KU LEUVEN

FACULTY OF ENGINEERING SCIENCE

imec

360-degree Perception of a Vehicle Surrounding Using a Network of Radars

Tudor Horia Niculescu

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, option Artificial Intelligence

Supervisors:
Prof. dr. ir. Wim Michiels
Dr. Hamed Javadi (imec)
Prof. Hichem Sahli (imec)

Assessors:
Dr. Vera Rimmer
Dr. Sam Michiels

Academic year 2024 – 2025

Contents

Abstract

Multi-object detection and tracking are the main perception tasks required for autonomous driving. Modern vehicles are equipped with a large array of sensors such as radar, lidar, and cameras. Unlike cameras and lidar, radar offers robustness in adverse weather conditions and provides direct velocity measurements, though it suffers from lower angular resolution and noise.

This thesis introduces a novel polar quantized radar data representation, which incorporates temporal information from consecutive sweeps. Through this lightweight early fusion approach, the radar point clouds provided by multiple sensors at multiple frames are condensed into a denser format, which is more suitable to vision models. Leveraging this representation, two models were developed: a center-point detection model inspired by CenterNet and a tracking extension for it inspired by CenterTrack. The backbone in these architectures is replaced with a modified version of the Swin Transformer, adapted to exploit the spatial and temporal relationships in the radar data. These models were evaluated on the nuScenes dataset using standard metrics, demonstrating their ability to track dynamic objects but highlighting challenges with static or small object detection in cluttered scenarios.

The results underscore radar's potential as a complementary sensor in autonomous systems. While standalone radar may face limitations due to data sparsity, its integration with other modalities offers a path to robust and efficient perception systems.

Chapter 1

Introduction

1.1 Background and Motivation

Learning-based perception and control models, as well as significant improvements in sensing technologies have sparked a steady growth in interest in autonomous driving. These developments have catalyzed rapid innovation and substantial investments by companies like Waymo, which have also gathered extensive datasets to train and refine autonomous driving systems. Standardization efforts, such as the introduction of the SAE Levels of Driving Automation, have further provided a structured framework to classify and evaluate these systems.

Autonomous driving in complex environments presents unique challenges. These environments often feature dynamic elements such as unpredictable road users and target occlusions. Such scenarios create significant hurdles for traditional data-driven models with manually crafted features, which struggle to handle the variability and complexity inherent in these conditions. In contrast, deep learning-based methods have emerged as a more viable alternative, enabled by increased computational power and the availability of large, annotated datasets like Waymo Open Dataset[29] and nuScenes [3]. These methods leverage multiple sensing modalities, including cameras, lidar, radar, and even microphones, to achieve robust performance.

Among these sensing technologies, radar has undergone considerable advancements, making it an increasingly attractive option for use in autonomous systems. Traditional radar applications, such as air traffic control, missile defense, and surveillance, have benefited from robust multi-target tracking algorithms like Joint Probabilistic Data Association (JPDA), Multiple Hypothesis Tracking (MHT), and Random Finite Sets (RFS) based filters. These algorithms have significantly improved the ability to distinguish between closely spaced targets in cluttered environments, enhancing radar performance in critical applications.

Technological developments such as Multiple-Input Multiple-Output (MIMO) radar have further transformed target tracking. MIMO radar systems offer higher spatial resolution, better target discrimination, and improved tracking performance. These capabilities, once primarily applied to traditional domains, are now of growing interest in autonomous driving.

In the context of autonomous vehicles, radar offers several advantages over other sensing modalities like cameras and lidar. Radar is highly robust in adverse weather conditions such as fog, rain, and snow, and it is less affected by poor lighting conditions. Furthermore, radar's inherent ability to measure the velocity of targets directly provides critical information for motion tracking and prediction. However, radar also has limitations, including lower spatial resolution compared to cameras and lidar, which restricts its ability to capture fine details. Additionally, radar readings are often noisy due to clutter, which can arise from static objects, multi-path reflections, or interference, posing challenges for reliable interpretation.

Given these strengths and limitations, radar has emerged as a compelling focus for applying deep learning techniques to multi-target tracking. This thesis explores the development of a deep learning model designed specifically for radar-based detection and tracking in autonomous driving scenarios, aiming to address the challenges of cluttered environments while leveraging radar's unique advantages.

1.2 Problem Statement

The goal of the thesis is to explore and develop deep learning methods for Frequency Modulated Continuous Wave (FMCW) radar perception of a vehicle surrounding in the context of autonomous driving. It is of interest to perceive the whole horizontal 360-degree surrounding of the ego-vehicle, which is not attainable by using a single radar, as FMCW radars usually have a Field of View (FoV) of about 120°[28]. As such, it is necessary to combine the information received from multiple radar sensors, which are mounted at various positions on the ego-vehicle. Multiple-object detection and multiple-object tracking are the perception tasks which shall be performed on the vehicle surrounding.

Multiple-object detection is a classical task in the field of Computer Vision, in which, for a single input frame, targets of certain types are detected and their position, size, and potentially class as well are predicted. This is generally done through outputting a set of bounding boxes, for example axis-aligned rectangles in the 2D RGB image case, which tightly enclose the targets, while also specifying the target class. Some example situations in which multiple object detection is used are medical diagnostics from static images - recognizing fractures on X-rays, detecting tumors on MRI or CT scans, or counting objects from images - livestock from an aerial view photo, or the number of cars stopped at a traffic light, for estimating traffic flow. In the autonomous driving scenario, a bounding box shall be outputted for targets of interest from around the ego-vehicle: other vehicles, pedestrians, street furniture, etc.

Multiple-object tracking (MOT), also named Multi-target tracking (MTT) is the task of identifying, and following, a set of targets over a time span. Unlike single frame object detection, MOT focuses on detecting objects that appear in a sequence of frames, and associating them through time, by assigning each different object a consistent identification number. As such, each object will form a track across a sequence of frames, for example a video in the RGB camera case. A track may spawn,

when a new object enters the surveilled field of view, or a track may be deleted when an object exits it. Objects that are occluded, or that reappear - during the intersection of two or more trajectories, must as well be accounted for. Examples of situations in which multi-object tracking is used are sports analytics - tracking players and the ball in order to do real-time strategy analysis, and extract performance metrics, or livestock tracking - to study animal behavior such as migration patterns and pack dynamics. In the autonomous driving case, each road-user, besides being detected and put into a bounding box, shall as well ideally be given a unique identifier which is consistent across all frames - this means that the identifier is kept the same from when the object enters the field of view of the sensors, until it exits.

1.3 Research Objectives

The first objective of the thesis is to examine state-of-the-art methods in camera and lidar-based perception, analyzing their architectural decisions based on the sensing modalities' specific advantages and disadvantages. By contrasting these modalities with radar approaches, this analysis will lay the foundation for understanding radar's unique potential and limitations in autonomous driving scenarios.

The second objective involves exploring a comprehensive 360-degree autonomous driving dataset to inform the design and evaluation of radar perception systems. These datasets integrate multi-modal sensor data collected from various environments, including urban streets, highways, and adverse weather scenarios. By studying the structure and annotations of such a dataset, we aim to gain insights into the challenges and opportunities specific to radar-based multi-object detection and tracking. Key areas of focus include the diversity of road user types, sensor overlap for ensuring complete coverage, and the quality of ground truth labels for evaluating radar performance. This understanding will guide the development of models tailored to radar's characteristics and requirements.

The third and fourth objectives are centered on the design, implementation, and evaluation of a deep-learning pipeline specifically tailored for radar perception. The pipeline fuses inputs from multiple radar sensors to construct a 360-degree representation of the vehicle's surroundings, addressing the field-of-view limitations of individual radars. It performs multi-object detection and tracking, taking advantage of the strength of the radar to measure the velocity of the target. The pipeline is implemented using several recent state-of-the-art papers and open-source software and data. The proposed pipeline undergoes quantitative and qualitative testing on an autonomous driving dataset. Domain-standard quantitative metrics such as Average Precision (AP) and Average Multi-Object Tracking Accuracy (AMOTA) provide measurable detection and tracking benchmarks, while qualitative evaluations examine performance in complex scenarios, such as dynamic environments and cluttered traffic.

1.4 Contributions

This thesis proposes a novel representation for radar sweeps that optimally addresses the unique advantages and challenges associated with radar sensors. The proposed representation compresses radar data into a polar format, which naturally aligns with the radial nature of radar measurements. Additionally, it incorporates temporal information by augmenting the representation with data from multiple consecutive radar sweeps. This temporal augmentation enriches the model's ability to recognize patterns and track objects over time, improving robustness in complex driving scenarios. Leveraging this denser representation, a radar-specific multi-object detection model was developed based on CenterNet [39], a widely recognized architecture in computer vision. To enhance compatibility with radar data, the backbone of CenterNet was replaced with the Swin Transformer [17], a vision transformer capable of associating data over multiple axes, in our case the spatial and temporal dimensions of the radar representation.

Building on the detection model, the research extends the architecture to include multi-object tracking, inspired by the way CenterTrack [38] builds upon CenterNet. This extension enables time association, allowing the model to not only detect objects but also track their movement across frames. Both the detection and tracking models were tested and evaluated on the nuScenes dataset, a benchmark in autonomous driving research. The evaluation compares the models' performance across the key metrics proposed by nuScenes.

1.5 Thesis Outline

The rest of this thesis is structured as follows. Chapter 2 goes through an introduction to the FMCW Radar signal processing chain, classical approaches to tracking from Control Theory, followed by a review of notable papers from the domain of data-driven deep learning tracking. The chapter ends with a presentation of the state-of-the-art papers from which this work is inspired. Chapter 3 explains the structure and limitations of the radar data from the nuScenes dataset, and describes how it is preprocessed. It continues by describing the structure of the proposed detection and tracking models: Radar CenterNet-Swin and Radar CenterTrack-Swin. Chapter 4 goes through the training and testing of said models, comparing them to the state of the art (when possible) and providing qualitative and quantitative results. Chapter 5.1 describes further ideas which could be followed up and potential improvements. Chapter 5.2 sums up the insights acquired through the development of this thesis project, by reviewing the results and drawbacks of the proposed methods.

Chapter 2

Background review

2.1 FMCW Radar Fundamentals

Radar, an acronym for radio detection and ranging, operates within the electromagnetic spectrum and was initially developed for military and navigation purposes. Its fundamental principle of operation is analogous to bat echolocation: a wave is transmitted, and the returning echo is analyzed to determine the position and speed of detected objects. FMCW (Frequency Modulated Continuous Wave) radar, a common type in automotive applications, operates within the 77-81 GHz frequency band. It transmits chirps, which are waveforms that linearly increase in frequency over time, hence its name. An example of a series of emitted chirps can be seen in blue in Figure 2.2c, with their returns in green.

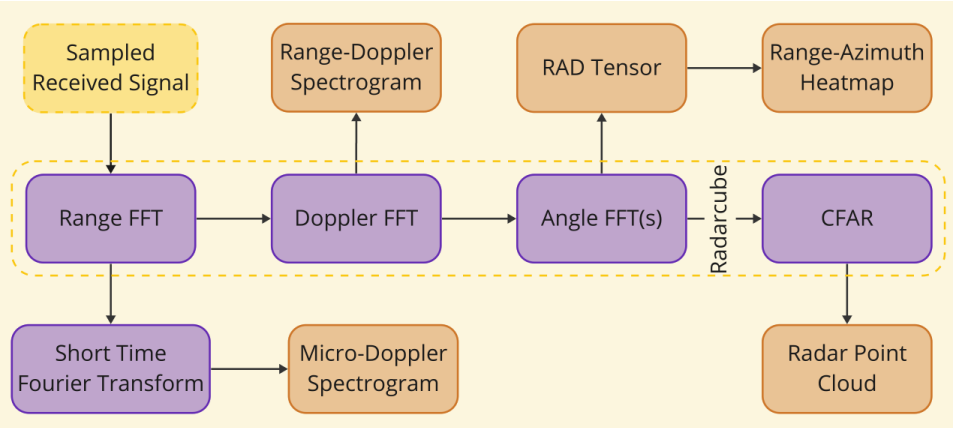


FIGURE 2.1: Processing flowchart for obtaining the various FMCW Radar data formats. Image taken from [28].

In a basic single-transmit and single-receive antenna configuration, range detection is achieved by analyzing the phase shifts in returning chirps. The first step involves applying a Fast Fourier Transform (FFT) to these return signals, yielding range information but lacking velocity or angular data – left side of Figure 2.2e. The FFT

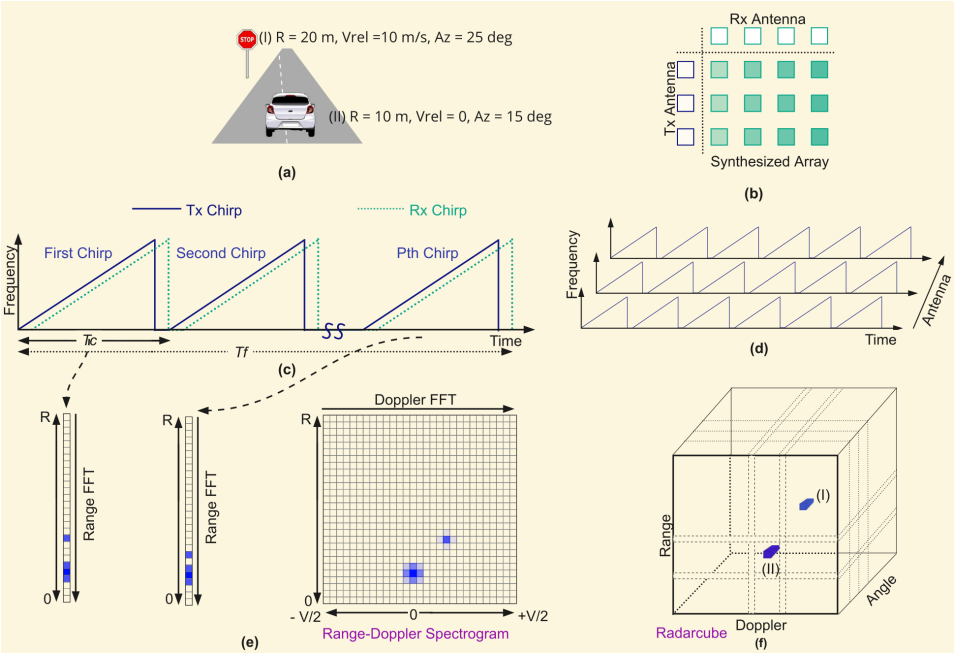


Figure 2.2: Overview of FMCW Radar signal processing pipeline. Image taken from [28].

is applied on a chirp by chirp basis, also named fast-time. An example output of this Range FFT is shown in Figure 2.2a, in which each of the two "blobs" come from a target . The car, which is closer, returns a stronger signal, which also covers multiple range bins, due to the higher radar reflectivity of the car. For radial velocity detection, a slow-time FFT is applied across the results of the fast-time FFT over multiple chirps, also called the Doppler FFT, capturing the Doppler shift introduced by moving objects. The resulting 2D range-Doppler spectrogram provides both range and velocity information – right side of Figure 2.2e. In the spectrogram, the car appears clearly in the middle, at a closer range, and zero relative velocity, while the stop sign leaves a smaller "fingerprint" on the right side – it has a positive radial velocity, meaning it is getting closer to the ego-vehicle.

With MIMO (Multiple Input Multiple Output) radar systems, multiple transmit and receive antennas generate a matrix of virtual antennas – Figure 2.2b. For each virtual antenna, a 2D range-Doppler spectrogram is computed, and an additional FFT is performed across the antennas to estimate azimuth and elevation information – Figure 2.2d. By applying the Angle FFT in the vertical direction, elevation information is obtained, while applying it on the horizontal direction results in azimuth information. The result is a radar cube, also known as the Range-Azimuth-Doppler (RAD) tensor, which integrates range, velocity, and angular data into a higher-dimensional structure – Figure 2.2f. The cells in the RAD tensor are represented by the radar cross-section (RCS), a measure of how strongly an object reflects radar signals. The RCS provides information about object size and material properties, aiding in object classification and identification.

The RAD tensor, although comprehensive, is computationally intensive due to its large size. It can be projected into a range-azimuth format by compressing the Doppler dimension, which simplifies the data at the cost of losing fine-grained velocity details. Alternatively, peak extraction methods such as the Constant False Alarm Rate (CFAR) algorithm can be applied to the RAD tensor to identify significant reflections. This process generates a radar point cloud, an unordered set of points where each point is characterized by range, angle, velocity, RCS, and other parameters. While point clouds are less data-intensive than RAD tensors, they retain essential information about detected targets. These representations are often produced directly on-chip to reduce computational overhead.

2.2 Model based tracking

2.2.1 Single target

State estimation, a very important aspect of target tracking, is a thoroughly developed topic in the domain of System and Control Theory. It is the process of inferring the hidden or unobservable state of a dynamic system based on a set of measurements or observations. It continually updates a belief, an estimate of the system's state, by combining the prediction of a mathematical model which describes the system's dynamics with the information received from the incoming measurements. Classical methods and algorithms are the Kalman filter [12], and the Particle filter [9]. A new trend in state estimation departs from the classical System theory, and explores data-driven, or hybrid-driven approaches [11].

Bayesian filter and the Kalman filter

Bayesian filtering is a probabilistic approach to estimate the state of a dynamic system based on a series of observations. As new data becomes available, it leverages Bayes' theorem to propagate the state estimate distribution through time, incorporating both prior information and the likelihood of the measurements. Bayesian filtering works in two main steps: prediction and update.

In the prediction step, at time step t, the prior belief of the system's state is calculated by using the posterior of the previous time step $p(x_{t-1}|z_{1:t-1})$ and a state evolution model $p(x_t|x_{t-1})$ using the following formula:

$$p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})\, dx_{t-1}$$

In the update step, the new incoming measurements z_t are incorporated with the estimated prior, in order to calculate the new posterior belief using Bayes' theorem. $p(z_t|x_t)$ represents the likelihood of observing z_t given the state x_t.

$$p(x_t|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \propto p(z_t|x_t)p(x_t|z_{1:t-1})$$

The denominator $p(z_t|z_{1:t-1})$ is constant relative to the state x_t so in practice it is usually omitted and the distribution normalized. These two steps are applied in recursive fashion, in order to estimate x_t at each new timestep.

The Kalman filter[12] is a specific implementation of Bayesian filtering and it is used for linear dynamical systems in the context of Gaussian noise. The filter provides an optimal and computationally efficient solution for state estimation problems, by estimating the state x_t and covariance P_t using measurements $z_{1:t}$. The Kalman filter uses the same two steps as Bayesian filtering: prediction and update.

