UCLouvain

École polytechnique de Louvain

Design and Implementation of a Multimodal Dataset Processing Pipeline for Vehicle Tracking Model

Author: **Thomas Bolteau**
Supervisors: **Benoît Macq, Christophe Craeye**
Readers: **Gauthier Rotsart de Hertaing, Christophe De Vleeschouwer**
Academic year 2022–2023
Master [120] in Electrical Engineering

Acknowledgements

This work is of considerable importance to me, as it represents the culmination of my six-year academic journey. It marks the completion of my master's degree in electrical engineering at the Université Catholique de Louvain, as well as my general engineering degree at Centrale Nantes, as part of my double degree program. These six years were filled with difficult problems, but also with equally elegant solutions, all of which contributed to my personal and academic development. These experiences allowed me to discover more about myself and shaped my interests and aspirations for the future.

I would like to express my sincerest gratitude to my supervisors, Professors Benoît Macq and Christophe Craeye, for their invaluable support and guidance throughout this work. I am also very grateful to Gauthier Rotsart de Hertaing for the generous time he has devoted to me over the past year. His contributions have been crucial to this project, and I sincerely hope that this work will make a significant contribution to his PhD studies.

My warmest thanks go to my family, who have supported me tirelessly despite the geographical distance over the last six years. Their inexhaustible energy, attention and care have undeniably been my bedrock. Without them, it would probably have been impossible to reach this stage.

Abstract

With the rise of machine learning getting quality dataset to train new model on has always been the highest problem to tackle. This thesis proposes a solution to the pervasive challenge of procuring high-quality, labeled datasets for supervised machine learning. The solution takes the form of a multimodal data pipeline, capable of extracting labels from data derived from synchronized camera images and Doppler radar heatmaps. After delving into the fundamentals of machine learning, the thesis presents a critical review of current practices in image and signal processing, paying close attention to techniques like YOLO for image processing and CFAR for signal processing. The paper then introduces a unique data structure and elucidates the proposed data pipeline, providing a meticulous walkthrough of the process, including the stages of RGB image processing, radar heatmap processing, data merging, and sequential data analysis. It also suggest the incorporation of a transformer model to allow prediction capabilities. Experimental results are then evaluated and discussed, with a particular focus on the implications of dataset characteristics, established approximate ground truth annotations, and the necessary computational resources. The study concludes with a candid discussion of identified limitations and potential improvements, summarizing the study's contributions to the domain multimodal data labelisation and preprocessing for supervised learning models.

Contents

Chapter 1

Introduction

1.1 Motivation

Over the past several decades, the automotive industry has undergone a significant transformation. The integration of increasingly complex electronics into vehicles has come to the fore. This integration has two primary objectives: enhancing the safety of passengers and collecting a comprehensive set of data. The collected data assists vehicles in adapting to various road conditions, thereby increasing safety and efficiency. The rationale is straightforward: the larger the volume of relevant data obtained, the more context is provided to the vehicle's decision-making system, potentially enhancing the vehicle's response in different scenarios.

A key ambition of the automotive industry, spanning several decades, is the development of autonomous vehicles. This once distant dream is now within our grasp, with major corporations dedicating substantial resources to this end. In 2023, it has now become common for vehicles to incorporate cameras and radar systems, primarily for obstacle detection when reversing.

However, this research proposes a more nuanced application of these tools. By merging data streams from cameras and radar during real-time driving, there is potential to develop more sophisticated collision avoidance systems. The implementation of such systems could contribute to enhanced safety measures, and it could broaden the practical functionalities of a vehicle.

This proposal presents a mutually beneficial prospect for both the industry and the user. For users, the possibility of improved safety features creates a more secure driving environment. For the automotive industry, the integration of these enhanced systems provides a pathway to diversify and improve their offerings.

Recent advances in machine learning have expanded our ability to interpret complex data sets. However, a challenge remains in developing systems capable of ingesting, processing, and learning from these data sets, mirroring how a child learns from experiences.

This learning process highlights the importance of training data in the development of these systems. Our ability to effectively teach these systems hinges on the availability of relevant and substantial data sets. François Ledent, a former Masters student, made a significant stride in this field through his master's thesis last year. He developed a hardware system capable of synchronizing data collection from a camera and radar. This system forms the basis for creating comprehensive data sets that could potentially aid in the more effective training of these models.

Despite Ledent's considerable accomplishment, a crucial area still needs to be addressed: the creation of these data sets. Without the appropriate data sets, our ability to train models effectively remains constrained. Therefore, this thesis aims at build upon Ledent's work by focusing on the generation of these vital data sets from the raw synchronized data. The difference these two things imply the the concept of label, very often useful for machine learning as explained in the second chapter. By doing so, it contributes to the ongoing efforts to refine these models and make them able to do well informed decision-making, edging us closer to the reality of autonomous vehicles.

1.2 Proposed Contributions

The scope of this thesis modestly aims to address the identified issues through the following contributions:

- **Creation of a Multimodal Dataset:** The cornerstone of this thesis is the creation of a multimodal dataset that combines radar and camera data. Tailored specifically to aid in vehicle movement prediction, this dataset strives to be a useful resource for those involved in the autonomous driving sector, supporting further research and development efforts.
- **Illustration of Multimodal Fusion Efficacy:** This work also aims to illustrate the potential benefits of fusing radar and camera data, a concept known as multimodal sensor fusion. While it's still an exploratory approach, the goal is to present its capacity to enrich environmental understanding, offset individual sensor limitations, and bolster the reliability of vehicle movement prediction.

- **Encouragement of Subsequent Research:** By planning to make the methodology publicly available, this thesis intends to encourage additional research into vehicle movement prediction. It hopes to provide a base upon which future advancements in the field can be built, potentially contributing to the evolution of more sophisticated, dependable, and safe autonomous driving systems.

In conclusion, the modest aim of this thesis is to contribute to the ongoing evolution of autonomous vehicles by addressing a pressing need for an enriched, resilient, and diverse dataset. The hope is that, through this work, it will enhance the predictive competence of autonomous systems, and contribute to improving the safety and reliability of autonomous vehicles.

1.3 Organization of the Thesis

This master thesis is divided in multiple part, each of them having a very distinct objective.

Introduction

This initial chapter presents the overall context of the thesis, outlining the motivation, problem statement, and the contributions of this research. It sets the stage for the subsequent chapters and provides an overview of the field of autonomous vehicles and the role of multimodal datasets in vehicle movement prediction.

Fundamentals of Machine Learning

This chapter introduce the reader to core machine learning concept that justify the choices made in this thesis. From the different type of learning to why labels are needed in our case.

Literature Review

This chapter presents a comprehensive review of the relevant literature. It discusses the important methods used in this thesis to process the camera data and radar data. It also introduce concepts about machine learning model used on the created dataset.

Proposed Approach

This chapter presents in detail the proposed data pipeline architecture. Going in detail in every subpart of it allows one to recreate it to solve similar problems. This chapter also describe the original data structure and the implementation of the machine learning model used to predict the vehicles trajectories.

Experimental Results

This chapter describes the created dataset and method used to analyse its quality. It describes the method employed to verify its relevance and describe the computationnal ressources needed to create it. It also includes a discussion about the results.

Conclusion

This chapter wraps up this thesis by explaining the limitations and possible future work of this thesis. It also lists the area for improvement for this work.

Chapter 2

Fundamentals of Machine Learning

Machine learning, a cornerstone of this research, encompasses a wide array of learning methods and tasks. Understanding these basic principles is key to comprehending the depth and implications of this study.

2.1 Concept of Learning in Machine Learning

The core idea behind machine learning is to enable computers to learn automatically, without explicit programming. The term "learning" in this context refers to the process of improving the performance or accuracy of a model over time as it is exposed to more data. The model uses the given data to draw conclusions or make predictions about new, unseen data. Learning takes place during the training phase, where the model iteratively adjusts its parameters to minimize the difference between its predictions and actual data.

2.2 Types of Learning

The learning process varies depending on the type of learning employed - supervised, unsupervised, or semi-supervised.

2.2.1 Supervised Learning

In supervised learning, the central method in this thesis, a model is trained using a labeled dataset. The concept of label is like a collection of question and answer that the algorithm will use to learn how to answer similar question than the one used to train it. Each instance in the training dataset comprises input features and

an associated label. The model learns to map input features to the correct labels, thereby making accurate predictions for new input data.

2.2.2 Unsupervised Learning

Unsupervised learning does not involve labeled data. Instead, it uncovers hidden patterns and structures from the input data. This method is often used for clustering, dimensionality reduction, and anomaly detection tasks.

