Mohamed Chaabane, Peter Zhang, Ross Beveridge, and Stephen O'Hara. DEFT: Detection Embeddings for Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021.

[122] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint Monocular 3D Vehicle Detection and Tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[123] Longtao Chen, Xiaojiang Peng, and Mingwu Ren. Recurrent Metric Networks and Batch Multiple Hypothesis for Multi-Object Tracking. *IEEE Access*, 7:3093–3105, 2018.

[124] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[125] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep Layer Aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[126] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[127] 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 International Conference on Computer Vision (ICCV)*, 2019.

[128] Hasith Karunasekera, Han Wang, and Handuo Zhang. Multiple Object Tracking With Attention to Appearance, Structure, Motion and Size. *IEEE Access*, 7:104423–104434, 2019.

[129] Wongun Choi. Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[130] Nicolas Franco Gonzalez, Andres Ospina, and Philippe Calvez. SMAT: Smart Multiple Affinity Metrics for Multiple Object Tracking. In *International Conference on Image Analysis and Recognition*, pages 48–62. Springer, 2020.

[131] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling Monocular 3D Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[132] 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 (CVPR)*, 2019.

[133] 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 (CVPR)*, 2020.

[134] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In *CoRL*, 2019.

[135] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware Short-term Motion Prediction of Traffic Actors for Autonomous Driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020.

[136] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018.

[137] Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani, and Mohan M Trivedi. TrackMPNN: A Message Passing Graph Neural Architecture for Multi-Object Tracking. *arXiv preprint arXiv:2101.04206*, 2021.

[138] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer Networks for Trajectory Forecasting. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10335–10342, 2021.