In the prediction step, the linear movement model encoded by matrix F propagates the mean and covariance of the estimated state, taking into account the Gaussian process noise represented by Q.

$$\hat{x}_{t|t-1} = F\hat{x}_{t-1}$$
$$P_{t|t-1} = FP_{t-1|t-1}F^T + Q$$

In the update step, using matrix H that transforms from state space to measurement space, a predicted measurement is calculated. The Kalman gain, K_t, represents the level of sureness in the incoming measurement z_t compared to the predicted measurement, by taking into account their covariances and the measurement noise R. Using this Kalman gain, the new state mean $x_{t|t}$ and covariance $P_{t|t}$ is calculated.

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H\hat{x}_{t|t-1})$$
$$P_{t|t} = (I - K_tH)P_{t|t-1}$$

A Kalman gain of zero would represent updating the state posterior as the predicted state $x_{t|t-1}$ because of a highly out-of-distribution measurement. Conversely, a gain of one would represent taking into account only the measurement, for example from a very precise sensor, and updating the state posterior with it.

In line with the Bayesian filtering, the Kalman filter works in a recursive manner, continuously propagating the mean and covariance of the state estimation through time. The filter provides a systematic approach to estimate the state of a linear system while accounting for noise and uncertainty, but only under the assumptions of a linear motion model, and Gaussian distributed process noise.

In the case of non-linear system dynamics, the Extended Kalman filter is applied, which approximates the linear transformation matrices F and H with real functions, and linearizes them around the point of interest through the first order multivariate Taylor expansion.

2.2.2 Multiple target tracking

Random Finite Sets and the PHD intensity function

Random Finite Sets (RFS) is a concept from Finite Set Statistics (FISST)[19] which is an extension of classical Bayesian statistics developed specifically for multi-target problems. The aim of FISST is to provide a simple engineering-oriented mathematical

framework to aid in the transporting of the classical single-target, single-measurement theory and algorithms to the multi-sensor, multi-target realm of MTT[18].

In simple terms, a RFS is a point pattern such as the readings from the radar or the estimated target states in the MTT problem. A RFS is a finite-set-valued variable which is defined by its discrete cardinality distribution, and by a family of joint positional distributions. The cardinality distribution represents the number of points in the set, while the family of distributions characterizes the distribution of the points conditioned on the cardinality distribution[7]. The main concern that it addresses is estimating a distribution of a changing number of variables through time, with the final goal of being able to deal with events such as target birth and death, which appear in most MTT problems. A more detailed explanation of RFS can be found in [19].

In this section, variables written in bold such as $\mathbf{x}$ represent set-valued variables.

The Probability Hypothesis Density (PHD) is the first-order statistical moment of an RFS, equivalent to the expectation for a simple random variable. It represents the density of points from the RFS in any location. For an RFS X defined on $\mathcal{X} \subset \mathbb{R}^d$, the PHD is defined by v, a non-negative function on $\mathcal{X}$ such that for any region $S \subset \mathcal{X}$[20]:

$$\mathbb{E}(|X \cap S|) = \int_S v(\mathbf{x}) d\mathbf{x}$$

The result of the integral represents the number of points from the RFS X that can be found in the region S. In order to generate estimates of the points' locations, first their total number is estimated through $\hat{N} = \int v(\mathbf{x}) d\mathbf{x}$. By extracting the top $\hat{N}$ peaks from the PHD v, the locations of the targets is then estimated.

The GM-CPHD filter

The Gaussian Mixture Cardinality Probability Hypothesis Density (GM-CPHD) filter[34] is the adaptation of the Kalman filter to the multi-measurement, multi-state domain by making use of RFSs and the PHD intensity function and is an extension of the Gaussian Mixture PHD filter[6]. Compared to the GM-PHD filter which only propagates in a Bayesian fashion the targets' PHD intensity, the GM-CPHD also estimates cardinality distribution - the number of targets. Similarly to the PHD filter, the CPHD filter takes into account target birth probabilities, target detection probabilities, and also models the clutter intensity.

As multi-target Bayes inference is intractable, the PHD intensities are approximated using Gaussian mixtures. Similarly to the Kalman filter, the CPHD filter propagates the PHD intensity function v and the cardinality distribution p through time, updating the posteriors from time step $k-1$ by making use of the new measurements Z_k. At the beginning of timestep k, they previous priors are $p_{k-1}(n)$ which represents the probability of having n targets in the estimated state, with $\int p_{k-1}(n) dn = 1$, and $v_{k-1}(\mathbf{x})$, the PHD intensity represented as a Gaussian mixture:

$$v_{k-1}(\mathbf{x}) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)}\right)$$

The prediction equations are as follows:

$$p_{k|k-1}(n) = \sum_{j=0}^{n} p_{\Gamma,k}(n-j) \sum_{\ell=j}^{\infty} C_j^{\ell} p_{k-1}(\ell) p_{S,k|k-1}^{j} \left(1 - p_{S,k}\right)^{\ell-j}$$

$$v_{k|k-1}(\mathbf{x}) = \gamma_k(\mathbf{x}) + p_{S,k|k-1} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_{S,k|k-1}^{(j)}, \mathbf{P}_{S,k|k-1}^{(j)}\right)$$

where

$$\mathbf{m}_{S,k|k-1}^{(j)} = \mathbf{F}_{k|k-1} \mathbf{m}_{k-1}^{(j)}$$

$$\mathbf{P}_{S,k|k-1}^{(j)} = \mathbf{Q}_k + \mathbf{F}_{k|k-1} \mathbf{P}_{k-1}^{(j)} \mathbf{F}'_{k|k-1}$$

$$\gamma_k(\mathbf{x}) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_{\gamma,k}^{(i)}, \mathbf{P}_{\gamma,k}^{(i)}\right)$$

In a similar way to the Kalman filter, the means and covariances of the Gaussians from the intensity mixture are predicted taking into account the linear dynamical model $\mathbf{F}$ and process noise $\mathbf{Q}$. Their weights are subsequently scaled by $p_{S,k|k-1}$, the probability of target survival. A birth PHD γ_k is also added in order to take into account potential target births at time step k.

The cardinality is adapted to take into account the birth cardinality $p_{\gamma,k}$ and all the possible ways of choosing j targets to survive out of the existing, potentially infinite number of targets, times the chance of actually having j targets in the previous cardinality posterior, denoted by $p_{k-1}(j)$.

As the update equations are more complex, and can be found in [34], we will just briefly discuss the logic behind them here. The PHD mixture posterior v_k is the GM obtained after "applying" a Kalman filter on each possible pair of Gaussian component from the estimated prior mixture $v_{k|k-1}$, and each new measurement. The Kalman gain is now calculated individually for each assigned pair, taking into account the likelihood of making that certain measurement to target association, in the context of predefined model parameters, such as target birth, death, detection probability, and clutter distribution. By doing so, the number of Gaussians in the new posterior increases from $J_{k|k-1}$ to $J_{k|k-1} \cdot (|Z_k| + 1)$, as each Gaussian from the estimated prior is combined with $|Z_k|$ potential measurement associations, and one extra for the mis-detection possibility. As the number of components in the PHD mixtures increases without bound, their number can quickly get out of hand. In this situation, pruning and merging techniques have to be used before progressing to the next time step. Through pruning, Gaussian components with weights lower than a certain threshold are eliminated. Through merging, components with similar means and covariances are merged into a single one, greatly reducing the number of components. Further information on this topic including algorithms can be found in [35]. The cardinality posterior p_k is also updated by using a similar likelihood as the one used for the calculation of v_k. By following these four update steps, the new posteriors p_k and v_k are obtained from the old ones p_{k-1} and v_{k-1} and thus the algorithm progresses to the next time step $k+1$.

In order to extract the target states prediction, first the number of targets is estimated using MAP (Maximum a posteriori):

$$\hat{N}_k = \text{argmax } p_k(\cdot)$$

and then the components with the highest $\hat{N}_k$ weights from the mixture are extracted, their means representing the estimated target states.

Target tracks can be identified by assigning labels to the mixture components, and keeping label continuity with algorithms such as the ones presented in [5]. The behavior of the GM-CPHD filter can be also improved through heuristic modeling for the cases of intersecting or splitting trajectories. Besides a birth PHD, a spawn PHD can also be added to the PHD intensity in the update step. The spawn PHD is derived from the predicted prior by adding Gaussian noise. Additionally, the GM-CPHD filter can be extended to non-linear target dynamics by replacing the Kalman prediction and update sub-steps with the extended Kalman filter ones.

2.3 Data-driven tracking concepts

2.3.1 Paradigms

In the Deep Learning space, there are two main approaches to the tracking problem: tracking by detection, and joint detection and tracking.

Tracking by detection is a two-step approach, which completely separates the detection from the association task. As object detection is one of the most researched tasks in computer vision, many high-quality models for camera such as YOLO[36] are available. The RGB image, a high density input format, is ideal, as it provides information such as texture and color, which can help with object detection and classification tasks. Conversely, the lower angular resolution of lidars and radars may lead to worse performance in detection and classification compared to camera.

The detection results – object proposals are fed as input to the tracking algorithm, which has to update its tracks accordingly: extend existing ones with the new detections, create new tracks from the new detections, or delete old tracks which do not receive new predictions anymore. This optimal object-to-track association problem is usually done on criteria such as proximity, appearance features, and motion patterns, but when many objects are present, misidentification – assigning an object to a wrong track may happen. Classical algorithms such as the Hungarian Method[13] are used to solve this association problem with polynomial complexity in the number of detections.

Tracking by detection's main advantage – its modular structure which permits making use of the state of the art detectors for any sensor, is also its greatest weakness. Errors from the detector – such as missed detections are propagated through to the tracker, potentially leading to multiple subsequent frames of erroneous tracking predictions. This, combined with the lower performance of lidar and radar detectors, and the increased latency introduced by separate detection and tracking steps, makes tracking by detection less suitable for real-time applications such as autonomous driving.

Joint detection and tracking is the opposite approach. In this case, the detection and the tracking are both done by a single model, which concurrently predicts object detections, and their tracks. Through doing this, the tracker becomes more robust, as previous frames can be used to provide temporal context. As detections are also conditioned by previous tracks, missed detections are rarer, and performance in occlusion scenarios is better. This higher logical model complexity requires careful consideration of both the detection and tracking tasks but permits baking in situation and sensor specific modifications, that better suit the tracking scenario at hand.

Another considerable advantage of joint detection and tracking compared to tracking by detection is the fact that target re-identification (reID) is not necessary as a completely separate section of the model. In joint detection and tracking, reID is integrated into the model by learning appearance features alongside object detection. reID is comparatively more difficult to apply the sparser the data is. Consequently, reID functions better on the informational-dense RGB images of the camera, than on sparse point clouds, such as the ones provided by radars.

2.3.2 Data association through time for tracking

In joint detection and tracking, associating information across multiple frames requires a model architecture capable of capturing temporal dependencies. Since object detection and tracking systems must reliably associate an object's identity over time, especially in dynamic environments, selecting an appropriate architecture is critical. Various approaches can be adopted, each offering distinct advantages and disadvantages depending on the application and performance requirements.

One straightforward approach is stacking frames from consecutive timesteps as separate input channels to the network. In this method, the network, often a Convolutional Neural Network (CNN), is tasked with feature extraction across both spatial and temporal dimensions. The advantage of this method lies in its simplicity: by treating multiple frames as a single input, the temporal correlations are implicitly learned by the feature extractor, without requiring a specialized temporal mechanism. However, this approach relies heavily on the CNN's capacity to capture temporal patterns, which may not always be efficient or accurate in modeling long-term temporal dependencies.

A more advanced approach leverages Recurrent Neural Networks (RNN), such as Bidirectional Long Short-Term Memory (Bi-LSTM) networks. These networks use hidden states to pass information across multiple timesteps, allowing the model to retain information about previously tracked objects, including occluded or partially visible entities. While this method excels at preserving temporal continuity and handling occlusions, its primary drawback is the black-box nature of the hidden state, which makes it difficult to interpret or understand how the model is associating object identities over time. In safety-critical applications such as autonomous driving, this lack of interpretability is undesirable, as model decisions need to be transparent and verifiable.

Another emerging method involves using Temporal Attention or Transformer-based architectures, which apply attention mechanisms between consecutive frames.

These models attend to relevant parts of the input over multiple timesteps, focusing on important regions or objects, and have shown promise in modeling temporal dependencies more explicitly. Compared to RNNs, temporal attention mechanisms offer greater explainability, as the attention maps indicate which parts of the previous frames have influenced the current decisions of the model. This increased transparency makes transformers an attractive option for autonomous driving, where interpretability and robust decision-making are essential.

The choice of how many consecutive frames to feed into the model also presents important trade-offs. Fewer frames lead to shorter sequences, making training faster and easier, and allowing for better parallelization. However, shorter sequences may not capture long-term trajectories or behaviors, which are crucial for accurate tracking in dynamic environments. On the other hand, using longer sequences enables better modeling of object trajectories and behaviors, but at the cost of larger model sizes, slower inference times, and reduced parallelizability during training. Therefore, selecting the optimal sequence length involves balancing the need for real-time performance with the ability to model complex temporal dynamics.

2.3.3 Comparison of input data formats

Both the RAD tensor and radar point cloud formats serve as key data representations in radar processing, and their utility becomes evident when compared to other sensing modalities like cameras and lidar. The RAD tensor's dense grid structure shares similarities with the pixel grids of RGB images captured by cameras. This density enables the application of machine learning architectures such as Convolutional Neural Networks (CNNs) and vision transformers, which are adept at extracting hierarchical features from structured data. However, radar data has unique properties—for instance, the inclusion of Doppler information and RCS values—that distinguish it from visual data and allow for the analysis of motion and object characteristics in a way cameras cannot.

The radar point cloud, on the other hand, resembles the data output of lidar systems but with significant differences. While lidar point clouds are often denser and more precise in angular resolution, radar point clouds include additional information such as velocity and RCS, which are absent in lidar data. However, radar point clouds are more prone to clutter due to ground reflections, multi-path effects, and lower angular resolution. These differences influence the choice of feature extraction models. For example, techniques like PointNet++[26] and PointPillars[14], designed for lidar point clouds, may struggle with radar data due to its sparseness and irregularities. The reduced angular resolution, which stems from larger bin sizes, and the inclusion of clutter require specialized processing to achieve reliable detection and tracking.

The concept of the receptive field further highlights the differences between RAD tensor and point cloud processing in deep learning. The RAD tensor's regular grid structure allows for systematic receptive field expansion through 3D convolutions, enabling the model to capture both local and global spatial and velocity relationships. This structured approach simplifies the integration of contextual information, such as motion patterns and object interactions. In contrast, radar point clouds, being

unordered and sparse, require adaptive methods to define local neighborhoods and aggregate features hierarchically. While this flexibility accommodates the irregular nature of point clouds, it introduces additional complexity in managing receptive fields and processing clutter.

2.4 Related work

2.4.1 Road User Detection from the RAD Tensor

In Palffy et al.[23] the advantages and disadvantages of the two main radar data formats, the RAD Tensor and the radar point cloud, are discussed in the context of road user detection and classification. In the case of autonomous driving sensing, the former, which is also named low-level data, is especially useful for classification, as the dense Doppler dimension provides detailed information about the velocity patterns of the differently-moving parts of the target, which can be interpreted as a "signature". For example, a human walking gait pattern appears as a specific pattern in the Doppler-time image, and the same happens as well for two-wheeled and four-wheeled vehicles. The latter, the radar point cloud, also named target-level data, has been used predominantly for detection, traditionally after processing it with a clustering algorithm such as DBSCAN[8]. The major downside of this is that the same clustering parameters have to be used for all classes, which leads to error-prone clustering. For example, a car could be clustered as two separate objects, the front part and the back part, while a group of pedestrians or cyclists could be all lumped together. Even worse, individual pedestrians which due to the low angular resolution of radar data may be represented as a single point, could be sorted as clutter and ignored. Therefore, the authors decide to combine the information from both modalities in order to cover up each other's weaknesses, and propose a moving road user detection and classification model, which uses as input a single frame from each of the two modalities.

Initially, each radar-target velocity is compensated for the ego motion of the vehicle, thus obtaining the absolute velocity of the target. After filtering out the static or slow moving targets, a cube around the target's position is cropped out of the RAD tensor. This cropped cube is processed through a series of 3D convolutions, which squash the azimuth and range dimensions, with the aim of encoding the region around the target into a feature set only in the Doppler dimension. These features are then processed through a series of 1D convolutions, which downsample and concentrate the Doppler "signature" of the target, thus extracting class information. This resulting tensor is flattened and concatenated with the target level data: range, azimuth, Doppler, and RCS which are passed through fully connected layers that produce the final multi-class classification. After these steps, the results are classifications for each of the radar targets from the initial radar point cloud. In order to obtain detected targets, DBSCAN is applied for clustering the points from each class. By doing this after classification, different threshold parameters for clustering can be chosen for each class: euclidean distance, and distance in the velocity dimension. The

key insight from this work is how the information from both types of input data is used, in order to compensate for the lower-density and accuracy of the radar sensor.

2.4.2 Deep-Learning Kalman Filter Replacement

Pegoraro et al.[25] put forward a deep learning replacement of the Kalman filter. Its goal is to track a single target, a person, in an indoor scenario, using a statically mounted FMCW radar. The tracking problem is modelled as a sequence-to-sequence prediction problem, with the input being the radar data, and the output being the estimated position of the target and its uncertainty. This way of modelling the problem lends itself to a RNN-based approach, which is a popular model architecture used for sequence-to-sequence tasks. The format of the input data they are using is the RAD tensor, which is projected into the RD (range-Doppler), and respectively into the RA(range-azimuth) representations. Both input radar images are passed through their own multilayer CNN block in order distil their information and extract a one dimensional tensor. These two tensors are then concatenated to form the "compressed" observation. This observation is obtained in the same manner from the radar frames from all timesteps, thus forming the input sequence. The recurrent block they propose is made out of GRU[4] cells, which are a simplified version of the LSTM cell, that contains only the input and forget gates, and thus have fewer parameters. The output of each cell is a hidden state which is passed on as input to the subsequent GRU cell, together with the compressed observation resulted from radar images of the next timestep. The job of the GRU cell is to extract the information from the observation and encode it into the hidden state, thus transmitting the temporal information necessary for tracking. Each hidden state is also processed through 3 different fully-connected layers which output the state estimation, and two different types of uncertainty: aleatoric covariance, and epistemic covariance. The former reflects the innate noise of the state evolution - the target's movement, and of the measurement process - radar accuracy, while the latter represents the uncertainty in the prediction made by the model. These two covariances are combined to arrive to the standard covariance which is outputted also by the Kalman filter. The final model manages to get a 4x times better performance than the UKF (unscented Kalman filter) in both target position and target velocity, measured by the RMSE. These results highlight how a deep-learning approach can surpass traditional approaches even in traditional problems, such as single-target tracking, which has already been considered a "textbook" solution for a long time. The weakness of Kalman's assumptions is brought into the spotlight, which cannot model irregularity of simple human movement.