2.2.3 Semi-Supervised Learning

Semi-supervised learning leverages both labeled and unlabeled data for training. This method is often used when obtaining labels is expensive or time-consuming. It capitalizes on the large volume of unlabeled data and a small amount of labeled data to enhance learning efficiency. The exact ratio of labeled to unlabeled data can vary, but typically, the majority of the data is unlabeled. The key idea is to use the labeled data to guide the learning process, while also leveraging the unlabeled data to better capture the underlying structure of the data distribution.

There are several methods used to accomplish this. One approach is to train a supervised model on the labeled data, then use that model to make predictions on the unlabeled data. These predictions can then be added to the training set (often with lower confidence), and the model can be retrained on the combined data.

2.3 Different Tasks in Machine Learning

Machine learning tasks typically fall into two broad categories: regression and classification.

2.3.1 Regression

Regression is a predictive modeling technique that estimates the relationship between variables. This task is a focal point of this thesis, given its appropriateness for predicting vehicle trajectories based on other variables.

2.3.2 Classification

Classification, in contrast to regression, involves predicting the class or category of a given input. It is less relevant to this thesis as our main task involves continuous, rather than discrete, outputs.

2.4 Concept of Labels in Machine Learning

In the realm of supervised learning, labels are a vital component. Labels are the 'answers' or 'truth' that a machine learning model tries to predict. Some exemple of labeled datset are (but not restricted to) cats and dogs picture associated with "cats" or "dogs" labels, some sentences with the overall sentiment ("joy", "sadness", "anger", ...) expressed in them, music file with the style of each one of them ("classic", "rock", "jazz", ...) . In a dataset, each instance of input data has an associated label. During training, the model uses these labels to learn the underlying patterns or relationships between the input features and the labels. This learning enables the model to predict labels for new, unseen input data. In the context of this thesis, labels are specific data points related to vehicles' positions (x, y, z coordinates) and velocity.

2.5 Challenges in Multimodal Labeling

The task of labeling data in a multimodal context, such as the one involving camera and Doppler radar data in this work, presents distinctive challenges. The camera provides 2D data without depth information, while the radar compensates by providing the depth and radial velocity information. However, the absence of a definitive ground truth in such a situation makes labeling a complex and laborious process.

2.6 Understanding Transfer Learning

Transfer learning is a vital strategy in machine learning, which has gained significant attention in recent years due to its efficiency and effectiveness. As the name indicate, transfer learning is the process of transferring learned knowledge from one model, often trained on a large-scale task or dataset (known as the source task), to a related but distinct task (known as the target task).

The traditional approach to machine learning involves training models from scratch on specific tasks using a large volume of labeled data. While this approach can yield accurate models, it is resource-intensive, requiring substantial computational power and time. It also necessitates a considerable amount of labeled data, which can be difficult or expensive to acquire. Transfer learning addresses these challenges by leveraging pre-existing knowledge from other related tasks to enhance the learning process.

A typical application of transfer learning involves using a pre-trained model—often a neural network trained on a large dataset like ImageNet—for a new, related task. The initial layers of the pre-trained model, having been exposed to a wide range of data, have learned generic features applicable to many tasks. These layers are often used as a fixed feature extractor, or fine-tuned with the new task's data to better suit the specific requirements.

Transfer learning holds numerous benefits. By building on pre-existing models, it allows for efficient training with less data, as much of the necessary learning has already taken place. This results in quicker model development and often better performance, especially when the available data for the new task is limited. Moreover, it is an effective way to circumvent the issues associated with the scarcity of labeled data in many fields. On the other side, if the model has been trained on data weakly relevant to the target task the result vary. Bias can appear and highly reduce the performances.

Chapter 3

Literature Review

3.1 Image Processing

3.1.1 YOLO

YOLO (You Only Look Once) is a state-of-the-art object detection algorithm that employs pretrained convolutional neural networks (CNNs) for accurate and efficient recognition of common objects in images, such as cars, trucks, bicycles, and more. The version utilized in this thesis is YOLOv5. Although YOLOv6 and YOLOv7 were available during the writing of this thesis, YOLOv5 was chosen given its extensive implementation examples and robust support from the research community.

The architecture of YOLOv5 can be explored with respect to its various versions, each tailored for specific performance and resource requirements. In this study, we have selected the "small" version of YOLOv5, as it offers a favorable balance between speed and performance (see Figure 3.1). This choice is further justified in the chapter discussing performance metrics, where the necessity of an efficient model for the tasks at hand is demonstrated. The small version retains the core features of the YOLOv5 architecture while ensuring faster processing times, making it suitable for real-time applications or scenarios with limited computational resources.

The architecture of YOLOv5, depicted in Figure 3.2, consists of three primary components: CSPDarknet, PANet, and YOLO Layer. These components work in tandem to enable efficient and accurate object detection across different sizes of YOLOv5.

CSPDarknet53

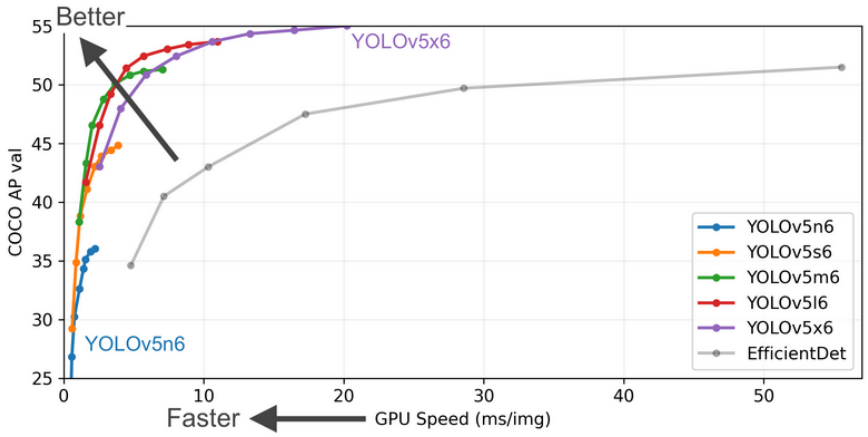


Figure 3.1: Size performance comparison regarding different versions of YOLOv5 tested on the Common Object in Context dataset by Average Precision value. Source GitHub ultralytics/yolov5

The core of the YOLOv5 architecture is formed by a structure known as CSP-Darknet53. This incorporates Darknet53, a deep network consisting of 53 layers specifically designed to process image data, as well as the Cross Stage Partial (CSP) method [11][6]. The CSP method has been designed to overcome some of the problems typically associated with large networks, such as the vanishing gradient issue and redundant gradient information, both of which can hinder the learning process.

The creators of the CSP method highlight its capacity to retain the benefits of DenseNet's feature reuse (where the network reuses previously computed features), while reducing the overflow of redundant gradient information by cutting off, or truncating, the gradient flow.

Although a comprehensive exploration of this part of the system would require more depth than is possible here, it's worth noting that the CSP method has a beneficial impact on Darknet by reducing the number of parameters involved. This reduction ultimately makes the system faster when processing or 'inferring' information from new data.

Path Aggregation Network (PANet)

The Path Aggregation Network, or PANet, serves as an integral part of the YOLOv5 architecture. It functions like a Feature Pyramid Network [20], a specific type of network designed to process image data across multiple scales. The main

goal of PANet is to boost the accuracy of pinpointing specific pixels within images, a process that's particularly crucial when predicting image 'masks' or areas of interest within an image.

To further boost efficiency and performance of the system, PANet adopts a strategy known as the Cross Stage Partial Network (CSPNet). In the context of an engineering system, this is akin to a strategy that enhances the workflow or process efficiency of the system.

Finally, by effectively combining and fine-tuning image features of different sizes and resolutions, PANet expands the YOLOv5 architecture's capacity to detect objects across a wide range of scales. This results in improved detection and recognition performance, regardless of the object's size or resolution within the input image.

Head of the network

At the end of the YOLOv5 process, there are three layers specifically designed to carry out two tasks: first, figuring out the positions of rectangles (or 'bounding boxes') that contain objects within the image; second, assigning the appropriate category or 'class' to the objects that are detected. These layers are crucial for the final step in object detection, helping the system to achieve accurate and high-quality results.

This part of the network also utilizes something called "anchor priors." These are like preliminary estimates or starting points that help refine the initial guesses for where the bounding boxes should be placed. Using these anchor priors helps to improve the accuracy of the bounding box predictions and decreases the chances of mistakenly identifying an object where there isn't one, known as a false positive.