2.4.3 Tracking by Detection from the Radar Point Cloud

Tilly et al.[31] propose a tracking by detection model for FMCW radar point cloud data. The input data is manually labelled, with measurement to track associations made for all road users. Each point is preprocessed into cartesian components for the target position, and ego-motion compensated velocity, to which the RCS is

appended. The BEV (Bird's Eye View) projection of the point cloud is segmented into a grid, with the radar-points from each cell being processed by a PointNet++[26] network to extract a feature tensor. Consecutive radar sweeps, are processed in this way, and stacked into a tensor with dimensions number of timesteps, width and height of the grid, and feature dimension. This resulting tensor is passed through an modified version of a FPN (Feature Pyramid Network)[15] with the goal of extracting information from different scales, in order to increase the receptive field of the model. This output is passed to two branches: the confidence branch for predicting the probability for each cell of having a target, and the tracking branch which outputs the position in the cell, and the velocity of the supposed target. On both branches, 3D convolutions are used in order to merge the time dimension. The confidence branch is passed through a ResNet[10], a CN-based architecture with multiple residual connections, which at the end outputs the confidence heatmap. This output is concatenated with the output of the tracking branch, and they are passed through another ResNet to achieve the tracking branch's final output. The tracking branch is trained by calculating the RMSE(Root Mean Square Error) loss of the cartesian position and velocity components for each cell, while the confidence branch is trained using binary cross-entropy. During back-propagation, the errors from the tracking output are not propagated to the confidence branch in order to avoid contamination, but at the same time allowing the tracking to be conditioned by the confidence predictions.

For the second part of the tracking by detection model, the authors solve a LSAP(Linear Sum Assignment Problem) for associating detections with already existing tracks, creating new tracks, or deleting tracks with no recent detections. On each track, a linear Kalman filteris applied, which takes as input the previous supposed heading, and the new measurement - the fresh detection - to predict the position of the target in the next timestep. The major downside of this cell-based prediction approach is that there can be only one target per cell, which is not always the case when tracks intersect, for example when groups of pedestrians or of cyclists stay clumped together or separate into multiple groups. Through an ablation study, the authors demonstrate that not using the velocity information from the radar point cloud leads to a significant degradation of model performance. This insight suggests that the radial velocity provided by the radar is a key component to radar deep learning models, and should be exploited in order to compensate for the lower data-density of radar compared to the other two automotive sensors, lidar and camera.

2.5 Models used in Methodology

2.5.1 CenterNet: Detecting Objects as Center-Points

Zhou et al.[39] propose a novel, light-weight approach for the task of object detection in the field of computer vision. Their framework models objects as single center-points, from which other properties such as bounding box size can be regressed. This approach comes as an opposite to the then state-of-the-art object detection architectures, based on region proposal, anchors, or keypoint estimation, which were much more laborious. An overview of the model architecture can be seen in Figure 2.3.

Input images are passed through an encoder-decoder fully-convolutional backbone which generates a feature embedding for each pixel. From each feature embedding, the value of the heatmap and of other bounding box parameters, such as height and width, are each predicted by a regression head – a simple convolutional layer. Each pixel of the heatmap represents the confidence of an object having its center point at that location.

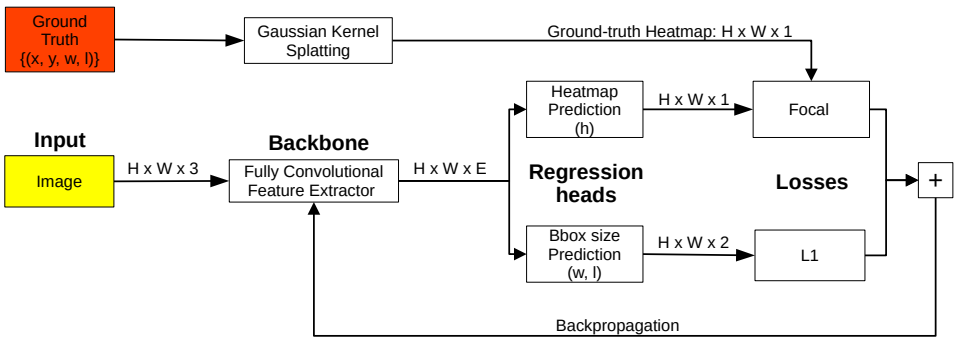


FIGURE 2.3: Simplified CenterNet[39] training architecture.

For obtaining the set of object detection predictions, a certain number of the top peaks from the heatmap are selected as object predictions – meaning that it is estimated that an object that has its center there. The size of the object – the height and width of its bounding box is simply the output of the height and width regression heads, applied to the same feature embedding, the one at the object's center point location. See Figure 2.4.

In order to train the model, it is necessary to generate ground-truth heatmaps from the ground-truth bounding box annotations. For for each image, the ground-truth heatmap is modeled as an overlapping of multiple 2D Gaussians, each corresponding to a ground-truth bounding-box annotation. Each Gaussian has a peak of 1, a size proportional to the corresponding ground-truth bounding-box size, and is centered at the ground-truth center-point of the object. In the case multiple Gaussians overlap, the value of the heatmap is simply the maximum value.

The model is trained end-to-end by using the focal loss[16], which is a modified version of the classic cross-entropy loss, but that has positive and negative examples weighted differently. In sparse signal situations such as the one for CenterNet, the center points – the positives, are very few in comparison with the size of the input image, thus the focal loss increases the weight for them. On the other hand, it reduces the penalty for the negatives – the background, which are much more abundant. Added to this loss, there are also vanilla L1 losses for the other regression heads' outputs - bounding box size and center point offset. These additional regression heads are supervised only at ground-truth center point locations, while the heatmap supervision is done across the whole image.

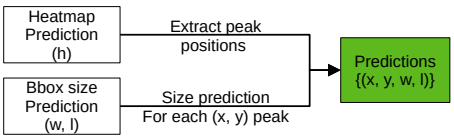


Figure 2.4: Prediction extraction from the CenterNet[39] regression heads.

The model can also be extended to also predict object class, simply by predicting multiple heatmaps, one for each class. This is done by adding extra regression heads, each corresponding to a different heatmap. Conversely, more bounding-box size regression heads are not added, as the initial set can still function to predict the height and width of the objects from the feature embedding, indifferent of class.

As such, CenterNet is a versatile framework that performs object detection, human pose estimation – by associating a class to each joint, and even 3D object detection, all at levels comparable with the then state-of-the-art. Besides the logical simplicity, the CenterNet architecture requires low computational cost and performs at a high frame-rate.

2.5.2 CenterTrack: Tracking Objects as Center-Points

Built on top of the CenterNet architecture by the same authors, CenterTrack[38] extends the model to a joint tracking and detection architecture with only a couple of simple modifications: instead of inputting a single image, two consecutive images are given to the backbone. Additionally, another regression head is added that predicts target displacement.

The principal idea from CenterTrack is to add a feedback mechanism between consecutive frames, that aids the model in performing temporal association across consecutive frames for tracking. This is done by extending the original CenterNet input by stacking the previous image together with a heatmap which is generated from the tracks of the previous timestep, see Figure 2.5. By adding the previous heatmap to the input, the current stage object detections are conditioned on the previous stage tracking results. Thus, the model input which comprised only of the current image for CenterNet, receives four extra input channels: red, green,

blue – from the previous image, and one extra channel for the previous heatmap for CenterTrack. An example of a CenterTrack input tensor can be seen in Figure 3.9A. For comparison, Figure 3.8A shows a simple RGB image, the CenterNet model input tensor.

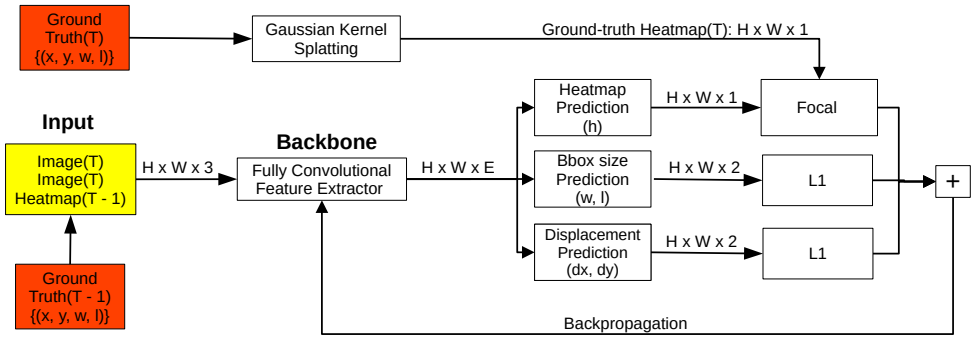


FIGURE 2.5: Simplified CenterTrack[38] training architecture.

During training, the ground-truth heatmap of the previous frame is used. During inference, to propagate only relevant information, only the predictions from the previous stage with a confidence (heatmap prediction) higher than a certain threshold are used to generate the feedback heatmap. The heatmap is generated with the same Gaussian splatting kernels as in CenterNet. By forming the model input this way, the backbone is tasked to do the heavy-lifting, by generating a rich enough feature embedding also containing temporal information, from which it is possible to predict target displacement.

To predict target movement, an additional output regression head that estimates the displacement of the center points of the targets between the two consecutive frames is added. Previous tracks, together with the current frame detections and their estimated displacements are given to a greedy track association algorithm that generates tracks ids, see Figure 2.6. By inverting the displacement estimation and adding it to the center point of the current detected predictions, the current center points can be associated to the detections from the previous timestep in a greedy fashion. In reverse order of confidence, current predictions are matched to the closest previous predictions, while taking into account the object size. Matched predictions are marked with the same track id, to propagate the target track. Current predictions which remain without a match are considered the start of new tracks and are given a new track id, while previous tracks that are not continued by a new match are simply lost.

The model is trained identically to CenterNet, end-to-end, with the focal loss for the heatmap prediction, and with L1 supervision only at ground-truth center-point positions for the other regression heads, including the displacement head.

Data augmentation techniques are used to make the model more robust and to simulate potential errors. Adding, removing, or jittering heatmap peaks helps

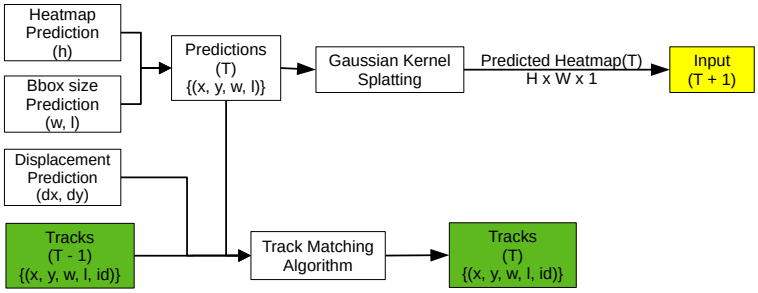


Figure 2.6: Formation of track predictions during inference. Uses the outputs of the regression heads and the tracks from the last timestep. The estimated heatmap used in the following inference step is generated from the current predictions.

to bulletproof against false positive tracks, occlusions, and center point position prediction errors respectively. Additionally, the model is not only with consecutive frames, but also with images that are a couple frames apart. A key observation is that the backbone manages to learn these complex scenarios only through data augmentation, and there is no need for additional heuristics to be implemented on top of the greedy snapping algorithm.

CenterTrack manages to perform the more complex tasks of 2D MOT and monocular 3D MOT tracking with only these simple additions to the CenterNet detection model, while preserving its simplicity and low computational complexity.

2.5.3 Swin Transformer

The Attention Mechanism

The Transformer[33] is a deep learning architecture, originally developed as an alternative to RNNs, that were then the main choice for sequence-to-sequence prediction tasks in the field of Natural Language Processing (NLP). The Transformer was originally an encoder-decoder architecture, but encoder-only and decoder-only architectures have also found their use. Transformers have been adopted in multiple fields of the deep learning domain: in NLP for text generation and machine translation, in Computer Vision (CV) as Vision Transformers (ViT), or for image generation, and for multi-modal applications such as captioning images or audio-video analysis, just to name a few.

The main advantage of the Transformer is its unmatched potential for parallelization, which has at its core the attention mechanism. By applying an attention layer, sets of embeddings – vector representations of the basic units of the respective input modalities: words in NLP, regions of pixels in CV, or waveforms in the audio domain, are processed in relation to other embeddings. Self-attention is employed to extract deeper understanding and relationships from a sequence by relating it to itself. Example use cases would be image feature encoding, or text understand-

ing. Standard attention on the other hand is mostly used when the task requires extracting relationships between two different sources, such as question answering, or cross modal-tasks such as image captioning. Nevertheless, both attention and self-attention are computed using the following formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where Q represents the queries, K the keys, and V the values. The formula computes a weighted combination of the values, based on the similarity between keys and queries. The $softmax$ function normalizes the score in order to compute the value weights, while $\sqrt{d_k}$ is used as a scaling factor to stabilize the gradients. The size of the queries and keys is d_k and the size of the values is d_v. The Q, K, and V matrices are obtained by applying a linear transformation to the embeddings X from the sequence of interest:

$$Q = XW^Q; K = XW^K; V = XW^V$$

The W^Q, W^K, and W^V matrices of shapes $d_{model} \times d_k$, $d_{model} \times d_k$, and $d_{model} \times d_v$ are actually the learnable parameters of the attention layer, where d_{model} is the input embedding size at that specific layer. When standard attention is performed, the queries are obtained from the embedding sequence that attends while the keys and the values are obtained from embedding sequence that is attended to. In the self-attention case, queries, keys, and values are all generated from the same sequence, that attends to itself. In practice, these linear transformations are implemented using a single fully-connected layer.

An extension to the attention mechanism which increases its robustness while maintaining the same computational complexity is the Multi-Head Attention. In it, multiple – h smaller query, key, and value matrices are used in each attention layer: W_i^Q, W_i^K, W_i^V with $i = \{1 \dots h\}$. These matrices have the first dimension h times smaller than their equivalents in the standard attention. In this way, the attention layer becomes more versatile, by permitting each head to focus on learning relationships from different positions of the original d_{model}-sized embedding.

Swin Transformer

The Swin Transformer[17] is a ViT architecture designed as a feature encoder for images, excelling in capturing multi-scale visual information. Unlike traditional convolutional networks, which use fixed-size filters, the Swin Transformer employs self-attention mechanisms at multiple scales, enabling it to capture both fine-grained and large-scale features. This architecture is composed of multiple stages, where each stage extracts increasingly abstract representations of the image, progressing from local to more global features.

Each stage in the Swin Transformer is made up of several Swin Transformer blocks. Within each block, two types of self-attention are applied: windowed multi-head

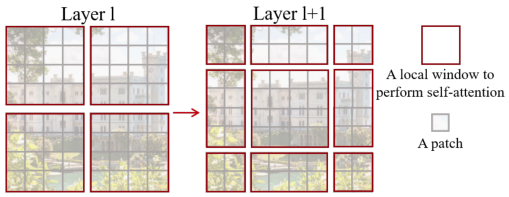


Figure 2.7: Windowed and Shifted Window attention. Image taken from [17].

self-attention (W-MSA) and shifted window multi-head self-attention (SW-MSA). The windowed attention divides the image into small, non-overlapping windows, allowing the model to focus attention locally within each window, which makes the computation more efficient compared to other vision transformers. The MSA is applied only between the features inside of each window. The shifted window attention then shifts each window diagonally by half its height and width, creating overlap between neighboring windows. This shift allows information to flow across windows, effectively connecting patches that were previously isolated and enabling a stronger understanding of spatial relationships. A visualization of the shifted window mechanism can be sen in Figure 2.7.

After several Swin Transformer blocks, a patch merging layer reduces the spatial resolution of the image by merging adjacent 2x2 patches, effectively decreasing the number of patches while increasing the feature dimension. Swin Transformer blocks from the following stage maintain the same attention window size, but now they are practically acting on a 4 times larger area of the image, because of the patch merging, see Figure 2.8. This process lowers computational complexity while expanding the receptive field, allowing subsequent blocks to capture information at increasingly larger scales.

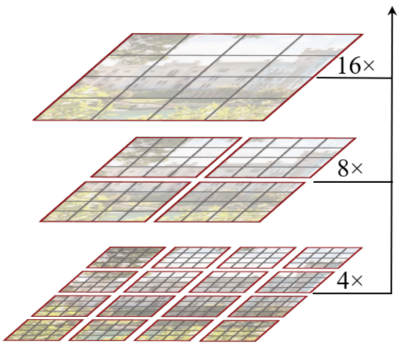


Figure 2.8: Multiple stages of patch-merging in the Swin Transformer. Image taken from [17].

As a result, the Swin Transformer progressively builds a multi-scale feature hierarchy, which is especially useful for tasks requiring both local detail and high-level context, such as object detection.

Chapter 3

Proposed Detection and Tracking Models

This chapter presents the architecture of the proposed detection and tracking models: Radar CenterNet-Swin and Radar CenterTrack-Swin. They are based on the core architectures of CenterNet[39] and CenterTrack[38] respectively, which are adapted to the radar scenario, by replacing their backbone with a modified version of the Swin Transformer[17].

The first section discusses the structure of the nuScenes dataset[3] and the proposed representation of the radar point cloud, compatible with the Swin Transformer. Following sections detail the necessary modifications to the CenterNet and CenterTrack that make them applicable them to the radar case.

3.1 Dataset

3.1.1 NuScenes Dataset

The nuScenes dataset comprises of 1000 (850 training and validation, 150 testing) 20-second scenes of urban and inter-urban driving from Boston and Singapore. The data is collected with three types of sensors: RGB-camera, lidar, and radar, across a variety of weather patterns and road conditions in order to form challenging scenarios. The objects in the scenes are annotated by human experts at a frequency of 2Hz with 3-dimensional bounding boxes, each characterized by 3-dimensional pose and rotation, object class (23 classes), and object status (parked, stopped, moving). Scenes which include rarer classes are manually chosen to be more frequent, in order to equilibrate class distribution.

There are five FMCW radar sensors of the model Continental ARS 408-21 [27] placed on the vehicle, as can be seen in Figure 3.1: one facing to the front, on the hood of the car, two facing sideways, near each lateral mirror, and the last two facing backwards, at the back of the car, near the brake lights. As the field of view in the far-range domain is small: 18°, and therefore the five radars cannot cover the whole 360-degree vehicle surrounding, we will be interested only in the near-range mode of

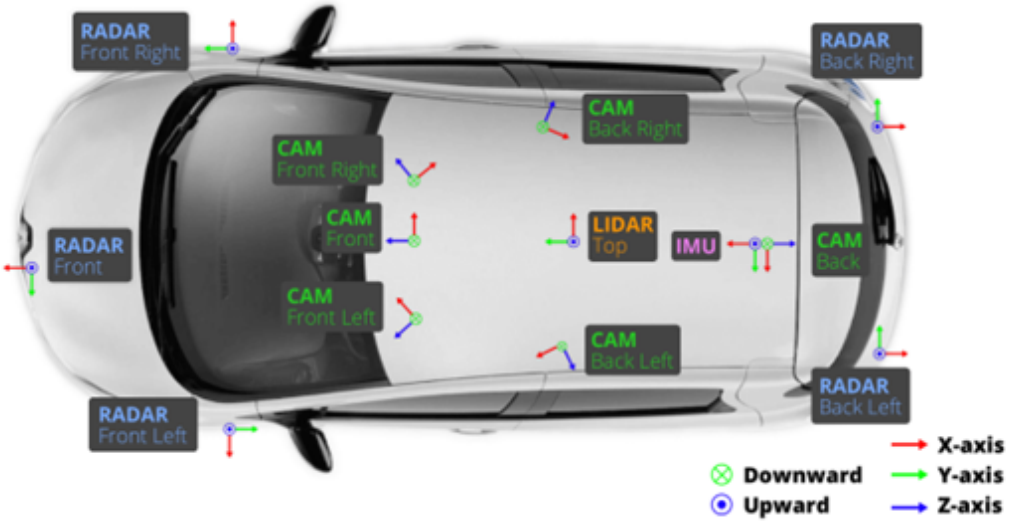


Figure 3.1: Radar positioning on the nuScenes vehicle as described in [3]

the radars. The specification of the sensor can be seen in Table 3.1.