These components, working together, form the YOLOv5 architecture. This system is capable of effectively and accurately identifying objects in images, making it valuable across a wide range of applications and needs.

3.1.2 3D from 2D images

Recovering the 3D position of an object from its 2D projected image is a fundamental challenge in computer vision with applications in robotics, autonomous driving, and augmented reality. Various methods have been developed over the years to tackle this challenge, including classical techniques such as triangulation

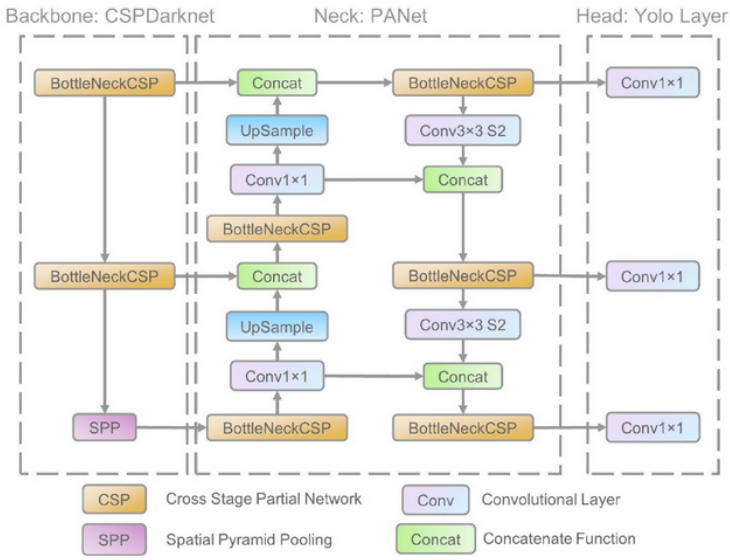


Figure 3.2: YOLOv5 architecture [22].

and more recent deep learning-based methods. Triangulation, an early and well-established method, relies on intersecting rays to estimate the 3D position of an object from multiple 2D projections. Deep learning-based methods, which have recently gained popularity due to their superior performance in numerous computer vision tasks [23], employ convolutional neural networks to learn the mapping from 2D images to 3D positions. In this thesis, we opted for classical triangulation techniques, since the RADAR sensor can provide accurate distance and radial velocity measurements for objects relative to the camera.

Triangulation

By combining information on the position of the vehicle in the image plane and its depth from the RADAR measurements, we can calculate the relative 3D position of the vehicle with respect to the hardware system. To achieve this, we need to know certain geometric parameters of the camera, such as its focal length and image size.

$$yaw = \tan^{-1}\left(\frac{x - c_x}{f}\right)$$

where x is the horizontal pixel coordinate of the object in the image, c_x is the horizontal coordinate of the principal point (i.e., the optical axis of the camera) in the image, and f is the focal length of the camera.

$$pitch = \tan^{-1}\left(\frac{y - c_y}{f}\right)$$

where y is the vertical pixel coordinate of the object in the image, c_y is the vertical coordinate of the principal point in the image, and f is the focal length of the camera.

With these values, we can compute the position of the vehicle relative to the hardware as follows:

$$\begin{bmatrix} x_{abs} \\ y_{abs} \\ z_{abs} \end{bmatrix} = \begin{bmatrix} distance \cdot \cos(pitch) \cdot \cos(yaw) \\ -distance \cdot \sin(yaw) \\ distance \cdot \sin(pitch) \end{bmatrix}$$

This model, known as the pinhole camera model [14], assumes that the lens does not distort the image. While this is a strong assumption given the limited information about the camera, it should not pose a significant issue in our implementation, as the camera has a relatively small field of view (20°). This narrow field of view reduces the impact of potential lens distortion on the estimation of the 3D position.

3.2 Signal Processing

3.2.1 Radar principle and data gathered

If explaining the inner working of the doppler radar is not the purpose or the scope of this thesis, it is necessary to understand the nature of the recovered data to be able to process them efficiently. For more detail, François Ledent explained in detail the equations and inner working of the radar system in his thesis [15].

Radar systems, including Doppler radars, operate on the fundamental principle of radio wave reflection. When a radio wave encounters an object in its path, it is reflected back towards the source. This reflected wave, or echo, can be analyzed to provide information about the object, such as its distance, velocity, and size.

Doppler radar enhances this basic radar principle by using the Doppler effect, a shift in frequency that occurs when a wave source is moving relative to an observer. By measuring the frequency shift of the returned echo, the Doppler radar can determine not just the distance of the object but also its radial velocity—how fast it's moving toward or away from the radar.

The data gathered by a Doppler radar includes the time delay between the sent and received signals, the frequency shift of the returned signal, and the intensity of the returned signal. These data points can be processed to deduce the distance, velocity, and size of the detected object.

Inner working of Doppler radar

The fundamental operation of a Doppler radar system can be broken down into three main steps: signal transmission, reflection, and reception.

Firstly, the transmitter emits a continuous radio wave of known frequency, f_t, in a specific direction. This wave travels through the air at the speed of light, c, until it encounters an object.

Upon encountering an object, a portion of the wave is reflected back toward the radar system. If the object is stationary, the frequency of the reflected wave, f_r, will be equal to the transmitted frequency. However, if the object is moving, the frequency of the reflected wave will be shifted due to the Doppler effect.

The Doppler effect can be explained by the following formula:

$$f_d = f_r - f_t = \frac{2v}{\lambda} \tag{3.1}$$

where f_d is the Doppler frequency shift, v is the velocity of the object, and λ is the wavelength of the transmitted wave. As seen from this equation, the frequency shift is directly proportional to the velocity of the object.

Finally, the receiver in the radar system detects the reflected wave. By comparing the frequency of the reflected wave, f_r, with the original transmitted frequency, f_t, the system can calculate the Doppler shift, f_d, and hence the radial velocity of the object. The time it takes for the wave to return, Δt, can also be used to calculate the object's distance, d, using the formula:

$$d = \frac{c\Delta t}{2} \tag{3.2}$$

Processing the data

Once the Doppler radar system has gathered the raw data, it undergoes several steps of processing to extract useful information. This typically involves the use of the Fast Fourier Transform (FFT), a powerful algorithm that computes the discrete Fourier transform of a sequence.

The raw radar data is typically in the time domain, where each point represents the received signal amplitude at a specific time. By applying the FFT, this data can be transformed into the frequency domain, where each point represents the amplitude of a specific frequency component in the signal.

This frequency-domain data can then be analyzed to determine the Doppler shift, and hence the radial velocity, of each detected object. The FFT output is given by:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{3.3}$$

where $X(k)$ is the k^{th} output, $x(n)$ is the n^{th} input, N is the total number of points, and j is the imaginary unit.

The FFT output can be used to generate a heatmap (see Figure 3.3), which is a graphical representation of data where the individual values are represented as colors. In the context of Doppler radar data, a heatmap can be used to visualize the distribution of detected objects over distance and speed, providing a powerful tool for analyzing and interpreting the data.

3.2.2 CFAR

CFAR (Constant False Alarm Rate) algorithms are commonly used for object detection in Doppler radar heatmap systems, as they can adapt to different environments and maintain a consistent false alarm rate. This adaptability allows the algorithm to efficiently differentiate between noise and legitimate targets, thus controlling the rate of false alarms. However, it's important to address the issue of non-detection, or missed detections, which are instances where the radar fails to identify an object that is actually present.

The performance of a radar system is evaluated not just based on the false alarm rate, but also on the probability of detection. While the CFAR algorithm is designed to maintain a consistent false alarm rate, it also has an impact on the system's detection capability. The adaptability of the CFAR algorithm often reduces clutter, which in turn can improve the detection probability in certain scenarios. Nevertheless, there may be conditions or environments where the algorithm could potentially miss detecting legitimate targets, impacting the overall accuracy of the system.