Parameter	**Value**
Distance range	0.20 ... 70m @ ±45° / 0.20 ... 20m @ ±60°
Resolution distance measuring	0.39 m
Accuracy distance measuring	±0.10 m
Azimuth angle augmentation	-60° ... +60°
Resolution azimuth angle	3.2° @ 0° / 4.5° @ ±45° / 12.3° @ ±60°
Accuracy azimuth angle	±0.3° @ 0° / ±1° @ ±45° / ±5° @ ±60°
Velocity range	-400 km/h ... +200 km/h
Velocity resolution	0.43 km/h
Velocity accuracy	±0.1 km/h

Table 3.1: ARS 408-21 performance in the near range mode [27]

Each of the five radars capture data at 13Hz frequency. The radars are not synchronized perfectly between each other, and they are not synchronized with the human-expert ground-truth annotation keyframes either, as they were lined up with the lidar and camera data. Thus, the sensor frames from each radar which are closest time-wise to the ground-truth annotation timestamp are considered samples, and are approximated as being captured at the same timestamp, and have ground-truth available. For each radar, between two consecutive samples, there are usually 5-7 other radar frames, which don't have ground-truth annotations available, and shall be named sweeps. In Figure 3.2 the temporal alignment of the sweeps from the five radars can be seen, with samples represented by black lines, and sweeps by dots.

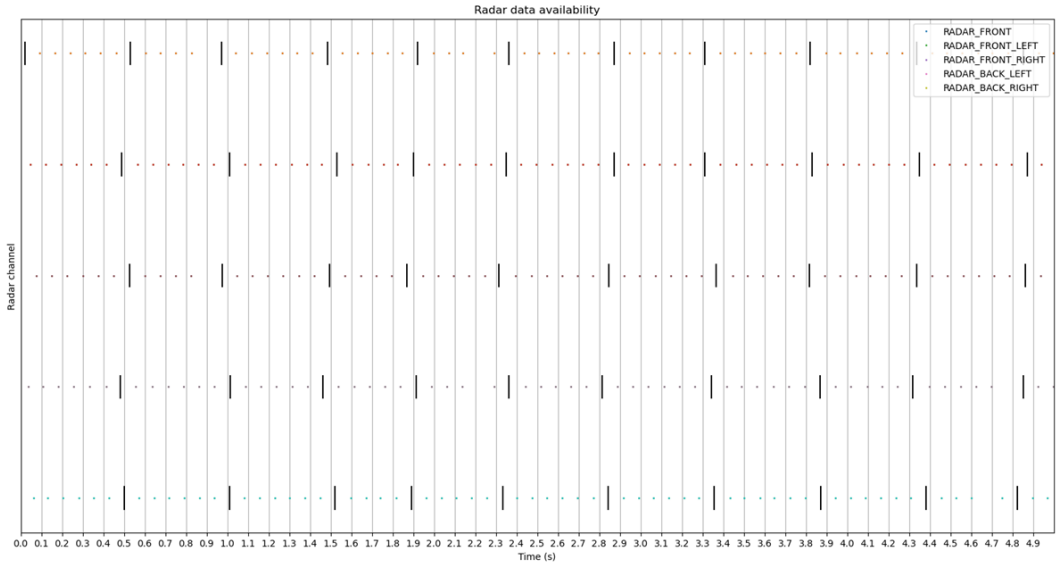
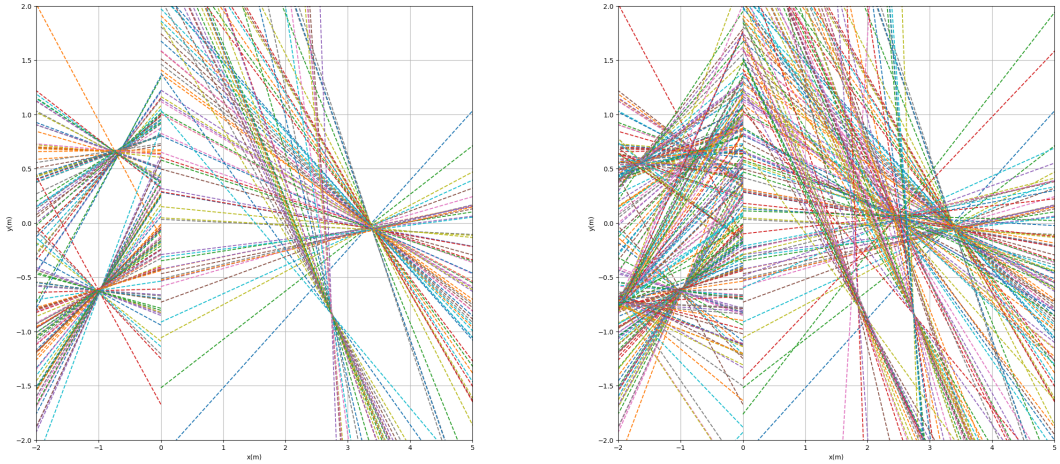


FIGURE 3.2: Radar sweeps time alignment

3.1.2 Dataset preprocessing

Each radar frame is represented as a set of points, each being described by an angle, range, radial velocity, and RCS (radar cross section). It is important to note that the radial velocities are compensated by the ego-vehicle translation and rotation velocities. Thus, the radial velocities are just the radial components of the absolute (world coordinates) velocities of the target objects. It is important to note that we are using only the magnitude of the radial velocity, this means that we are approximating – not taking into account the positions of the radars on the vehicle. In other words, we are approximating the ego-vehicle as a point, only in the aspect of radial velocities. The extended support lines of the radial velocities from a single 360-sweep in top-down view can be seen in Figure 3.3a. The front left radar has not detected any points in this case. The 4 visible intersections coincide with the positions of the radars on the vehicle, as can be seen in Figure 3.1. In Figure 3.3b two consecutive sweeps can be seen, while the ego-vehicle is moving forwards (positive x direction).

The point clouds from the five radars, $R_f, R_{fl}, R_{fr}, R_{bl}, R_{br}$ - front, front-left, front-right, back-left, back-right, from a sample timestamp t are combined by simply aggregating the point clouds, which are sets of points. In order to do the same for the sweeps, starting from each sample, for each radar, we go back in time through the sweeps, approximating that the five radars are synchronized, and aggregating them as well, for sweeps $t-1, t-2, t-3 \ldots$. This is done starting from each sample only for N_s sweeps backwards as in Formula 3.1. We are using $N_s = 6$ because there are usually between 5 and 7 sweeps between consecutive samples. A sample is represented by $Sw(t)$ while its previous sweeps as $Sw(t-i)$.



(A) Radial velocities extensions, 1 sweep (B) Radial velocities extensions, 2 sweeps

Figure 3.3: Comparison of radial velocities extensions for 1 and 2 sweeps.

$$\begin{aligned} \mathbf{R} &= \{f, fl, fr, bl, br\} \\ R_x(t) &= \{(\theta, r, v_r, rcs)\} \qquad\qquad x \in \mathbf{R} \\ Sw(t-i) &= \bigcup_{x \in \mathbf{R}} R_x(t-i) \qquad i = 0 \dots N_s - 1 \end{aligned} \tag{3.1}$$

These 360-degree radar sweeps, which are currently point clouds, arbitrarily-sized sets of points, are further transformed into a fixed-sized format, in order to be compatible with the Swin Transformer backbone. To do this, the full 360 degree angle space is divided into equal sized angle bins N_{ab}. If multiple points from a sweep land in the same bin, only the closest one is taken, the rest are discarded. This is in line with the functioning of the radar, as we are more interested in the points coming from closer targets, which have a lower chance of becoming occluded in the future frames. An example of a radar sweep processed in such a way can be seen in Figure 3.4. The blue points represent the result of the angle bin quantization, while the points which are lost due to bin collisions are in yellow. All remaining empty bins are filled with zeros. As such, a 360 degree sweep can be represented as a $1 \times N_{ab} \times 3$ (r, v_r, rcs) input tensor. By stacking a sample with the N_s sweeps before it, we manage to obtain a $N_s \times N_{ab} \times 3$ tensor, approximately representing the radar information from the past 0.5 seconds. This is in line with the annotation frequency of nuScenes, 40 annotated frames in 20 seconds. In this way, the radar data for a whole scene is processed into 40 samples of $6 \times N_{ab} \times 3$ (r, v_r, rcs).

Aggregating the sweeps can be done in two ways, either in absolute frame of reference, or in ego-vehicle or relative frame of reference. The nuScenes dataset is already preprocessed and offers the compensated radial velocity readings. This means that they are compensated for ego-vehicle translation and rotation, and thus are in the absolute frame of reference. In such case, it would make sense to also stack the N_s sweeps in absolute coordinates. For example, the readings of a target vehicle,

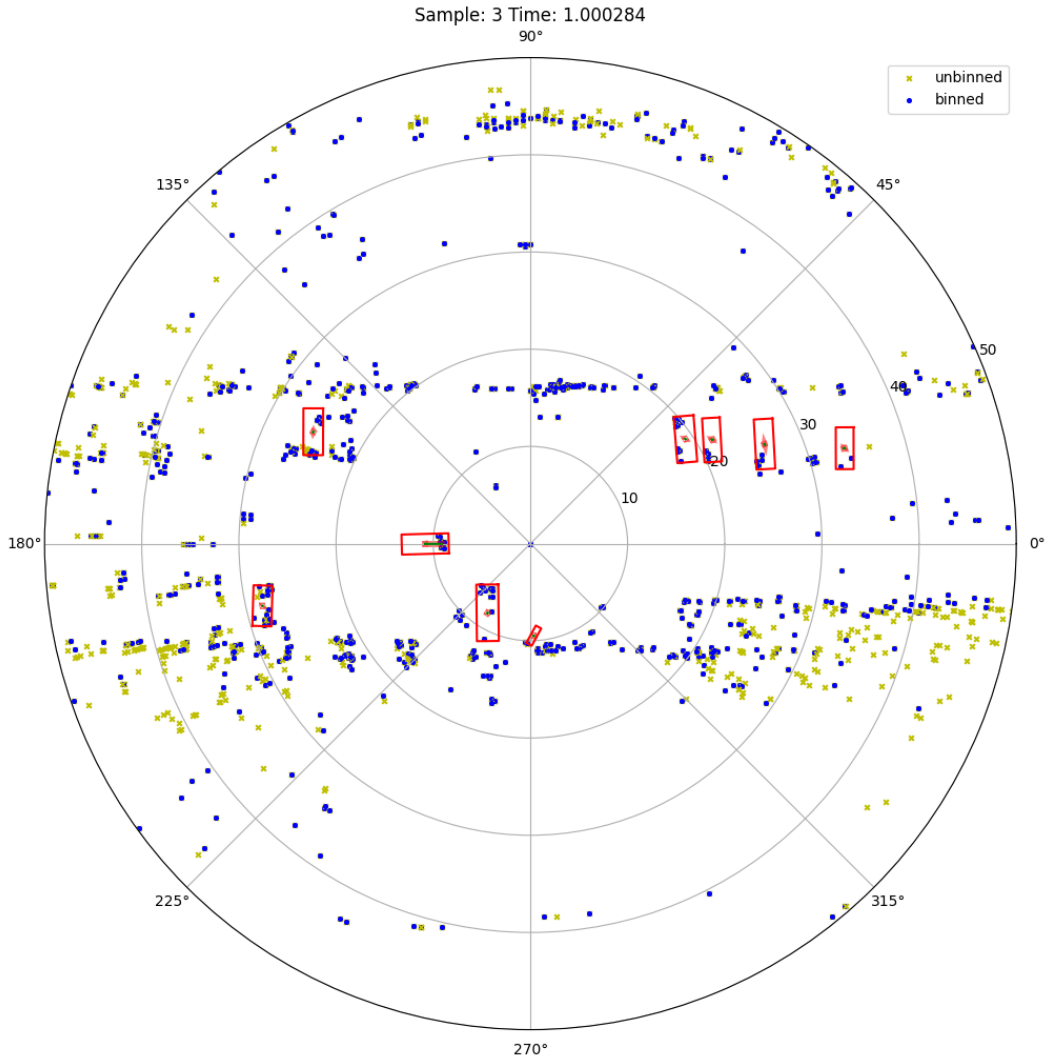


FIGURE 3.4: A processed input data sample with $N_{ab} = 360$ angle bins, comprising of $N_s = 6$ stacked sweeps. Binned points during the quantization procedure are in blue. Yellow points are the radar readings that are dropped due to bin collisions. Ground-truth targets are represented in red. The image shows that the quantization does not lose readings that come from relevant targets, but mostly the background ones.

which is static in absolute coordinates and which had a radar reading in each of the N_s sweeps, would appear as multiple overlapped points when aggregating in absolute coordinates, regardless of the movement of the ego-vehicle. Conversely, moving targets would leave behind tracklets, whose length depend only on the absolute (compensated) velocities of the targets. After being processed and stacked in absolute coordinates, the points are brought back to the coordinate frame of the position of the ego-vehicle at the most recent sweep. This means that the range r and angle θ of

each point from the initial radar sweeps $Sw(t) \dots Sw(t - N_s + 1)$ will be represented in the coordinate frame of $Sw(t)$, while their actual positions reflect the positions of the targets in absolute coordinates. Conversely, the radial velocities v_r and radar cross sections rcs remain unaltered. They each represent the compensated v_r and the rcs read by the radars during $Sw(t - i)$ when the car was at a different position than at $Sw(t)$.

The ground truth bounding-box annotations – which correspond to each of the 40 samples, are each described by a center point: (θ, r), velocity, split into the radial and tangential components: (v_r, v_t), size, the width and length of the 2D top-down projected bounding box: (w, l), and relative orientation: θ_o which represents the difference between the rotation angle of the bounding box, and θ the angle of its center point in polar coordinates. The variables describing a bounding box can be seen in Figure 3.5.

All sweeps are represented in the coordinate frame of the ego-vehicle at the sample timestamp $Sw(t)$. The ego-vehicle is always positioned in the center of the image, and the vehicle's orientation is always towards $\theta = 0$, with θ increasing counter-clockwise, up until it covers the whole 360° surrounding. The same is true for ground-truth annotations as well, as they are available only for samples.

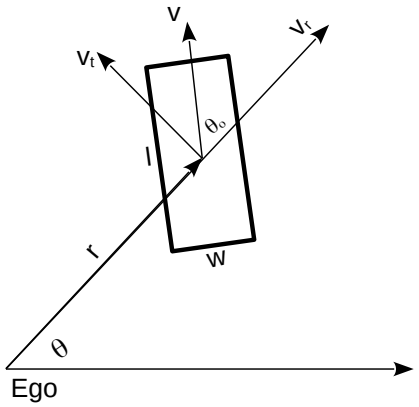


Figure 3.5: A bounding-box annotation

3.2 Radar CenterNet-Swin Detection Model

The proposed model is inspired by advances in the computer vision domain, particularly building on the principles of CenterNet [39] and its extension, CenterTrack [38]. These models, which are originally developed for object detection and tracking in image data, are modified in order to adapt them to our radar frame inputs. In our approach, The original CenterNet backbone is replaced by a Shifted Window (Swin) Transformer [17], a powerful transformer-based architecture known for its strong performance in visual tasks. The Swin Transformer has been specifically modified to better suit our radar data, allowing for multi-scale temporal feature extraction.

The detection model receives as input a processed radar sample: $N_s \times N_{ab} \times 3$ (r, v_r, rcs) and outputs a set of predicted bounding boxes. An overview of the model architecture can be seen in Figure 3.6.

3.2.1 CenterNet modification

As presented in Subsection 2.5.1, CenterNet predicts a 2D heatmap, which represents the confidence of having a keypoint - an object center point at each pixel location. In our radar scenario, our predicted heatmap shall be 1-dimensional: $1 \times N_{ab}$ - each cell represents the confidence of having an object in that specific angle-bin. Similar to CenterNet, besides the heatmap, additional regression heads are used to predict the range and the size of a potential object for each angle bin. In order to achieve this in our radar scenario, the backbone is replaced with a modified version of the Swin Transformer[17].

The modified Swin Transformer backbone receives as input a $N_s \times N_{ab} \times C$ tensor, and outputs a $1 \times N_{ab} \times E$ tensor, where E is the final feature dimension. Similar to CenterNet we use multiple regression heads: the main one is for the heatmap prediction, and secondary ones for other required predictions: range, radial and tangential velocities (v_r, v_t), bounding-box size (w, l), bounding-box orientation $(sin\theta_o, cos\theta_o)$. As an observation concerning the regression heads, logically connected values share the same input weights but different output weights. Examples of connected values are the pairs representing velocities, bounding box size, and bounding box orientation. In order to obtain the final orientation of each target θ_o, the inverse of the tangent function is used on the outputs of the respective regression head. The rotation was chosen to be represented through the *sine* and *cosine* of θ_o as this is a continuous representation of the full 360° 2D rotation space. Using this representation, the discontinuity between 0 and 2π is avoided, and rotation becomes easier to learn for a neural network. The advantage of using such a representation is demonstrated in [40].