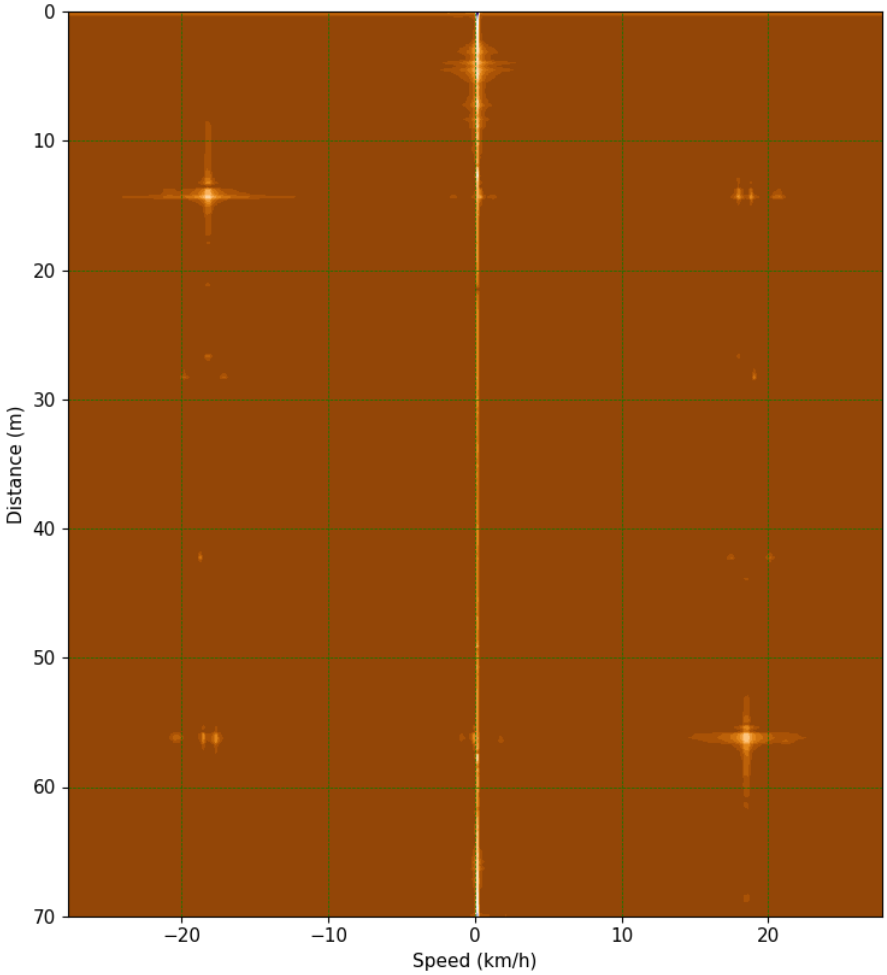


Figure 3.3: Heatmap showing distribution of detected objects over distance and speed (intensity in logarithmic scale, the brighter the higher)

Different variants of CFAR have different impacts on these two key parameters. In this thesis, the Greater Of CFAR (GO-CFAR) implementation has been chosen, as per François Ledent's recommendation [15]. GO-CFAR is a specific variant that generally provides a better balance between detection probability and false alarm rate, making it suitable for a variety of scenarios. Nevertheless, the performance of GO-CFAR, like other CFAR implementations, can vary depending on specific environmental conditions and system parameters..

However, CFAR techniques have some limitations. For instance, they may struggle to perform well in situations where multiple targets are closely spaced, as the targets can influence the noise estimation and cause false detections or missed

detections. Additionally, the performance of CFAR algorithms can degrade in the presence of clutter edges, such as stationary objects, where the background noise level changes abruptly. This is because the local noise estimate, used to set the detection threshold, can be skewed by the abrupt change in noise level at the clutter edge. If the clutter edge falls within the guard band used to estimate the noise level, the detection threshold may be set too high, leading to missed detections. Conversely, if the guard band falls mostly in a low-noise area near a clutter edge, the detection threshold might be set too low, increasing the likelihood of false alarms. In such cases, CFAR techniques may not be able to maintain the desired false alarm rate [3]..

Let $X_{i,j}$ denote the power of the cell at the $(i,j)^{th}$ position in the 2D heatmap. The kernels K_1, K_2, K_3, and K_4 represent the four quadrants surrounding the reference cell (i,j) (seef Figure 3.4), with the guard size G and window size W applied in both horizontal and vertical directions.

Figure 3.4: Representation of K_1, K_2, K_3 and K_4 where white pixel represent the non null values linearly disposed [15].

François Ledent designed the kernels, their non null value are disposed linearly with a width of m to a distance from k to l of the central point. The value of these value are $(m(l-k))^{-1}$ [15].

The power averages for these four kernels can be computed as follows:

$$\mu_k = \frac{1}{N_k} \sum_{(p,q)\in K_k} X_{p,q}, \quad k = 1,2,3,4, \tag{3.4}$$

where N_k denotes the number of cells within the k^{th} kernel K_k. The noise estimate for the GO-CFAR technique is obtained by taking the maximum of the averages for the four kernels:

$$\mu_{M_{i,j}} = \max(\mu_1, \mu_2, \mu_3, \mu_4). \tag{3.5}$$

The detection threshold is calculated by multiplying the noise estimate by a scaling factor α, chosen to maintain a constant false alarm rate:

$$T_{i,j} = \alpha \cdot \mu_{M_{i,j}}. \tag{3.6}$$

If the power of the reference cell $X_{i,j}$ exceeds the threshold $T_{i,j}$, the cell is classified as a target. The GO-CFAR approach adapts the noise estimate calculation to account for variations in the radar heatmap's horizontal and vertical dimensions. By basing the threshold on the maximum value of the surrounding noise estimates, GO-CFAR provides improved performance in situations with clutter edges and non-homogeneous backgrounds. However, while this approach does help mitigate the issues associated with clutter edges compared to traditional CFAR methods, it does not completely eliminate them. The performance of GO-CFAR at clutter edges can still be influenced by the specific characteristics of the clutter and the distribution of the noise, potentially leading to degraded detection performance in certain scenarios.

The values of α, k, l, m has been chosen by trial and error on the dataset to get the best performances possible.

The advantages of GO-CFAR include improved performance in clutter edge situations, as the algorithm accounts for abrupt changes in the noise levels by selecting the maximum value between the two averages. Additionally, it inherits from the adaptability and computational efficiency of other CFAR techniques, making it suitable for real-time applications.

However, GO-CFAR also has some limitations. It may increase the false alarm rate in the presence of closely spaced targets, as the maximum of the two averages might be influenced by target returns rather than noise or clutter [8]. In this study, we have chosen to limit the dataset to images containing only one vehicle. This decision was made in order to simplify the initial exploration of GO-CFAR's performance and potential, allowing us to focus on the underlying behaviour of the algorithm in a controlled context. However, we acknowledge that this choice does limit the generalizability of our findings, and future work may explore the performance of GO-CFAR with multiple closely spaced targets. Moreover, the choice of the parameters for GO-CFAR has to be manually tuned, which may not always lead to an optimal solution, and this is a further aspect that will be investigated in this work.

3.3 Machine Learning

3.3.1 Transformer Network

Transformers are a type of deep-learning architecture introduced by Vaswani et al. in the paper "Attention Is All You Need" [5]. They have gained significant popularity due to their effectiveness in a wide range of natural language processing (NLP) tasks. They are now used beyond NLP using various modalities as inputs [18]. Transformers are based on the concept of self-attention mechanisms, which allow the model to weigh and consider different parts of the input sequence when making predictions.

The core of the Transformer architecture is the self-attention mechanism, which computes a weighted sum of the input representations based on the relationships between the elements in the sequence. For instance, consider a sentence in a text: 'The cat, which already ate a fish, was not hungry.' Here, 'was' is contextually related to 'The cat,' despite the words in between. The self-attention mechanism, central to the Transformer architecture, allows each word to attend to every other word in the sentence, directly capturing this relationship. When processing 'was,' the model associates it with 'The cat' by assigning a higher attention weight, effectively understanding that it's the cat that wasn't hungry. The mathematical representation of this self-attention mechanism can be described as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3.7}$$

where Q, K, and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors. The self-attention mechanism computes a score between the query and key matrices, which is then used to weigh the contribution of the value matrix.

A transformer network consists of an encoder and a decoder stack (see Figure 3.5), both composed of multiple layers of identical modules which have for main purpose to calculate the self-attention mechanism.. The decoder stack is similar to the encoder stack but has an additional attention layer that attends to the output of the encoder stack.