Each of these regression heads is actually a single feed forward layer, acting on the E-sized feature from each of the N_{ab} angle bins. This output is then passed through an activation function, tailored to the type of regression head: *sigmoid* for values between 0 and 1, such as the heatmap, *ReLU* (Rectified Linear Unit) for unbounded positive values such as range and size predictions, and *tanh* (hyperbolic tangent) for values between -1 and 1, such as orientation and velocities. The bounding box

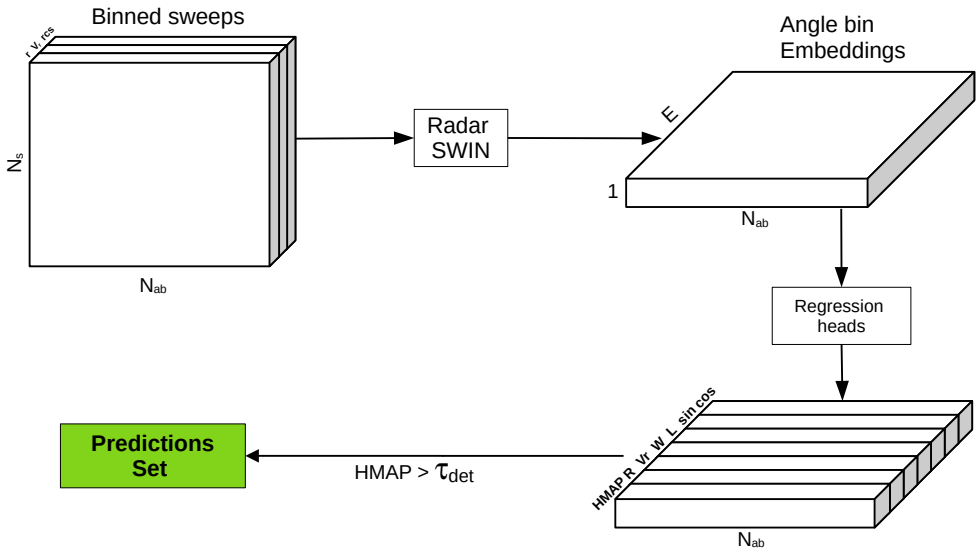
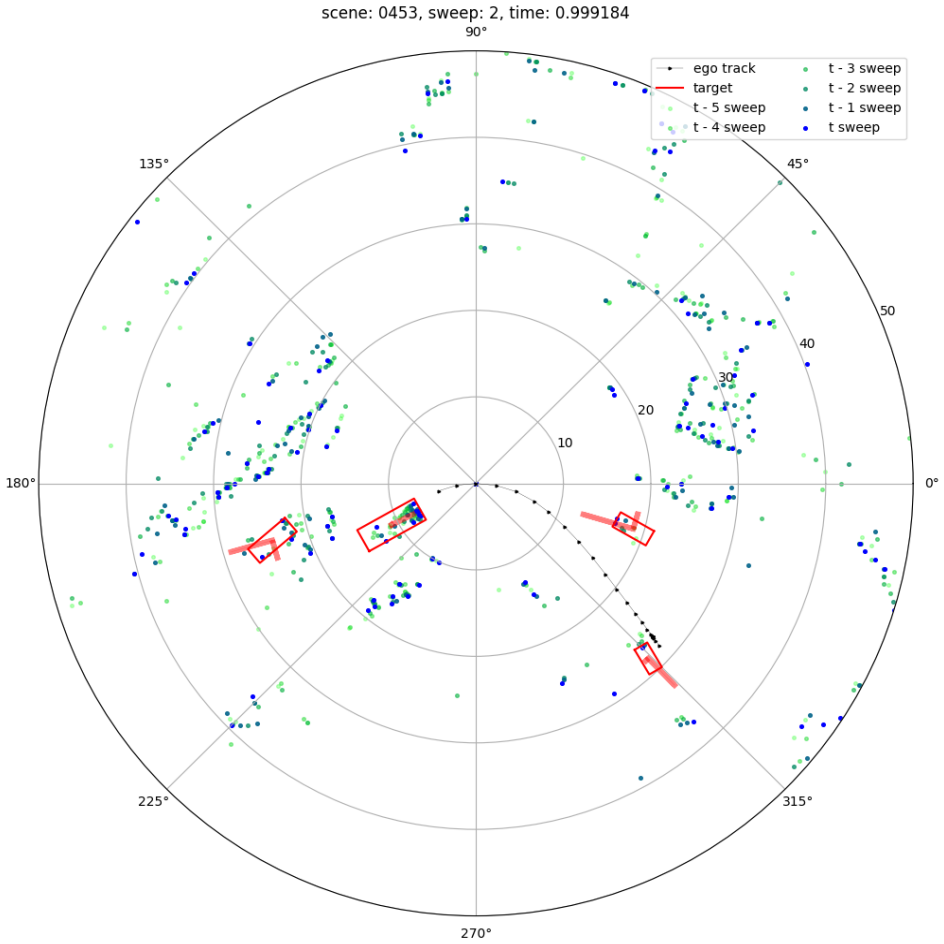


Figure 3.6: Radar CenterNet-Swin detection model architecture inspired by CenterNet[39]. Compared to CenterNet, the backbone is replaced with a modified version of the Swin Transformer[17]. The regression heads are modified to predict the bounding-box parameters necessary for radar detection.

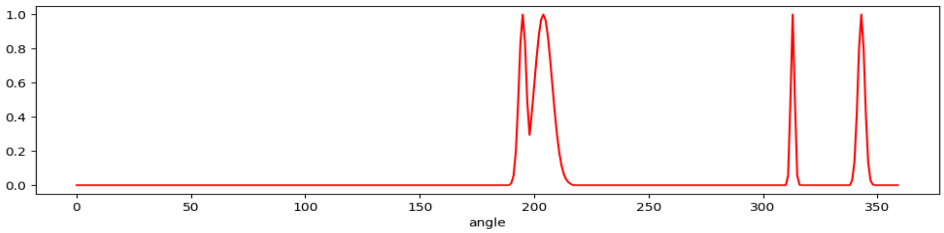
sizes (w, l) and the velocities (v_r, v_t) are normalized by the range r in order to guide the backbone to learn a general representation for the objects. For example: an object of a certain size, at a certain distance away from the ego-vehicle will generally receive the same amount of radar readings, spread over the same number of angle bins, as an object twice as big, which is twice as far away. Similarly, to motivate the velocities normalization, the radar track over the N_s sweeps of an object with a certain tangential velocity will cover the same amount of angle-bins as the same object if it was twice as fast, and twice as far away.

In order to generate the ground truth heatmap for training, from each ground truth bounding box we have to generate a gaussian on our 1-dimensional angle bin representation. From each bounding box, the 2 furthest corners (largest angle difference between them) are selected. A gaussian with the center at the middle point of the segment, and covering the whole segment with $+-3\sigma$ (standard deviation) is generated. As in CenterNet, the maximum value from overlapping gaussians is selected obtaining the $1 \times N_{ab}$ heatmap. An example of a processed radar sample which includes $Ns = 6$ sweeps, and its associated heatmap can be seen in Figure 3.7.

At inference time, the heatmap peaks, meaning the angle-bins in which the heatmap prediction is larger than both neighbors, and which are over a certain cutoff threshold τ_{det} are extracted and considered the model's predictions for the θs of the target center points. As in CenterNet, the outputs of the other regression heads for



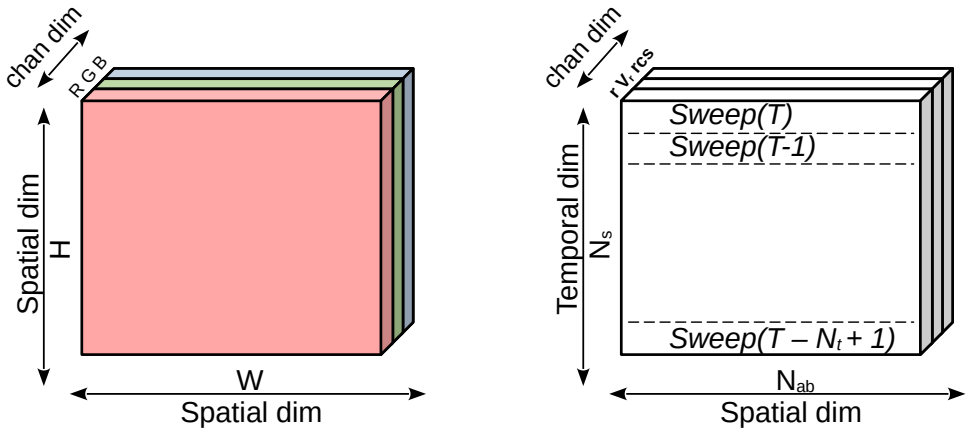
(A) Fully processed radar sample



(B) Associated ground-truth heatmap

FIGURE 3.7: Input data-ground truth pair example.

the angle-bins for which there is a peak in the predicted heatmap are taken in order to complete the information about the bounding boxes. As such, a set of predictions has the following format: $(\theta, r, v_r, w, l, \theta_o)$.



(A) Camera image representation $H \times W \times C$

(B) Radar sample representation $N_s \times N_{ab} \times C$

Figure 3.8: Comparison between CenterNet camera image and Radar CenterNet-Swin input tensors

3.2.2 Swin Transformer Modification: Radar Swin

In order to adapt the Swin Transformer to function as a backbone in the CenterNet-style architecture, a series of modifications are made, to adapt it to our radar data input format. The Swin Transformer was originally designed for square image inputs – as is mostly the case in the vision domain. The modifications extend its versatility, enabling it to process non-square data formats and thus making it suitable for radar sweep data.

Preprocessed radar sweeps, when represented in image-like formats, are structured as $N_s \times N_{ab} \times C$, where N_s is the number of sweeps (temporal dimension), N_{ab} is the number of angular bins (spatial dimension), and C represents the number of channels. In this setting, the radar sweeps can be perceived as an image with height N_s, and the width N_{ab}, as shown in Figure 3.8.

One major adaptation is the introduction of variable-sized attention windows, with independently adjustable height and width for each stage of the model. This flexibility allows the attention mechanism to consider different scales of information along the temporal and angular dimensions as needed. By tuning the window size at each stage, the model can focus on short-term temporal patterns at one stage and broader angular relationships at another. Additionally, variable-sized patch merging factors enable the model to reduce each dimension independently, allowing for down sampling that aligns with the specific structure of radar sweeps. This controlled reduction of spatial and temporal resolution helps preserve the most relevant features across dimensions while keeping the computational load manageable.

Furthermore, as the N_s dimension represents time, the modified Swin transformer

doubles as a temporal relational layer as well, by associating the sweeps not only through space but also through time. Therefore, a temporal attention mask is introduced to ensure that information from past sweeps remains temporally isolated from future sweeps. This is particularly important for radar data, where causality must be preserved; each frame should only have access to data from prior frames to ensure accurate and realistic feature extraction.

As a result of these modifications, the adapted Swin Transformer outputs a feature embedding of size $1 \times N_{ab}/X_{ab} \times E$, where X_{ab} is the total multiplied reduction factor on the angle-bin dimension, which is the result of passing through all patch-merge stages of the modified Swin Transformer backbone. This embedding provides a compact yet informative representation of the radar sweeps, preserving essential spatial and temporal features for subsequent analysis or prediction tasks. The final embedding is further passed to the previously described CenterNet regression heads to extract the model predictions.

3.2.3 Losses

For training supervision, we follow a strategy similar to CenterNet[39], employing the penalty-reduced pixel-wise logistic regression with focal loss[16] to guide the heatmap prediction. In Formula 3.2 for the total heatmap loss, $\hat{H}$ represents the predicted heatmap for a sample, H the ground truth heatmap, and N represents the number of ground-truth annotations associated to the sample. Consequently, $\frac{1}{N}$ is a normalization factor to bring the sum of all positive instances to 1. The loss parameters $\alpha = 2$ and $\beta = 4$ are chosen the same as in CenterNet.

$$L_H = -\frac{1}{N} \sum_{i \in \{0 \ldots N_{ab}\}} \begin{cases} \left(1 - \hat{H}_i\right)^{\alpha} \log\left(\hat{H}_i\right) & \text{if } H_i = 1, \\ (1 - H_i)^{\beta} \left(\hat{H}_i\right)^{\alpha} \log\left(1 - \hat{H}_i\right) & \text{otherwise.} \end{cases} \tag{3.2}$$

Additionally, the regression head predictions are supervised only at ground truth heatmap peaks, with an L1 loss applied to the parameters r, v_r, w, l, and $sin\theta_o, cos\theta_o$, corresponding to the object's center range and radial velocity, and the bounding-box width, length, and orientation respectively. In the Formulas 3.3, k goes through the ground-truth peak positions - the angle bins of the target objects' center points. The values for v_r, w, l are multiplied back with the range prediction in order to penalize the errors in absolute measures. This is done because during preprocessing, these values were normalized by the range r in order to facilitate the backbone to learn a general representation.

$$
\begin{aligned}
L_r &= \frac{1}{N}\sum_{k=1}^{N}|\hat{r}_k - r_k| \\
L_{v_r} &= \frac{1}{N}\sum_{k=1}^{N}|\hat{r}\cdot\hat{v_r}_k - r\cdot v_{rk}| \\
L_w &= \frac{1}{N}\sum_{k=1}^{N}|\hat{r}\cdot\hat{w}_k - r\cdot w_k| \\
L_l &= \frac{1}{N}\sum_{k=1}^{N}\left|\hat{r}\cdot\hat{l}_k - r\cdot l_k\right| \\
L_{sin\theta_o} &= \frac{1}{N}\sum_{k=1}^{N}\left|\hat{sin}\theta_{ok} - sin\theta_{ok}\right| \\
L_{cos\theta_o} &= \frac{1}{N}\sum_{k=1}^{N}\left|\hat{cos}\theta_{ok} - cos\theta_{ok}\right|
\end{aligned}
\tag{3.3}
$$

To balance the relative importance of these regression targets with heatmap prediction, the L1 losses are scaled by a factor of $\lambda = 0.1$, as done in CenterNet, which reduces their influence compared to the focal loss. This brings the final loss signal to Formula 3.4.

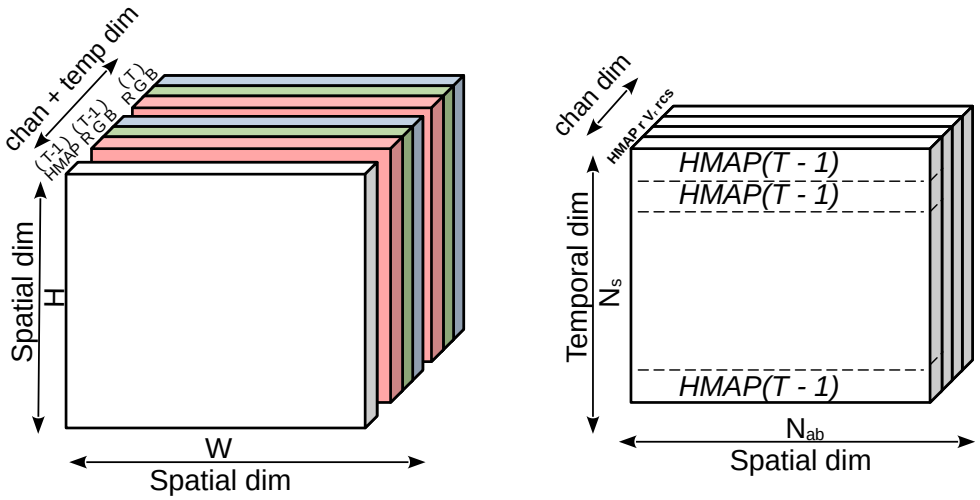
$$
L_{det} = L_H + \lambda\left(L_r + L_{v_r} + L_w + L_l + L_{sin\theta_o} + L_{cos\theta_o}\right) \tag{3.4}
$$

3.3 Radar CenterTrack-Swin Tracking Model

3.3.1 CenterTrack modification

The model is extended to tracking using the same logic as CenterNet is extended to CenterTrack[38]. In the CenterTrack case, as presented in Subsection 2.5.2 the input to the model is comprised of two consecutive images: $Image(T)$ and $Image(T-1)$, and the predicted heatmap: $Heatmap(T-1)$. These are concatenated on the channel direction, and thus the same backbone architecture as in CenterNet can be used, which instead of receiving a single image input of $H \times W \times 3$ (R, G, B), receives a $H \times W \times 7$ tensor. This is the result of stacking the two RGB images, with 3 channels each, with the heatmap which is single channel. Furthermore, the feature encoding outputted by the backbone is also processed by an additional regression head which predicts the displacement of a center point between the two consecutive frames T and $T-1$.

In order to convert this to the radar case, there are two aspects which need to be dealt with. The first one is the fact that the dimensions of the preprocessed radar samples are not equivalent to image dimensions. The second one the fact that the dimensionality of the prediction heatmap is not the same as the input radar sample dimensionality.



(A) Camera image representation $H \times W \times (2C + hmap)$

(B) Radar sample representation $N_s \times N_{ab} \times (C + hmap)$

FIGURE 3.9: Comparison between CenterTrack and Radar CenterTrack-Swin input tensors. Each "channel" of the tensors is annotated.

In the image case, both the height H and the width W represent spatial dimensions, while in the radar case, our preprocessed samples are represented by $N_s \times N_{ab} \times 3$ (r, vr, rcs) tensors, which have the spatial dimension N_{ab} but also the temporal dimension N_s. In the image case, by stacking $Image(T)$ and $Image(T-1)$,

new meaning is added to this channel dimension, by giving it temporal information as well. As a radar sample already includes temporal information, it is enough to use $Sample(T)$ as an input to the model.

Concerning the heatmap, in our radar representation, a heatmap is of the format $1 \times N_{ab}$. In order to keep the data shape and thus modified Swin backbone similar to the detection case, we simply add the heatmap as an extra channel to the point cloud data, besides the usual 3 channels (v, v_r, rcs). In order to do this, we copy $Heatmap(T-1)$ and stack it N_s times, to get a $N_s \times N_{ab}$ tensor. This is finally stacked with the processed point cloud, which results into a tensor shaped $N_s \times N_{ab} \times 4(\mathbf{hmap}, v, v_r, rcs)$. A visualization of the CenterTrack and Radar-Swin-CenterTrack input formats can be seen in Figure 3.9.

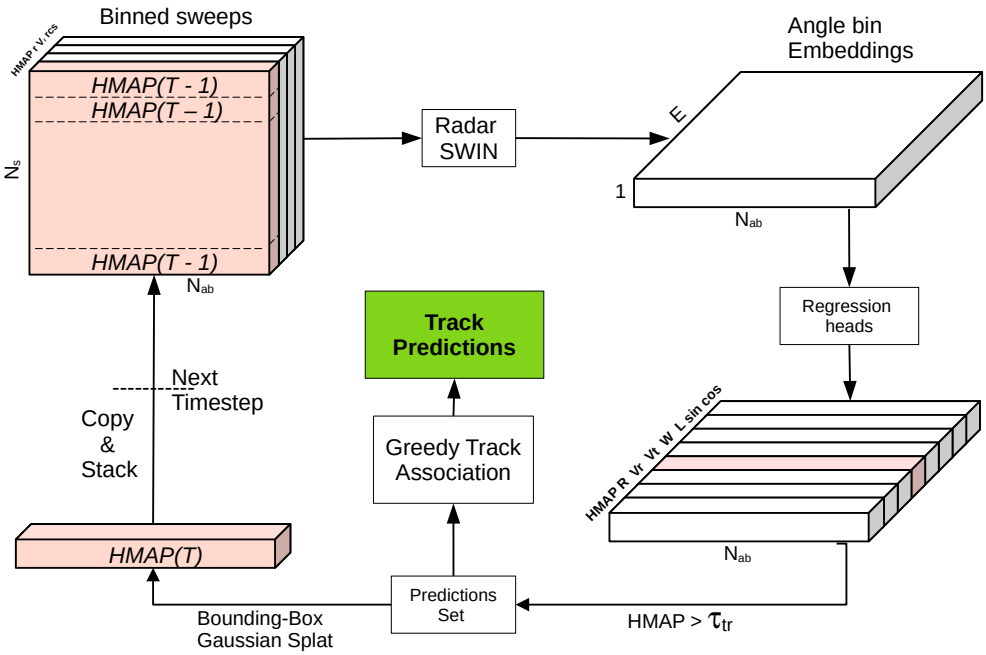


Figure 3.10: Radar CenterTrack-Swin tracking model architecture inspired by CenterTrack[38]. Key additions compared to Figure 3.6 are highlighted in pink: an extra regression head for predicting the tangential velocity v_t, and the heatmap feedback mechanism.