One important aspect of the transformer architecture is the positional encoding, which injects information about the position of tokens (which represent an encoded abstraction of the input) in the sequence. Since the transformer does not have any inherent notion of sequence order, positional encoding is added to the input

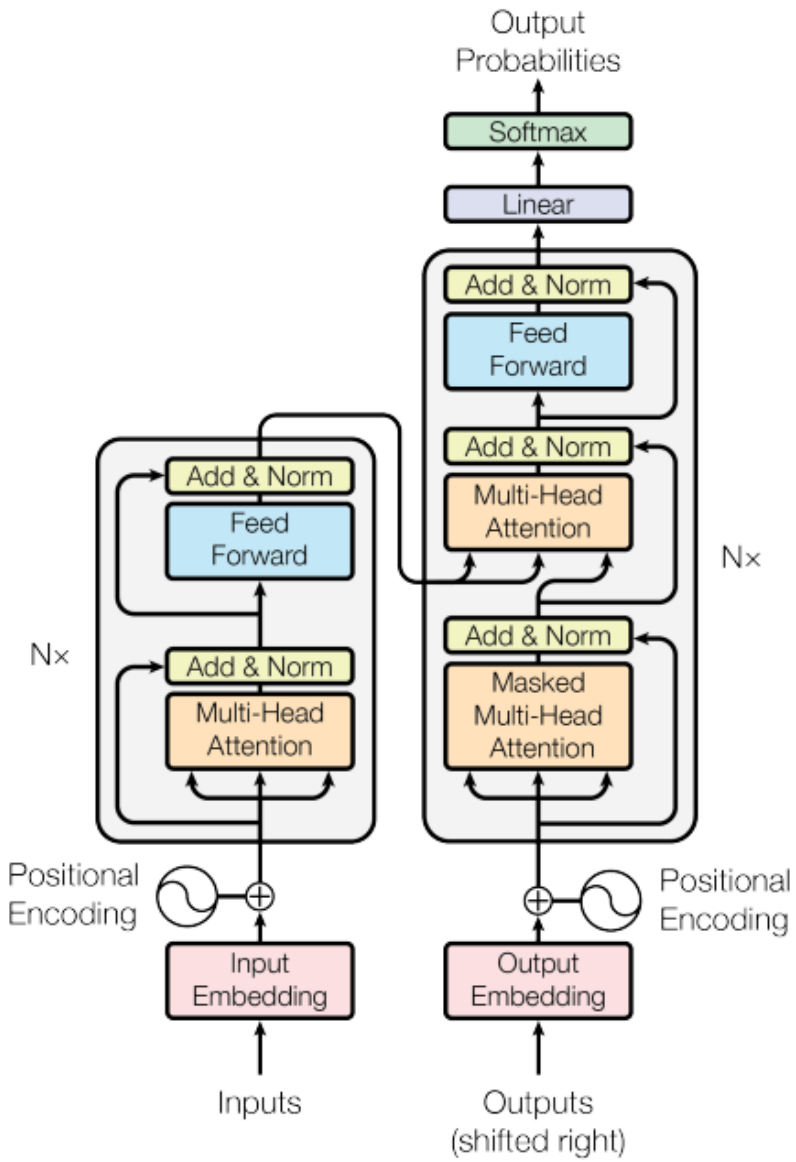


Figure 3.5: Original Transformer network architecture [5]

embeddings (the aggregation of all the token representing the input) to allow the model to utilize positional information.

Transformers have several advantages, including their ability to model long-range dependencies, parallelization of computation, and effectiveness across a wide range of tasks. However, they also have some limitations, such as their large number of

parameters, high memory requirements, and risk of overfitting on smaller datasets.

3.3.2 Transformers' Impact in Trajectory Forecasting

The application of Transformer networks has transcended the bounds of natural language processing and has made significant strides in fields like computer vision and trajectory forecasting. Trajectory forecasting is a critical problem that has gained considerable attention, particularly due to its relevance in fields such as autonomous driving and pedestrian navigation systems.

Traditional methods method often use Kalman filter [13] to predict trajectories. Others sometimes involve the use of recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) models [10]. While these models have experienced some success, they often struggle with capturing long-range dependencies, thereby making precise predictions challenging in complex situations. The Transformer network, with its self-attention mechanism and ability to focus on various aspects of the input sequence, offers a potent alternative to these traditional models.

One pivotal application of the Transformer network in trajectory forecasting is exemplified in the study "Transformer Networks for Trajectory Forecasting" [9]. This research underscores the potential of Transformer models in outperforming established LSTM-based methods when predicting pedestrian trajectories. Such achievements have encouraged further exploration of Transformer networks in the domain of trajectory forecasting.

A distinct advantage of the Transformer architecture in the context of trajectory forecasting is its proficiency in modeling long-range dependencies within the input data. The self-attention mechanism, an integral part of the Transformer network, allow the model to scrutinize diverse chapters of the input sequence and to comprehend the interactions between them at the cost of more complex calculations. This allow not to have to explicitly model the behavior of the tracked object for predicting its trajectory with traditional techniques such as Kalman filters. This capability makes Transformer networks particularly suited to predict complex trajectories within dynamic environments, enhancing the reliability of trajectory forecasts.

Moreover, recent studies have further affirmed the superiority of Transformer networks in the arena of trajectory forecasting, marking them as state-of-the-art solutions [16]. These investigations serve to validate the effectiveness of the Transformer's self-attention mechanism and its application beyond the realm of

natural language processing, solidifying its role in the advancement of trajectory forecasting. Despite the challenges associated with large data requirements and computational resources, the merits of the Transformer network continue to justify its growing prevalence in this field.

3.3.3 Mixing Modalities in Transformers network

Mixing modalities in transformer networks has gained significant attention in recent years, enabling models to leverage multiple data sources for various tasks, such as vision-and-language understanding, audio-visual processing, and more. The paper "Multimodal Learning with Transformers: A Survey" [18] provides a comprehensive overview of different approaches to multimodal learning with transformers.

There are several ways to mix modalities in transformer networks, which can be broadly classified into the following categories:

- Early fusion: In this approach, different modalities are combined at the input level, either by concatenating or summing the modality-specific embeddings. The fused representation is then fed into a single transformer network that learns to process the multimodal information.
- Late fusion: In contrast to early fusion, late fusion combines modality-specific representations at the output level. Each modality is processed independently by separate transformer networks, and the resulting outputs are fused using a task-specific fusion mechanism, such as concatenation, element-wise addition, or another learned fusion operation.
- Intermediate fusion: Intermediate fusion strategies combine modality-specific representations at different stages of the network, allowing for more complex and flexible fusion schemes. Cross-modal attention mechanisms, such as co-attention and hierarchical attention, can be employed to mix information from different modalities within the transformer layers. This approach enables the network to selectively attend to relevant information from each modality, depending on the task and context.

Each fusion strategy has its advantages and trade-offs. Early fusion is the simplest approach, but it can suffer from information loss due to the early mixing of modalities. Late fusion preserves modality-specific information, but it may not fully exploit the relationships between modalities. Intermediate fusion methods provide a balance between these extremes by allowing for more sophisticated interactions between modalities throughout the network.

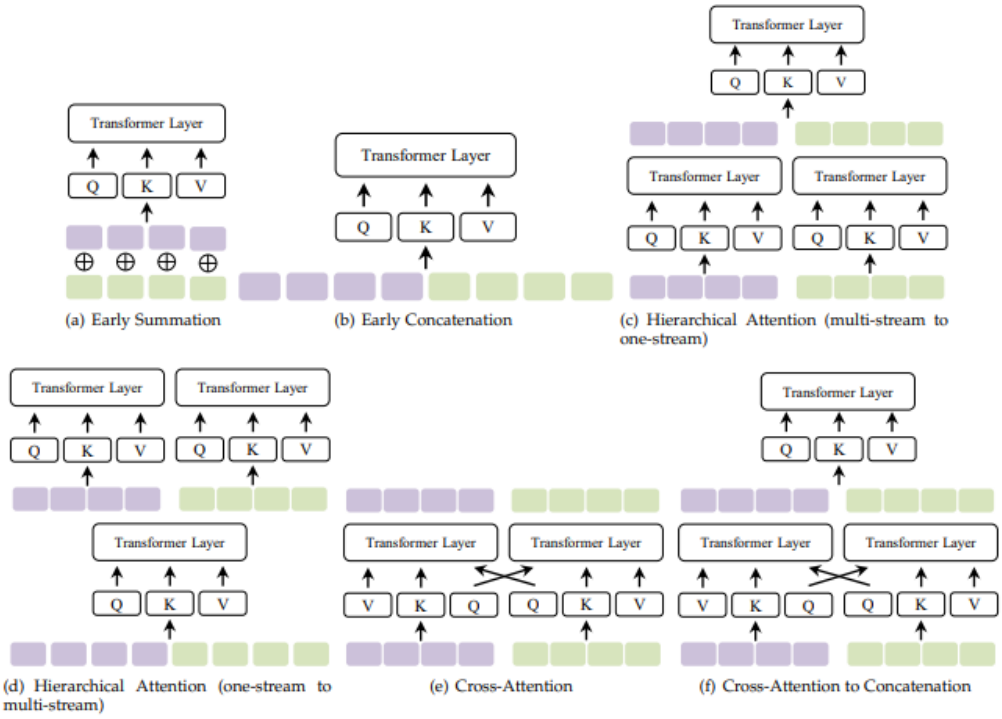


Figure 3.6: Different ways to combine modalities in a transformer network [18]

In addition to these fusion strategies, design of multimodal pipeline require a clever selection of embedding (ie. a good strategy to select abstract token to represent the inputs) approach to optimize pattern extraction possibilities by the transformer network.

In this thesis the two existing modalities are two different types of images (RGB images and radar heatmap). If were possible to consider them regarding a Vision transformer [4], it is also possible that selecting different embedding methods for RGB images and radar heatmaps leads to performance improvement but this study is out of the scope of this thesis.

Chapter 4

Proposed Approach

This chapter aims to elucidate the journey of information from the raw data, captured by the hardware system as outlined previously [15], to the final dataset. The objective of the pipeline, introduced further on, is to label the previously captured data and any future data obtained from the system. After giving an overview of the general pipeline design, we will delve deeper into each of its subparts.

4.1 Original Data Structure

Data archives

The data captured by François Ledent is organized into a series of compressed files, with each archive corresponding to a single capture session. As each RGB image and radar data file is named according to the timestamp of capture, this facilitates the handling of each session without concern over whether two timestamps in sequence are close enough in time to be related. However, it's important to note that the camera system operates at a higher frame rate than the radar system, which necessitates treatment to align the sequences, as will be explained in more detail later [15].

Legacy Tools

François Ledent developed a suite of Python functions designed to facilitate high-level information extraction from the initial dataset. These tools include CFAR (Constant False Alarm Rate), draft alignment functions, functions for creating heatmaps (position-speed) from raw binary data, and more. Significant time was devoted to documenting these functions, addressing edge cases in the implementation, and refining the code base. However, the initial drafts were highly

beneficial as they had already implemented the most mathematically-intensive calculations.

One of the central tasks in the data processing pipeline is sequence alignment, which involves associating the best corresponding image with the data from the radar system, which has the lowest frame rate [15]. Multiple alignment methods were described in François Ledent's thesis, but the implementation chosen for this project uses the approach of matching the closest timestamp. This approach was favored for its ease of implementation, low CPU usage, and intuitive logic. The process is depicted in Figure 4.1. For each radar heatmap available, the closest image in time will be chosen. This iterate over the whole capture session until the last radar data is associated with an image or that no more images are available.

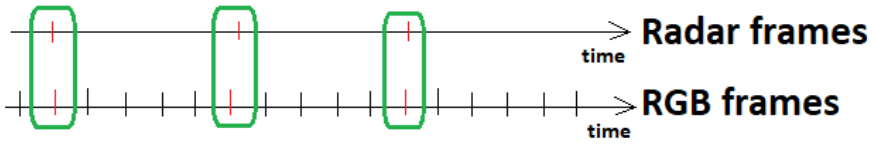


Figure 4.1: Frame synchronization illustration, two frames within the same green box are associated, while others are discarded.

Once two frames are associated, they are given the same name based on the timestamp of the radar frame, and then processed in parallel through the pipeline. This step completes the synchronization of the multimodal data stream from the hardware system [15]. As the data is exported from the hardware system before this step, it is considered step 0 of the pipeline process. Depending on the structure of the input data (whether it is already synchronized or not), this step may be skipped. This step also revealed that some RGB frames were corrupted upon export from the hardware system and were therefore unusable. Unfortunately, these frames had to be discarded, and a comprehensive statistical study on them was not conducted, although their count was negligible compared to the total number of RGB frames. This process will be refered as **Verification** in the Experimental Result Chapter.

4.2 Overview of Data Pipeline

4.2.1 Objectives of Data Extraction and Dataset Limitations

To ensure the successful completion of this master's thesis, it was imperative to define the scope of the initial dataset accurately, taking into consideration the

limitations of the primary raw data and the available work capacity.

Limitations on Simultaneous Vehicles

Owing to the hardware system's use of a Doppler sensor with a limited number of antennas (two in each direction), it's unfeasible to employ an analytical method such as the Fast Fourier Transform to track multiple vehicles concurrently. Such a method would, at most, allow the computation of K-1 angles (where K equals 2 in our case) [15]. Another method suggested by François Ledent might provide a viable alternative; however, due to time constraints, it has not been implemented in this thesis but is considered a potential area for future improvements. Therefore, the number of vehicles per couple is restricted to one per RGB-Heatmap pair in our dataset. Any pair not meeting this criterion will not be labelled but will still be retained in the dataset.

Night Time Capture

Two data capture sessions were conducted during nighttime. The first session started around 10:00:00 PM local time on July 10, 2022, and the second started approximately at 10:29:39 PM local time on the same day. The reference numbers for these sessions are 7 and 8, respectively. The sessions differed in the angles of the approaching cars, as illustrated in Figure 4.2.

((a)) Image suitable for processing **((b))** Overexposed image

Figure 4.2: Comparison of car angles

This simple comparison demonstrates that it becomes significantly challenging to recognise an incoming car at night when it's facing the camera. Through iterative testing, we found that our pipeline failed to process session 8 adequately, but it

managed to handle session 7 relatively well (see chapter 5.1). Consequently, we decided to exclude session 8 from the final dataset.

Radar Interference

The hardware system occasionally detected interference during the raw data capture process, resulting in incorrect labelling (see example Figure 4.3). To address this issue, we have implemented safety features that will be discussed in subsequent chapters. In such cases, the problematic pair will be skipped and not labelled.

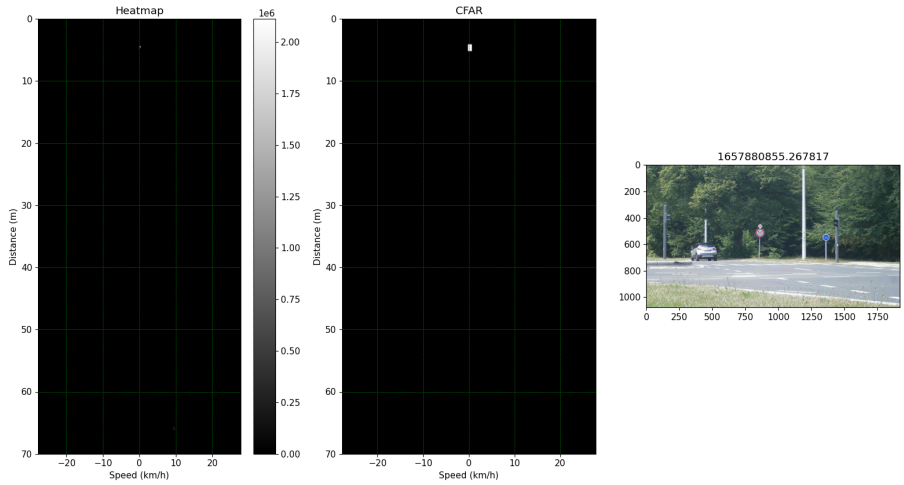


Figure 4.3: Incorrect labelling due to artefacts in radar acquisition (the higher the heatmap, the closer the object)

4.2.2 Structure of the Pipeline

The data pipeline's structure, as depicted in Figure 4.4, will be further described in individual chapters for each of its subparts.