Thus, the input tensor is formed from $Sample(T)$ and the ground truth heatmap from the previous sample $Heatmap(T-1)$. This tensor is then passed through the same modified Swin transformer backbone as in the detection case (see Subsection 3.2.2), with the only difference being that the parameter for the number of input channels C is increased by one, from three to four. Training is done identically to the detection model, by using the focal loss between the predicted heatmap $\hat{H}$ and the ground truth heatmap $Heatmap(T)$, but also adding another regression head for

predicting the tangential velocity v_t. The L1 loss is employed again, which is also scaled by the same factor of λ. Thus, the final tracking loss comes to Formula 3.5.

$$L_{tr} = L_{det} + \lambda L_{v_t} \tag{3.5}$$

During inference, similarly to CenterTrack[38], instead of using the ground truth heatmap from the previous sample $Heatmap(T-1)$, we use the heatmap generated from the center-point and bounding-box predictions of the model from the previous timestep $T-1$. Angle bins from $\hat{H}(T-1)$ which are peaks, and larger than a threshold τ_{tr} are considered trustworthy predictions $P(T-1) = \{(\theta, r, v_r, v_t, w, l, \theta_o, conf)\}$, with the confidence being the $\hat{H}(T-1)$ value from bin θ. From these predictions a new heatmap is generated $\hat{H}_{gen}(T-1)$ by following the same Gaussian splatting algorithm as in subsection 3.2.1 which takes into account the bounding-box sizes of the predictions. The generated heatmap is used in order to condition the model with the previous predicted tracks. $\hat{H}_{gen}(T-1)$ together with the radar sample $Sample(T)$ form the input for the current tracking step, which will predict the tracklets at timestep T – $P(T)$. The tracking model architecture, including this feedback loop is shown in Figure 3.10.

3.3.2 Track generation

For associating the tracks, a simple greedy matching algorithm is employed, as in CenterTrack. As each track from $P(T-1)$ is expressed relative to the ego-vehicle position at timestep $T-1$, they are first converted to the coordinate frame of the ego-vehicle at the current timestep T. Afterwards, they are propagated forward by their velocity, which is the composition of v_r and v_t. In decreasing order of confidence, each prediction from $P(T)$ is associated to the closest propagated prediction from the previous timestep. A match is considered if the distance between the prediction from $P(T)$ and the prediction from the coordinate-adapted propagated $P(T-1)$ is less than $l(T)$. This is another difference compared to CenterTrack, as in the image case, they are using the geometric mean of the height and the width of the predicted bounding box. This is not necessary in our representation as ideally, bounding boxes which represent vehicles, do not stack, as opposed to the image case when targets can occlude each other in the perspective projection of the camera. If a track is matched, the previous unique id is propagated to the current prediction, and both previous track and current prediction are eliminated from their respective unmatched set. Remaining unmatched tracks from $T-1$ are simply discarded, while unmatched predictions from T are assigned new unique ids.

3.4 Implementation

The whole project is developed in Python, using the `PyTorch` framework[24]. The dataset preprocessing is implemented using methods provided by the package `nuscenes-devkit` from GitHub[22], which traverse and read the compressed data format of the nuScenes dataset radar point clouds. The Swin Transformer backbone

and training framework is adapted starting from the official implementation from the `Swin-Transformer` GitHub package[21], by adding the aforementioned generalizations for the shifted window attention sizes and patch-merging layers. Additionally, the CenterNet regression heads and losses are implemented on top of the Swin Transformer, together with an interfacing with the preprocessed nuScenes dataset.

Chapter 4

Results

4.1 Dataset

The model training process is conducted on a subset of the publicly available nuScenes dataset, comprising 850 annotated scenes. This dataset is split into training and validation sets with a validation proportion of 0.2, resulting in 680 scenes allocated for training and 170 for validation. Each scene includes approximately 40 input samples, derived from radar sweeps collected at a rate of 2 Hz over a 20-second span. Notably, samples from each scene are kept together, ensuring that all samples from a given scene reside in either the training or validation set to maintain consistency in temporal and spatial characteristics across the two sets.

The dataset is preprocessed as presented in Subsection 3.1.2, using $N_s = 6$ and $N_{ab} = 360$. This means that for each sample, the sample sweep and the previous five sweeps are quantized into 360 angle bins, and stacked. Points which are more than 50m away from the ego-vehicle are dropped as well. The same filter is also applied to the ground truth annotations. Furthermore, the annotations which do not contain a radar-point reading inside of their bounding-box in the sweep sample are eliminated from the ground-truth, as the model would not be able to detect or track them when there are no readings available. From the remaining bounding boxes, the ground-truth heatmap is generated. The dataset processed in such a way is named *all targets* (`AT`) in the rest of the thesis. Additionally, the data set is also processed in another variant: *no static points, only vehicles* (`NSOV`), in which only radar point readings which are labeled by nuScenes as moving (in absolute terms) are kept. Besides this, ground-truth annotations which do not have the `vehicle.moving` are dropped as well. This results in a sparser dataset, but in which most points have a radial velocity reading. It is expected that training on this smaller but more precise dataset should result in better performing detection and tracking models, but only for moving road users. By considering these "subsets" of the nuScenes dataset, we can better evaluate the models by gaining insights into performance on specific cases.

An example input frame processed in both variants can be seen in 4.1. Inconsistencies in the radar processing pipeline, or in the nuScenes velocity compensation

algorithm may result in radar points labeled as "moving", such as the ones coming from wall on the right side of the vehicle, which should not be present in `NSOV`.

GT set	#GT annotations	Filtering Rules
NSOV	14253	#radar points ≥ 1 **and** `vehicle.moving`
AT	63276	#radar points ≥ 1
nuScenes Detection (insight)	58565	#radar points ≥ 1 **and** #lidar points ≥ 1
nuScenes Detection	159613	#radar points ≥ 1 **or** #lidar points ≥ 1

Table 4.1: Size of the validation ground-truth sets of the different subset preprocessing versions for the nuScenes dataset

The final size of ground-truth annotations set of two variants of subsets can be seen in Table 4.1. The number of ground-truths in the official nuScenes detection task is higher, as they include also annotations without a radar reading, but that have a lidar one. In the 50m maximum detection range, most ground truths that have a radar reading also have a lidar reading. This is demonstrated by the fact that there are 58K ground truths which have readings from both sensors, compared to the 63K only for radar. The extra 90K ground-truths, the difference between the nuScenes detection set and the AT set, do not make sense to be introduced in the training and validation sets of a radar-only detection model, as the targets would be impossible to predict without any radar data readings. Alas, nuScenes includes them due to the fact that the detection task is targeted mostly for camera, lidar, and fusion models. *Since the nuScenes detection task started in 2019, up to date, there have been only 3 radar-only detection model submissions.*

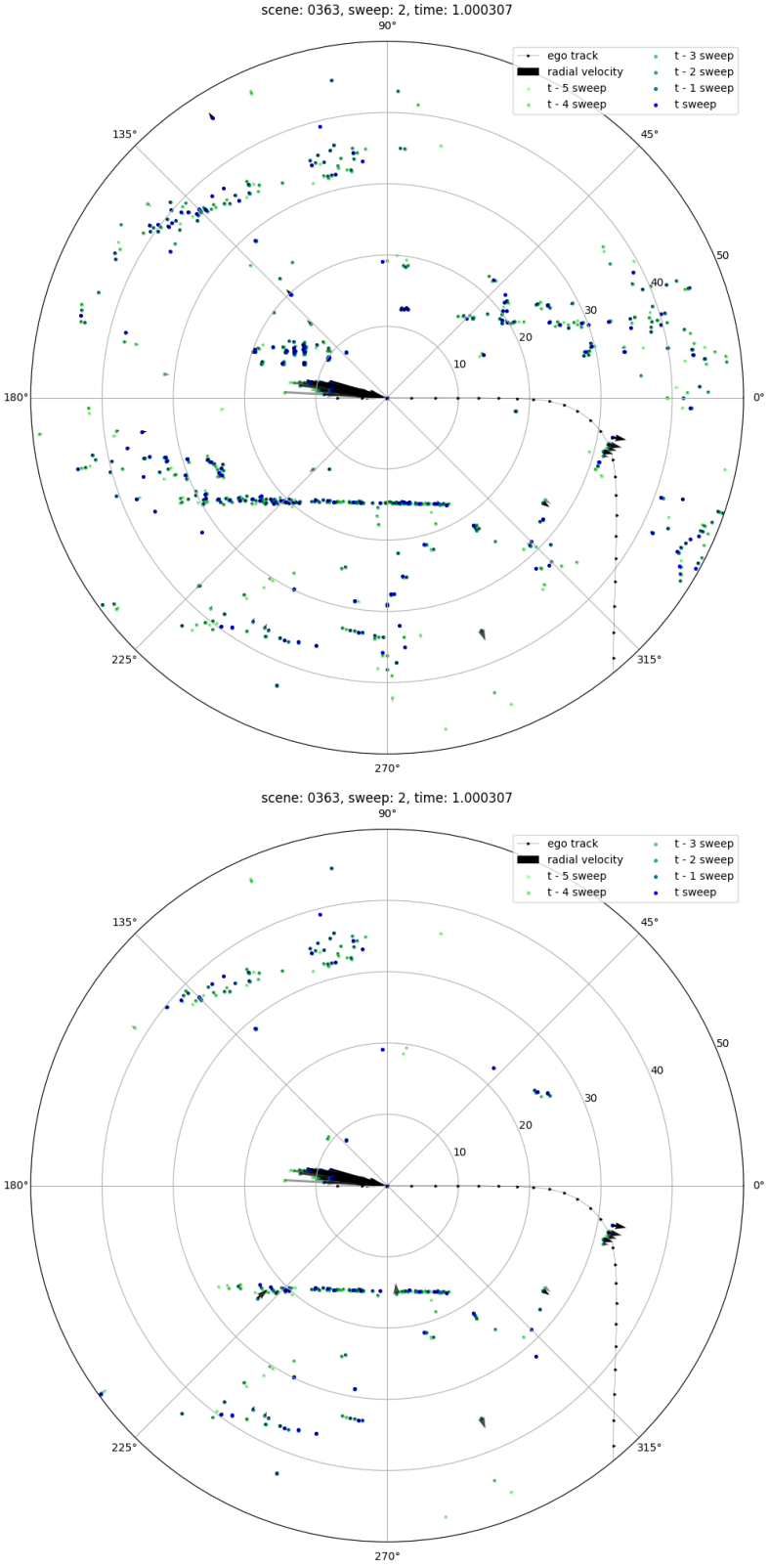


FIGURE 4.1: AT and NSOV versions of nuScenes radar data preprocessing

4.2 Model parameters

The parameters chosen for the modified Swin Transformer backbone can be seen in Table 4.2. The initial patch merge from the original Swin Transformer is skipped, and each initial "pixel" is embedded through a multi-layer perceptron into the initial embedding size $E = 24$. Similar to [17], the model is composed of multiple stages - 4, each containing multiple Swin Transformer Blocks. The stage depths are even numbers because each MSA (Multi-head Self Attention) block is followed by its SW-MSA (Shifted Window Multi-head Self Attention) block pair. All blocks from a stage have a certain number of attention heads, as shown in the table. The attention window size represents the height and width of the attention windows, which correspond to the N_s (temporal) and the N_{ab} (spatial) dimensions. All windows cover the whole temporal dimension, so that vertical shifting of the attention windows is not necessary, but only horizontal – spatial. The merge factor is the parameter for the adapted patch-merging stage which happens at the end of each Swin Transformer stage. The second patch merging factor from each tuple is constantly 1, meaning that patches are not merged at all in the angle-bin direction through the four stages, only the temporal dimension is being squashed. The reduction factor, similarly to the Swin Transformer, represents the reduction of the embedding size after each patch merge. The tensor output dimensions of each stage can be seen in Table 4.3. With these model specifications, a model has the size of 0.3MB parameters, with a forward pass costing 0.46 GFLOPS compute.

Parameter	**Value**
Initial Embed Dim	24
Stage Depths	{6, 12, 6, 6}
MSA Heads	{2, 4, 4, 4}
Attention Window Size	$\{(6,6), (6,12), (3,12), (1,12)\}$
Merge Factor	$\{(1,1), (2,1), (3,1), (1,1)\}$
Reduction Factor	{1, 2, 2, 1}

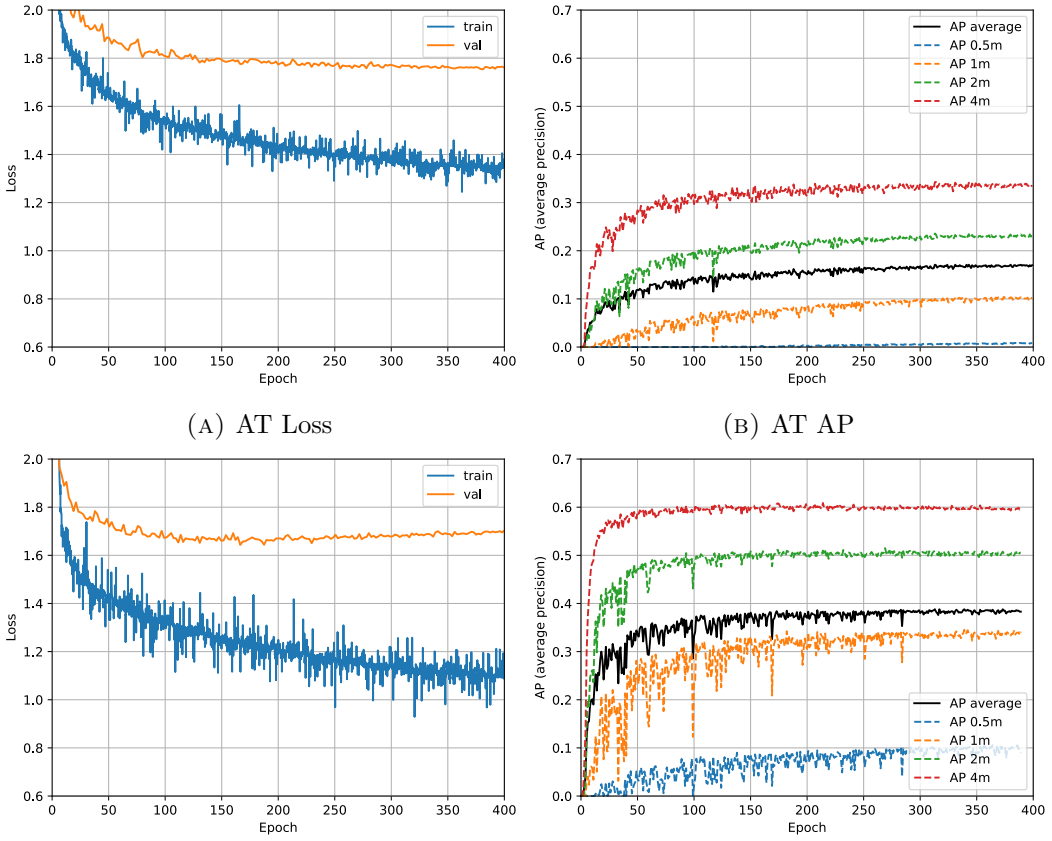
Table 4.2: Radar-Swin transformer parameters

Stage	**Output Dimension**
Input Processing	$6 \times 360 \times 3$
Initial Embedding	$6 \times 360 \times 24$
Stage 1	$6 \times 360 \times 24$
Stage 2	$3 \times 360 \times 24$
Stage 3	$1 \times 360 \times 36$
Stage 4	$1 \times 360 \times 36$

Table 4.3: Radar-Swin Transformer stage output dimensions

4.3 Detection model training

Two models are trained, one on the NSOV ground-truth set for detecting moving targets, and AT, trained on the whole ground-truth set of targets that receive at least a radar reading. The models are trained for 400 epochs using a batch size of 128. Each model is trained on a single NVIDIA A100 Tensor Core GPU, with the training time being near 10h, depending on the load of the compute servers and the availability of cores responsible for data loading. Similarly to the Swin Transformer[17], the first 4 epochs are used as warm-up, before a starting a cosine learning rate scheduler. For regularization purposes, a weight decay of 0.01 is used, together with a drop-path rate of 0.2. Figures 4.2a and 4.2c show the evolution of the training and validation losses during training. As the AP continues to slowly increase during training, the AT model at the end of the 400 epochs is chosen. The NSOV model begins overfitting, shown by the fact that the training loss continues to decrease while the validation loss is slowly increasing. Nevertheless, as the associated AP does not decrease but only stagnates, the resulting NSOV model at the end of the 400 epochs is also chosen as it has the best AP performance.



(A) AT Loss

(B) AT AP

(C) NSOV Loss

(D) NSOV AP

FIGURE 4.2: AT and NSOV detection models training evolution

4.4 Detection performance

4.4.1 Quantitative

The detection metric proposed by nuScenes[3] is a modified version of the well-known mAP (mean average precision).

The average precision (AP) metric is a commonly used performance measure in multi-object detection tasks, defined as the area under the precision-recall (PR) curve. Precision and recall are two fundamental metrics in detection, calculated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where TP, FP, and FN represent true positives, false positives, and false negatives, respectively. In the context of object detection, a matching function is used to determine whether a predicted bounding box corresponds to a ground truth bounding box. If the predicted box sufficiently overlaps with a ground truth box (based on a defined threshold), it is considered a match (true positive). If not, it is a false positive. Any ground truth box that is not matched with a prediction is counted as a false negative. Notably, in object detection, true negatives are not considered, as predictions for "background" regions (regions without target objects) is not required.

To compute precision and recall, predictions are first sorted by confidence score in descending order. For each prediction, the ground truth box with the highest matching score is identified. If the match is above the threshold, it is classified as a true positive; otherwise, it is a false positive. Once a ground truth box is matched to a prediction, it is removed from the set of available ground truths to ensure that each ground truth generates only one true positive. This process results in cumulative sums of TP and FP, which are then used to calculate precision and recall at each confidence threshold.

The PR curve, which represents precision as a function of recall, is generated from these values. Since recall is guaranteed to be non-decreasing with cumulative matching, the curve can be interpolated to smooth any fluctuations in precision. The area under this curve represents the average precision (AP), capturing the model's ability to balance precision and recall across varying confidence thresholds.

The nuScenes dataset introduces modifications to the classical AP metric to better suit the challenges of autonomous driving detection tasks. Instead of using Intersection over Union (IoU) as the matching criterion, nuScenes uses center-point distance, with thresholds set at 0.5, 1, 2, and 4 meters. This modification acknowledges the difficulty of precise localization at greater distances, especially in complex urban environments where accurate bounding box alignment may be challenging even with higher resolution sensors like lidar and cameras. The area under each precision-recall curve where recall or precision are less than 0.1 is also not considered, as it is usually unreliable. For each matching threshold, AP is calculated

independently, and the final nuScenes detection AP metric is obtained by averaging AP scores across these thresholds. Averaging the AP across the different classes produces the mAP score, which is the main nuScenes detection metric, but that is not relevant for this work as the model does not predict classes.

Additionally, nuScenes applies specific filtering criteria to improve the relevance of the AP calculation. Ground truth bounding boxes that do not contain at least one lidar or one radar point are excluded from the evaluation, as these objects would be inherently undetectable by any model. Furthermore, a maximum center-point distance to the ego-vehicle threshold is imposed based on object class: 4-wheeled vehicles are limited to a 50-meter range, 2-wheeled vehicles to 40 meters, and pedestrians and road furniture to 30 meters. These distance-based cutoffs ensure that objects are only included in the evaluation if they are within a range where accurate detection is feasible.

The evolution of the AP metric for the AT and NSOV models during training can be seen in Figures 4.2b and 4.2d respectively. The results come from validation rounds done every 5 epochs during training, using a personal implementation of the AP metric, inspired by the nuScenes version. An important point to note is the fact that the two models have different ground truths sets, due to the differences in the initial filtering. Therefore, the AP results should be understood in relation to the size of the ground truth sets: $TP + TN$. The total number of predictions on the validation set of the two models, on their respective input data can be seen in Table 4.4. Model predictions are generated by choosing the output heatmap peaks which are above the $\tau_{det} = 0.25$ threshold.

Model	**#GT**	**#Pred**
NSOV	14253	19275
AT	63276	100284

TABLE 4.4: Model number of predictions relative to #GT in the validation subset

Tables 4.5 and 4.6 show the model performances relative to the AT and NSOV ground truth annotations respectively. Both models perform comparatively better on the NSOV set than on the AT set. This is due to the fact that the non-zero radial velocities of the radar readings come as a clear indicator to the model than an object is present at that angle-bin position. Conversely, in the AT case, static targets are easily mixed up with the scene background, such as buildings or other objects that are not annotated, as they both produce sparse and inconsistent radar readings. The AT's model performance general performance on both GT sets of 0.17 AP, compared with the 0.38 AP of the NSOV model shows that there is room for improvement of the AT model, that could potentially be achieved through hyper-parameter tuning or increasing the model size.

Table 4.7 shows a imprecise comparison with the state of the art, as the nuScenes detection task also requires predicting object class, which the Radar-CenterNet-Swin model does not currently do. There are 10 classes required in the detection task: `barrier`, `traffic_cone`, `bicycle`, `motorcycle`, `pedestrian`, `car`, `bus`,

Model	AP	AP@0.5m	AP@1m	AP@2m	AP@4m
NSOV	0.027	0.000	0.010	0.038	0.060
AT	0.166	0.009	0.099	0.226	0.329

Table 4.5: Performance of models on the AT ground-truth subset

Model	AP	AP@0.5m	AP@1m	AP@2m	AP@4m
NSOV	0.384	0.115	0.337	0.499	0.586
AT	0.178	0.039	0.150	0.235	0.287

Table 4.6: Performance of models on the NSOV ground-truth subset

`construction_vehicle`, `trailer`, `truck`. Therefore, the metrics for the model trained on the AT set should actually be read without the "mean" component, which represents a mean across classes, as they are calculated class agnostic. As such, the table actually compares the AP metric of the AT model with the mAP metric of the nuScenes detection task submissions. In order to compute the metric for the AT model, the official nuScenes implementation is used, with the modification that all prediction and ground-truth annotations are considered to be of the `car` class for the AT model.

Model	mAP	mATE	mASE	mAOE	mAVE
RadarDistill[1]	0.205	0.461	0.263	0.525	0.336
KPConvPillars[32]	0.049	0.823	0.428	0.607	2.081
Radar-PointGNN[30]	0.005	1.024	0.859	0.897	1.020
Radar CenterNet-Swin AT*	0.053	0.786	0.377	0.561	0.828

Table 4.7: Approximate comparison with state of the art (*Our model does not predict classes, and is evaluated on the reduced AT ground-truth set)

Additionally, the nuScenes detection task evaluates the models on the full detection set, in which even ground truths that contain at least one lidar point but no radar points are included (see Table 4.1). The increase of the size of the ground-truth set, with annotations without radar points which are practically undetectable for a radar-only model explains the reduction in AP performance of the AT model compared to the previous evaluation on the AT ground-truth set from Table 4.5.

The remaining secondary metrics calculated by the nuScenes evaluation code are as follows. Average Translation Error (ATE) measures the euclidean center-point distance error on the 2D ground plane in meters. Average Scale Error (ASE) represents the difference in scale between matched ground truths and predictions, and is calculated as $1 - IoU$ after aligning center-points and orientations. The Average Orientation Error (AOE) represents the difference in rotation around the z-coordinate between ground truths and predictions, calculated in radians. Finally, the Average Velocity Error (AVE) represents the difference in m/s between the matched prediction and ground truth velocities. As the Radar-CenterNet-Swin model does not predict the tangential component, it is set to 0 in the predictions. Similarly to the mAP,

the metrics for the submitted models are calculated on a per-class basis, and then averaged, as opposed to the AT model where predictions and ground truths are matched class agnostic.

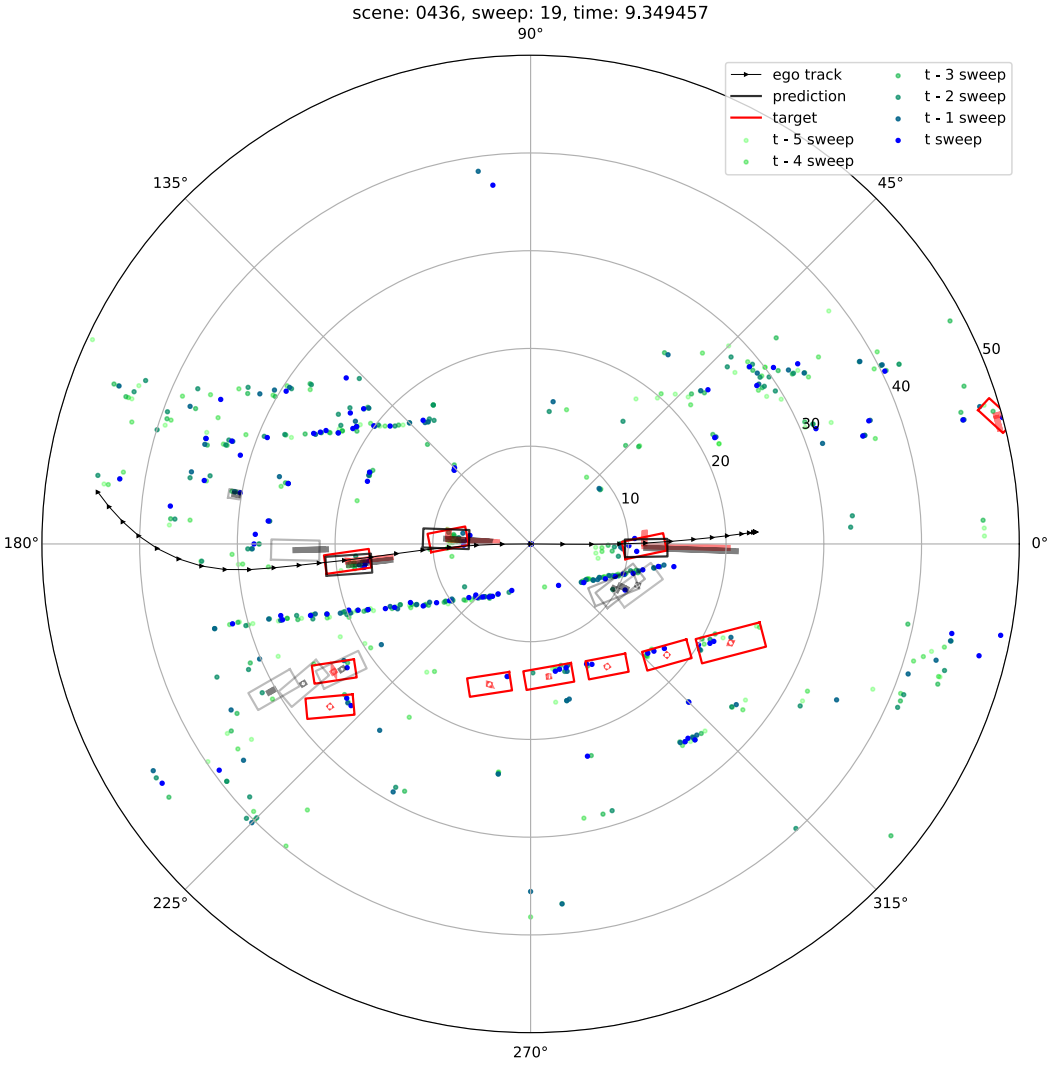
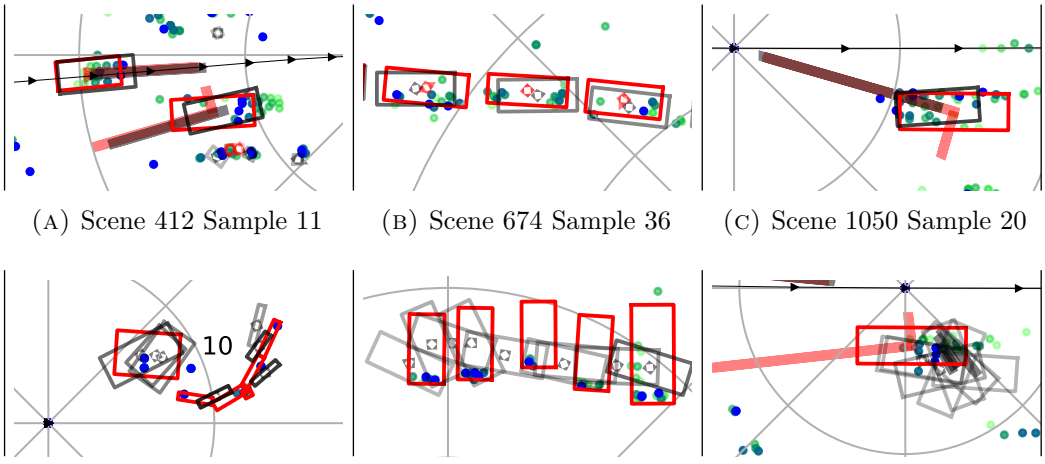


FIGURE 4.3: AT Detection example scene 436

4.4.2 Qualitative

Figure 4.3 shows an example prediction of the AT detection model on one of the validation set samples. The predicted annotations are black, with their transparency indicating the confidence of the prediction - the value of the heatmap at that angle bin.

The three moving vehicles are are predicted with high confidence and precision across all variables: range, orientation, bounding box size, and radial velocity, as they present clear tracklets which are easily matched by the model. The parked cars on the right side of the ego-vehicle (bottom side of the image) are almost completely missed, except for one, as their radar fingerprints are difficult to differentiate from the clutter produced by the environment. The model even predicts, albeit with low confidence but still higher than the τ_{det} threshold, some bounding-boxes at approximately $r = 10, \theta = 320°$. This shows the difficulty in differentiating static targets from the background readings.



(a) Scene 412 Sample 11 (b) Scene 674 Sample 36 (c) Scene 1050 Sample 20

(d) Scene 358 Sample 5 (e) Scene 664 Sample 36 (f) Scene 1050 Sample 21

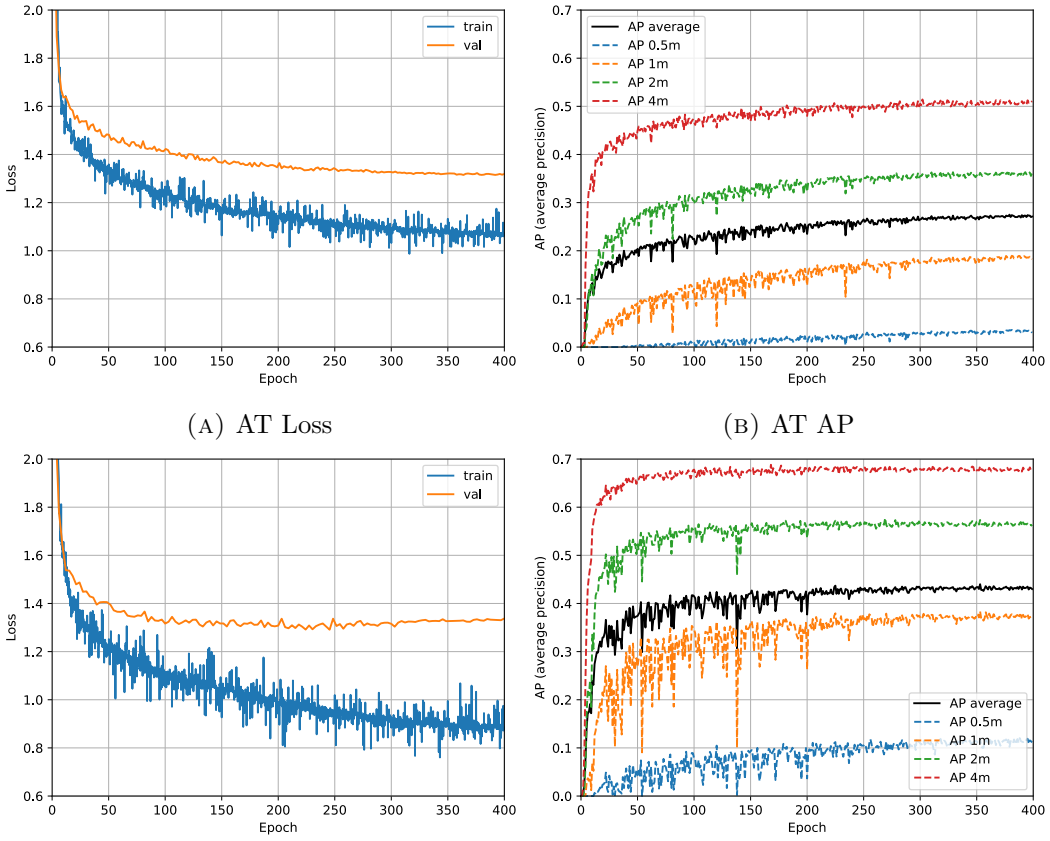
Figure 4.4: Detection snippets from various scenarios

The strengths and weaknesses of the detection model can be further seen in Figure 4.4, by comparing the top and bottom rows. Snippet 4.4a shows the case in which the model performs the best: moving vehicles, in comparison with static objects Snippet 4.4d, for which the orientation is hard to predict, especially considering the very sparse radar readings. The size prediction also seems to overfit to matching a standard car shape across most samples. Snippets 4.4b and 4.4e show the performance on cars parked on the side of the road, which appear in most driving scenarios. In the cases when there are sufficient radar readings, detection functions properly. There is also a tendency to suppose cars are parked parallel as opposed to perpendicular to the road and movement of the ego-vehicle. Snippets 4.4a and 4.4c exhibit how the detection model properly predicts the radial velocity, as the ground truth and prediction vectors match with high precision, being almost fully overlapped. In Snippet 4.4f one of the weaknesses of the current model is visible - targets which

occupy more than 12° in the angular dimension, present a noisy heatmap prediction, which leads to multiple extracted peaks and thus multiple bounding-boxes predicted for the same target. This upper limit in the angular receptive field is caused by the structure of the model, through the attention window size of maximum 12 bins, and the lack of hierarchical patch merging in the angle-bin dimension (see model parameters in Table 4.2).

4.5 Tracking model training

Similarly to the detection case, two tracking models are trained on the `all targets` AT and `no static, only vehicles` NSOV subsets of the nuScenes dataset. The model parameters are kept the same as in 4.2, with the exception that the input tensor has one extra channel, for the previous timestep heatmap. As the embedding dimension remains the same, the model size and the rest of the output dimensions do not change. As mentioned in Subsection 3.3.1, only one extra regression head is added for the tangential velocity prediction. As such, the model parameter number and forward pass compute cost remain virtually unchanged.



(A) AT Loss

(B) AT AP

(C) NSOV Loss

(D) NSOV AP

FIGURE 4.5: AT and NSOV tracking models training evolution

During training and also validation, the ground truth heatmap from the previous sample is appended as an extra channel to the radar data, as per the CenterTrack feedback idea. In order to do this, the first sample from each scene is skipped, as it does not have a previous sample heatmap. Consequently, the losses and AP performance of the models during training should be understood from this perspective, because as we will see, the addition of the previous ground-truth heatmap provides a lot of extra information. Even though the validation set AP is a detection metric, it has been kept as a secondary way to follow along with the model improvement during training, besides the training and validation losses. Similarly to the detection case, the NSOV model converges rapidly, with the validation loss starting to increase around epoch 300, due to the sparsity of the input data, which has multiple samples that do not even contain moving targets, thus no ground-truth annotations at all.

All of the other unmentioned training parameters are also kept identical to the ones presented in Table 4.3.

4.6 Tracking performance

4.6.1 Quantitative

The main tracking metric used by nuScenes is based on the Multi-Object Tracking Accuracy (MOTA) metric proposed in [2]. The MOTA metric aims to also take into account the number of id switches (IDS), which are calculated during a prediction to track matching algorithm. The matching algorithm has a similarity threshold parameter, which in the nuScenes case is the center-point euclidean distance of 2m.

Both the ground-truth G_t and prediction P_t sets, for each timestep t, are represented by a set of pairs of bounding-boxes and associated track ids. The goal of the matching algorithm is to generate the set $M_t = \{(g_i, p_j)\}$ of ground truth to prediction mappings at time t, using G_t, P_t, and M_{t-1}. M_0 starts initially empty.