The data pipeline consists of three primary chapters: radar data processing, camera processing, and data merging. Each arrow represents a transformation, and each box denotes a state of information. In the following chapters, each transformation will be referred to by their designated number as follows:

- **1a.** Radar heatmap filtration

- **1b.** YOLOv5 application on the camera data
- **2.** CFAR and Island algorithm application on the filtered radar data
- **3.** Quality assessment of the pair
- **4.** Data merging across modalities

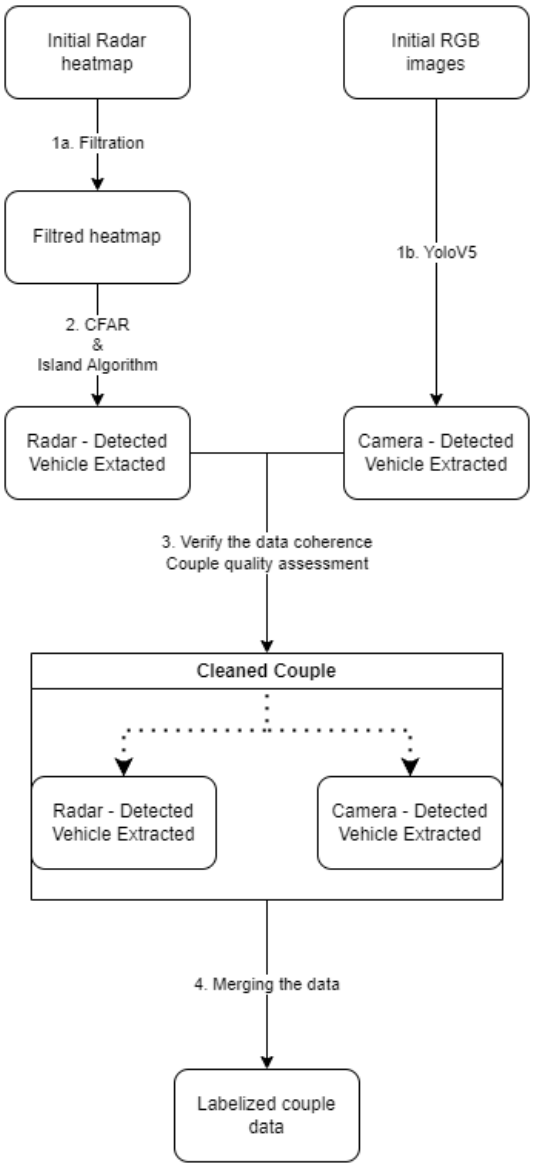


Figure 4.4: Structure of the Data Pipeline

Each of these stages is vital for the creation of a high-quality dataset and was identified as necessary through empirical evidence. The pipeline's final output consists of labelled radar-camera pairs. However, this alone is insufficient for dataset creation. While some may wish to train a model by examining every pair and thus might be interested in navigating through the entire dataset, others might prefer to access data in a vehicle-oriented manner. Hence, we developed an extension to the original pipeline, as illustrated in Figure 4.5.

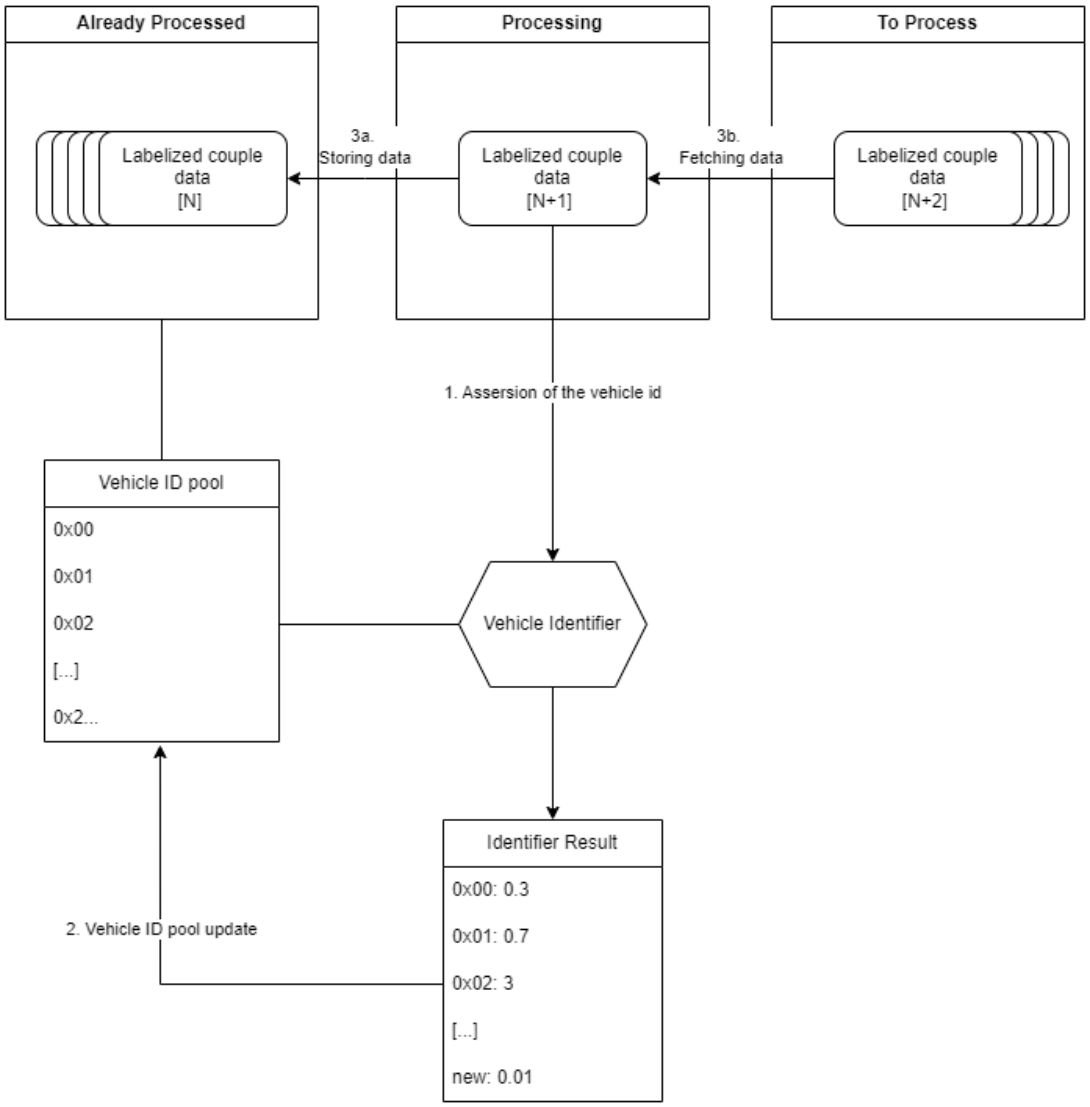


Figure 4.5: Sequential Extension of the Data Pipeline

As shown in Figure 4.5, the pipeline extension is markedly different from the original in that it is inherently sequential. This attribute hinders performance due to the lack of opportunities for parallel processing. Therefore, it was crucial to choose an efficient method for labelling and identifying each vehicle throughout the dataset. There are multiple solutions available such as DeepSort [21] or the creation of a Kalman Filter, a widely used tool for tracking problems [13]. However, setting up a simple Kalman Filter was deemed complicated due to the multimodal nature of the data but could be considered as an area of improvement. Similarly, incorporating the DeepSort tool, which performs well on mp4 files, proved burdensome for the pipeline. The generation of an mp4 video from labelled images was both heavy and unidiomatic given the data's multimodal nature. Moreover, integrating depth information into the video posed an additional challenge.

The chosen solution involved the creation of a state machine that would navigate each labelled pair. This identifier calculates multiple metrics on the pairs to determine if the detected vehicle is one of the id pool (consisting of previously detected ids) or if it's a new one. More details on this process will be covered in its dedicated chapter. This method provides an intuitive and explicit approach, yielding precise metrics that explain the identifier's choices, thereby addressing the current issues surrounding AI explainability [1].

The identifier's data then facilitates the creation of the final dataset.

4.2.3 Structure of the Final Dataset

Utilizing the information derived from the pipeline described above, the final dataset is structured as illustrated in Figure 4.6.

```
dataset
├──<capture_session_id>
│   ├──camera
│   │   └──<image_id>.jpg
│   ├──radar
│   │   └──<radar_id>.radar
│   ├──filtered_radar
│   │   └──<radar_id>.fradar
│   ├──vehicles
│   │   └──vehicles.json
│   └──metadata.json
└──...
```

Figure 4.6: File Structure of the Dataset

In practical terms, each capture session consists of four files:

- camera.tar.gz
- filtered_radar.tar.gz
- radar.tar.gz
- vehicles.tar.gz

Why include radar and filtered radar data?

This decision is admittedly debatable. If one wishes to filter the radar data differently, it is entirely possible. However, the filtering implemented in this study seemed sufficient, and it was decided to share it with interested parties, offering the end user more flexibility.

Why use tar.gz?

Numerous algorithms exist to compress files, typically representing a trade-off between the extent of compression achievable and the resources allocated to it. As discussed in the chapter describing the computational resources needed to run this pipeline, the priority was to minimize the complexity of the compression. Given the dataset's size (several hundreds of gigabytes), excessive compression would render it unusable for the end user.

Compressing each subfolder of the capture session allows users to decompress only the chapters that interest them. For instance, dataset analysis does not necessitate decompressing camera or radar data; the vehicle data suffices.