The matching algorithm associates ground truth-prediction pairs in decreasing order of similarity, meaning with the closest center-point distance in our case, up to the defined threshold of 2m. Every (g_i, p_j) mapping made is included in the M_t set, and gt_i and p_j are removed from the G_t and P_t sets respectively. In the case the g_i object was present at the previous timestep, and a mapping (g_i, p_k) was made in M_{t-1}, it is considered an id switch occurred, as the id of the prediction track changed when it should not have. Every mapped ground truth-prediction pair is thus either considered a TP or an IDS. After there are no more pairs with distance less than 2m to match, the algorithm stops. Unmatched ground truths are considered FNs, and unmatched predictions FPs, similar to the AP metric presented in Subsection 4.4.1. As such, the total number of ground truths (GT) is equal to the sum of TP, FN, and IDS. The total number of TP, FP, FN, IDS is calculated across all scenes, and all timesteps in order to calculate the final metric:

$$MOTA = 1 - \frac{IDS + FP + FN}{GT}$$

In order to get a better insight into the performance of the tracker, an integral metric can be built on top of the MOTA, by averaging its value at different recall thresholds. As proposed by [37], this is done by sorting the predictions across all scenes from all samples by decreasing order of confidence, and calculating the MOTA at the predefined recall thresholds. When doing this, at low recall values, the FN term will completely dominate the MOTA leading to the FP and IDS terms to become close to negligible. Therefore, the following metric is proposed which corrects this:

$$MOTA_r = max(0, 1 - \frac{IDS_r + FP_r + FN_r - (1 - GT) \cdot r}{GT \cdot r})$$

In which IDS_r, FP_r, FN_r are the number of id switches, false positives, and false negatives at the recall value r. By integrating this across L recall thresholds, the final Average MOTA (AMOTA) metric is obtained:

$$AMOTA = \frac{1}{L} \sum_{r \in \{\frac{1}{L}, \frac{2}{L} \dots 1\}} MOTA_r$$

Another proposed evaluation metric by nuScenes is the Multi-Object Tracking Precision (MOTP), together with its equivalent integral metric Average MOTP (AMOTP). The MOTP is simply calculated by averaging the center point distance between all ground truth-prediction matches across all samples and all scenes, and AMOTP by integrating the MOTP-recall curve, in the same fashion as for AMOTA.

For the nuScenes tracking task, the AMOTA and AMOTP metrics shall be calculated across each class, and then averaged for the final score. As in the detection case, because our tracking model does not predict classes, we will calculate a single value for AMOTA and AMOTP by matching predictions to ground truths in a class-agnostic fashion.

GT set	**#GT annotations**	**Filtering Rules**
NSOV	16720	#radar points ≥ 1 **and** `vehicle.moving`
AT	74501	#radar points ≥ 1
nuScenes Tracking	127944	#radar points ≥ 1 **or** #lidar points ≥ 1

TABLE 4.8: Size of the validation ground-truth set of the different dataset preprocessing versions

Even so, a couple modifications are made to the ground-truth nuScenes dataset compared to the detection case. The total number of classes is reduced from 10 to 7, as the `barrier`, `traffic_cone`, and `construction_vehicle` classes are ignored. Additionally, target tracks which are interrupted due to not passing the minimum number of radar or lidar points filter are not left as is. Instead, their position, yaw, and velocity is linearly interpolated between the two closest timesteps which pass the number of points filter. Taking this into account, Table 4.8 the size of resulting

ground truth sets can be seen, depending on the type of filtering. Compared to Table 4.1, the AT and NSOV ground-truth subsets are increased because of the track-continuity interpolation. On the other hand, the nuScenes Tracking set size is reduced compared to the nuScenes Detection, because the tracking problem ignores traffic cones and barriers, which according to nuScenes occupy a total of 21% of nuScenes cuboids. These objects which are very small, are detected by the radar much more rarely than by the lidar, which explains why through their omission, the AT and NSOV sets' is not reduced in comparison with their detection counterparts.

In order to extract the model predictions, as presented in Subsection 3.3.1, predictions with confidence higher than $\tau_{tr} = 0.4$ are selected, and then passed through the CenterTrack-style greedy track matching algorithm from Subsection 3.3.2. From these predictions the estimated heatmap is also generated, and further used as a feedback mechanism for the next tracking step. In this situation, erroneous detections continue to propagate and lead to a "swarming" of FP predictions. Due to this fact, prediction confidence threshold τ_{tr} is chosen higher than τ_{det}.

	NSOV	**AT**	**nuScenes**
AMOTA	**0.236**	0.029	0.023
AMOTP	**1.420**	1.799	1.863
RECALL	0.418	0.113	0.086
MOTAR	0.558	0.281	0.269
GT	16720	74501	127944
MOTA	0.219	0.030	0.021
MOTP	0.641	0.700	0.718
TP	6557	7878	10079
FP	2895	5668	7363
FN	9734	66071	116976
IDS	429	552	889

Table 4.9: Performance metrics of the AT model evaluated on the proposed GT subsets. (*Our model does not predict classes)

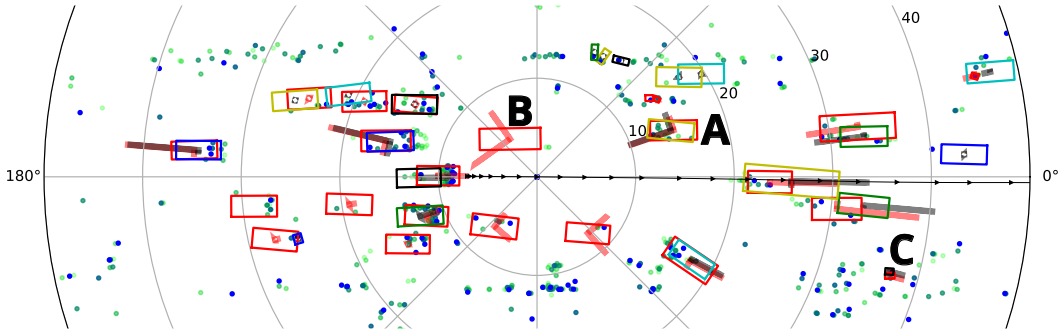
Table 4.9 shows the performance of the AT model, evaluated on the ground-truth sets from Table 4.8. All the metrics reported including and below MOTA row-wise are reported using only the model's predictions which are over a certain confidence threshold. This threshold is chosen in order to maximize the MOTA result. A further consideration is that when running on the NSOV ground-truth subset, the AT model's predictions for which the predicted speed is less than 0.1m/s are filtered out, as the NSOV ground truth contains only moving vehicles.

The overall results on the NSOV subset are clearly the best, showing the model's better accuracy and precision when tracking moving targets, which is in line with the performance of the Radar CenterNet-Swin detection model, and with radar's unique advantage of perceiving target velocity. This is further highlighted by the abrupt increase in FNs, representing mostly missed static targets, when further expanding the GT set to the AT and to the official nuScenes set respectively. The smaller

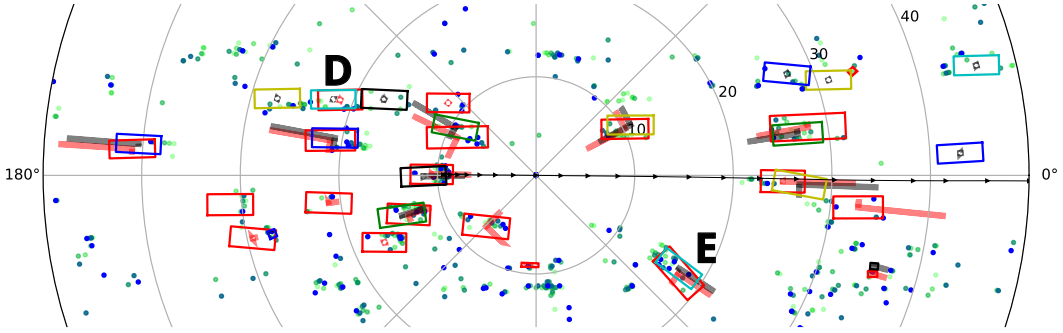
increasing trend in FPs is most probably a result of spurious predictions propagated through the heatmap feedback by the poor precision static target detections.

To put into context the difficulty of the nuScenes tracking challenge, only three tracks are proposed by nuScenes: camera-only, lidar-only, and open. *There is no radar-only track, and by the time of writing, there are no radar-only tracking challenge submissions.*

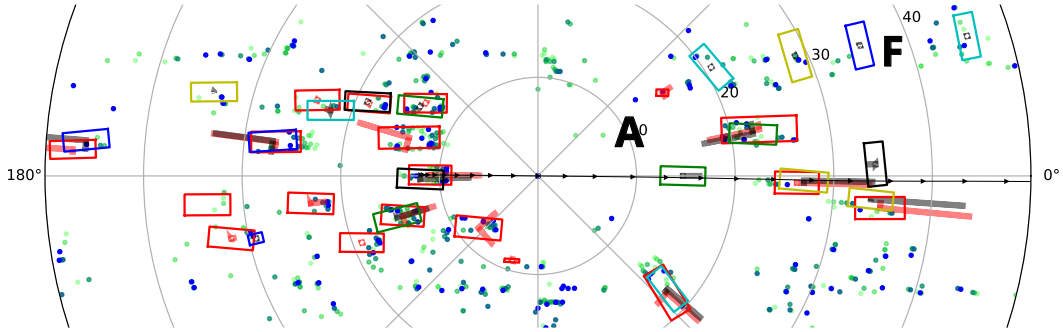
4.6.2 Qualitative



(A) Sample 7



(B) Sample 8



(C) Sample 9

FIGURE 4.6: Consecutive samples from scene 104

In Figure 4.6 the tracking results from three consecutive timesteps can be seen. Predicted track ids are represented by the color of the predicted bounding box,

meaning that a properly tracked object should have the same prediction color across all samples. Even though in the context of a single sample the same prediction color may appear multiple times, they all have different track ids. Due to the lack of high-visibility colors, some are reused. Bold letters are added to highlight interesting performance aspects. As we have seen in Subsection 4.4.2, oncoming cars such as `A` that pass next to the ego-vehicle receive very few radar readings (see `A` in Figure 4.6c). This transitions to interrupted tracks, as the simple track matching algorithm does not keep previous tracks alive, and it should be the job of the backbone to keep these tracks alive, as commented in [38]. The larger vehicle from `B` is in a similar situation, and shows how the track is regained after passing by the ego-vehicle. In the following sample Figure 4.6b the vehicle receives an imprecise detection, which then in Figure 4.6c is then associated to the parked vehicle above. `C` highlights a pedestrian which is close to 35m away, but is detected and tracked properly, at least for the first two frames. The parked vehicles at `D` are almost tracked properly, especially because they are parked parallel to the road. Vehicles which are missing the red ground-truth label, such as the black prediction in Figure 4.6b, show how missing the minimum of a radar point in the most recent sweep annotation filtering affects the ground-truth set, and thus the training of the model. `E` shows a car turning right and away from the ego-vehicle which is properly tracked through this maneuver. Lastly, `F` shows a set of spurious false positives, generated in Figure 4.6b mostly by static background readings, which then are probably further propagated through the heatmap feedback mechanism.

Chapter 5

Future Work and Conclusions

5.1 Future Work

This project opens multiple avenues for enhancement across the data level, Swin backbone level, CenterNet/CenterTrack framework, and performance analysis level. Each of these layers presents opportunities for targeted improvements to address specific challenges and optimize the system for autonomous driving applications.

At the data level, preprocessing the dataset in relative to ego-vehicle frame of reference is a promising area for exploration. By encoding the ego-vehicle's velocity directly into the input data in such a way, the model could better differentiate between driving scenarios, such as navigating city streets or parking lots at slower speeds versus traveling on larger avenues at higher speeds. This approach would allow the model to contextualize radar measurements with respect to the ego-vehicle's motion, improving detection and tracking accuracy in diverse environments.

In the Swin Transformer backbone, several enhancements can be considered. Also merging features in the angle-bin direction could enable the model to better capture larger-scale features, which would be particularly beneficial for detecting and tracking large vehicles, such as trucks, or handling vehicles with long tracklets, such as cars crossing the ego-vehicle's trajectory, with a high tangential velocity. Another area to investigate is the use of absolute positional embeddings and of relative attention bias. Absolute positional embeddings might prove advantageous in autonomous driving scenarios, where specific activities often occur within certain quadrants of the field of view (e.g., vehicles approaching from the left lane). Temporal attention masking could also be introduced to help the model focus on temporal features critical for predicting tangential velocity, improving its capability to track objects moving across frames.

In the CenterNet/CenterTrack framework, predicting multiple heatmaps—one per class—could improve the model's accuracy by separating the encoding and regression tasks for distinct object categories. For example, trucks and traffic cones could be represented with individual heatmaps, allowing the model to decode class-specific patterns more effectively. This refinement could also enhance tracking by reducing class-specific ambiguities. Additionally, the feedback mechanism for

tracking heatmaps could be revisited to better adapt it for radar data. Currently, this mechanism may contribute to spurious false positives; refining it to handle radar-specific noise could significantly enhance the reliability of the tracking system.

Performance analysis is another critical area for future work. Training models with varying depths, attention window sizes of the Swin Transformer, and other architectural parameters, could provide a more comprehensive comparison of model performance. Furthermore, analyzing model performance under specific driving scenarios—such as fast driving, stopping at traffic lights, or navigating adverse weather conditions—would yield more actionable insights. This could involve testing on different datasets that include challenging conditions like heavy rain or fog, offering a clearer picture of the model's robustness and generalizability.

5.2 Conclusions

This thesis explored the potential of radar as a sensing modality for autonomous driving by proposing a novel approach to radar data representation and developing models for detection and tracking tasks. A polar quantized representation of radar sweeps was introduced, leveraging the radial nature of radar measurements and incorporating temporal information from consecutive sweeps. Using this representation, a detection model based on CenterNet[39] and a tracking model inspired by CenterTrack[38] were designed, with significant modifications to adapt to radar-specific challenges, including the integration of the Swin Transformer[17] for spatial-temporal attention. Both models were evaluated on the publicly available nuScenes dataset using standard metrics, providing insights into the applicability of radar in autonomous driving scenarios.

The proposed angle-bin quantized radar sweep representation offers several advantages, particularly its low computational cost and simplicity, making it well-suited for early fusion of radar data from multiple sensors. This approach effectively aggregates information from multiple radar units to build a 360-degree perception system without introducing significant computational overhead. However, the representation's utility diminishes when raw radar data, such as complex frequency modulated continuous wave (FMCW) signals, is available. Raw data could potentially allow for the extraction of more informative features but would require substantially more computational resources and advanced processing pipelines. This trade-off highlights the balance between efficiency and data richness in designing radar-based perception systems.

The results from the detection and tracking models reveal both the potential and challenges of adapting vision-based architectures to radar data. While the models demonstrated decent performance in tracking moving targets, they struggled with detecting smaller or static objects, particularly in cluttered environments. These limitations stem from the sparse nature of radar data, which lacks the richness and density of camera or lidar data. The tracking model, however, showcased the ability to leverage radar's unique velocity information, enabling it to maintain consistency for dynamic targets across frames. The results underline the importance of tailoring

deep learning architectures to radar's characteristics to fully exploit its strengths.

Radar's limitations in autonomous driving detection and tracking primarily stem from its low reading density. To face this challenge, more complex architectures or fusion approaches that combine radar data with other sensing modalities, such as cameras or lidar are necessary. While radar alone may not be sufficient for primary object detection and tracking tasks, its robustness in adverse weather and its ability to provide precise velocity measurements make it an excellent auxiliary sensor. Future advancements in radar fusion systems could enable more comprehensive and reliable perception systems for autonomous driving.

Bibliography

[1] G. Bang, K. Choi, J. Kim, D. Kum, and J. W. Choi. Radardistill: Boosting radar-based object detection performance via knowledge distillation from lidar features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15491–15500, June 2024.

[2] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[4] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[5] D. Clark and J. Bell. Data association for the phd filter. In *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 217–222, 2005.

[6] D. E. Clark, K. Panta, and B.-n. Vo. The gm-phd filter multiple target tracker. In *2006 9th International Conference on Information Fusion*, pages 1–8, 2006.

[7] D. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*, volume I: Elementary Theory and Methods. Springer New York, NY, 11 2002.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[9] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, 140(2):107, 1993.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] X.-B. Jin, R. J. R. Jeremiah, T.-L. Su, Y.-T. Bai, and J.-L. Kong. The new trend of state estimation: From model-driven to hybrid-driven methods. *Sensors*, 21(6):2085, Mar. 2021.

[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, Mar. 1960.

[13] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, Mar. 1955.

[14] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2019.

[15] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection . In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, Los Alamitos, CA, USA, July 2017. IEEE Computer Society.

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

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

[18] R. Mahler. "Statistics 101" for multisensor, multitarget data fusion. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):53–64, 2004.

[19] R. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., 2007.

[20] M. Mallick, V. Krishnamurthy, and B.-N. Vo, editors. *Integrated Tracking, Classification, and Sensor Management*. Wiley, Jan. 2014.

[21] microsoft. Swin-transformer. https://github.com/microsoft/Swin-Transformer, 2023.

[22] nutonomy. nuscenes-devkit. https://github.com/nutonomy/nuscenes-devkit, 2023.

[23] A. Palffy, J. Dong, J. F. P. Kooij, and D. M. Gavrila. Cnn based road user detection using the 3d radar cube. *IEEE Robotics and Automation Letters*, 5(2):1263–1270, 2020.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[25] J. Pegoraro and M. Rossi. Human tracking with mmwave radars: a deep learning approach with uncertainty estimation. In *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, pages 1–5, 2022.

[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5105–5114. Curran Associates Inc., 2017.

[27] C. E. Services. Ars 408-21 long range radar sensor 77 ghz. https://conti-engineering.com/wp-content/uploads/2020/02/ARS-408-21_EN_HS-1.pdf.

[28] A. Srivastav and S. Mandal. Radars for autonomous driving: A review of deep learning methods and challenges. *IEEE Access*, 11:97147–97168, 2023.

[29] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020.

[30] P. Svenningsson, F. Fioranelli, and A. Yarovoy. Radar-pointgnn: Graph based object recognition for unstructured radar point-cloud data. In *2021 IEEE Radar Conference (RadarConf21)*, pages 1–6, 2021.

[31] J. F. Tilly, S. Haag, O. Schumann, F. Weishaupt, B. Duraisamy, J. Dickmann, and M. Fritzsche. Detection and tracking on automotive radar data with deep learning. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–7, 2020.

[32] M. Ulrich, S. Braun, D. Köhler, D. Niederlöhner, F. Faion, C. Gläser, and H. Blume. Improved orientation estimation and detection with hybrid object detection networks for automotive radar. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 111–117, 2022.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[34] B.-t. Vo, B.-n. Vo, and A. Cantoni. The cardinalized probability hypothesis density filter for linear gaussian multi-target models. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 681–686, 2006.

[35] B.-T. Vo, B.-N. Vo, and A. Cantoni. Analytic implementations of the cardinalized probability hypothesis density filter. *IEEE Transactions on Signal Processing*, 55(7):3553–3567, 2007.

[36] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7475, 2023.

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

[38] X. Zhou, V. Koltun, and P. Krähenbühl. Tracking objects as points. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV*, page 474–490, Berlin, Heidelberg, 2020. Springer-Verlag.

[39] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. arXiv:1904.07850, 2019.

[40] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019.