4.3 RGB Image Processing

The Camera processing step, which is referred to as step **1b.** in Figure 4.4, serves as the focal point of this chapter. This step primarily makes use of the YOLOv5 model [12] to identify the location of vehicles within the camera images. The model is designed to detect a variety of vehicles including cars, trucks, buses, bicycles, and motorcycles.

The model excludes pedestrians from its scope of detection due to the incapability of the radar to glean information from individuals on foot. As a result, these instances are tagged as invalid in the subsequent steps of the data pipeline.

The YOLO (You Only Look Once) algorithm produces a series of predictions for every detected object, furnishing details such as the class of the object, size and position of the bounding box, and the level of confidence the model holds in its predictions. We created a debugging tool that overlays this bounding box information onto the images including class detected and confidence of the model. This is helpful for visual inspection and analysis, as illustrated in Figure 4.7.

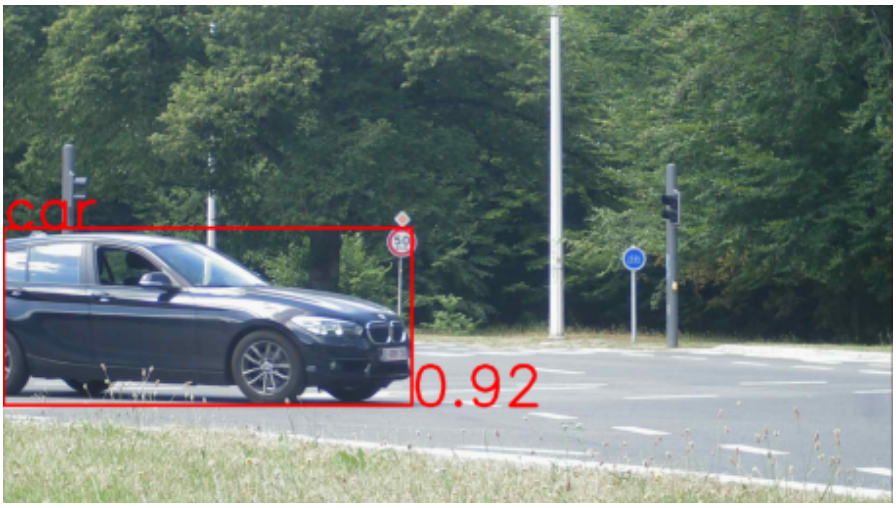


Figure 4.7: Example of YOLO analysis

In the final iteration of the dataset, camera data does not contain any annotations. This decision allows users to reprocess the dataset according to their specific requirements, albeit at the cost of sacrificing debug capabilities.

4.4 Radar Heatmap Processing

This chapter elaborates on the Radar heatmap processing, which encompasses steps **1a.** and **2.** in Figure 4.4. The implementation of these steps presented a significant challenge, largely due to the high level of noise in the raw data, as is evident in Figure 4.8. The logarithmic scale representation of the radar heatmap reveals multiple artifacts and noise that complicate the extraction of vehicle information. This phenomenon is also demonstrated in Figure 4.9, which presents a snapshot of the background noise in the scene.

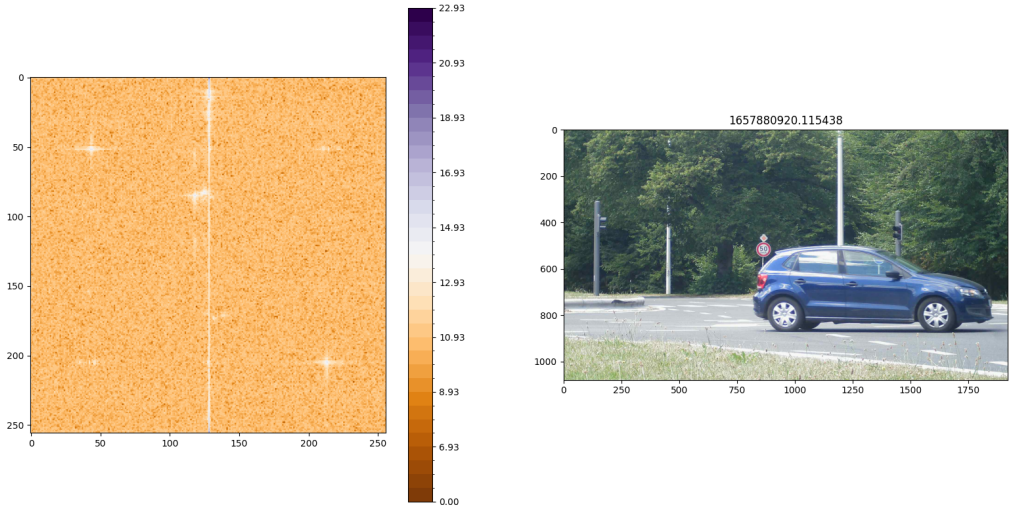


Figure 4.8: Example of data couple with unfiltered radar data. Intensity in dB (for representation purpose) and position of the heatmap referred as the pixel position (0-255).

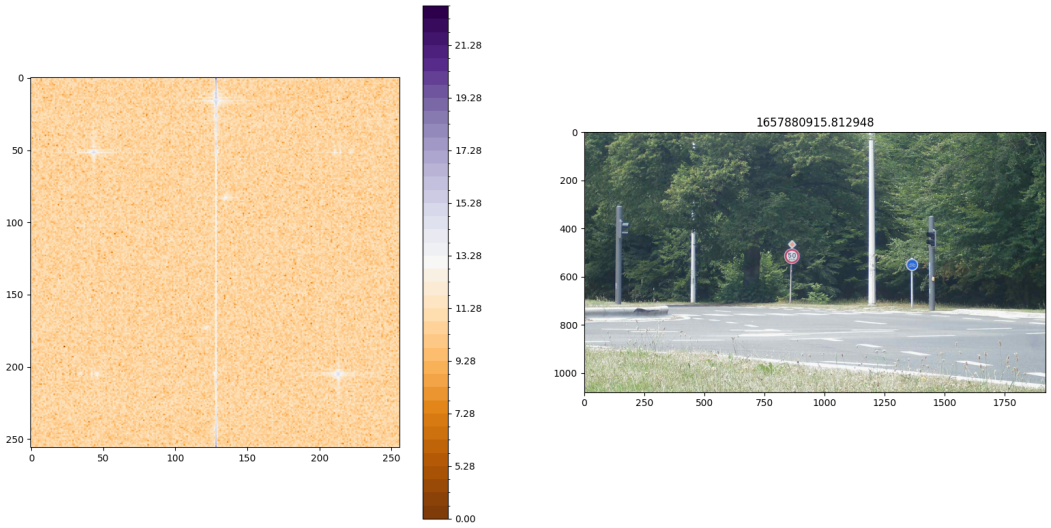


Figure 4.9: Example of data couple background. Intensity in dB (for representation purpose) and position of the heatmap referred as the pixel position (0-255).

4.4.1 Background Filtering

In order to effectively filter the noise present in these images, we applied a sequence of operations as detailed below.

Let us define

$$X : \text{data matrix (not in logarithmic scale)} \tag{4.1}$$
$$B : \text{background data matrix} \tag{4.2}$$
$$M : \text{mean data matrix} \tag{4.3}$$
$$K : \text{convolution kernel} \tag{4.4}$$
$$\tag{4.5}$$

Then the output D of the filtering operation can be expressed as:

$$D = (X - B - M)_+ * K$$

where
- $*$ denotes the 2D convolution operation. - $(x)_+$ is the ReLU (Rectified Linear Unit) function which maps negative elements to 0. - The subtraction in $(X-B-M)$ is element-wise.

The crux of our work lies in the precise determination of the background data matrix, the mean matrix, and the appropriate convolution kernel.

The background data matrix is computed by averaging data couples where no vehicles are detected by the YOLO algorithm, as in Figure 4.9, excluding instances such as Figure 4.8. This approach aids in isolating the static elements within the data.

To tackle the dynamic aspect of the noise, we leverage the mean data matrix. Given a sequence of data matrices X_i, where $i = 1, 2, ..., T$ is the temporal index and each X_i is an $m \times n$ dimensional matrix, we can calculate the temporal mean matrix of absolute values, denoted as $M = [m_{jk}]$. The Root Mean Square value has not been prefered to the absolute value to mitigate the effect of short duration high intensity point over overall lower intensity noise to not filter out too much vehicles. The calculation follows the formula:

$$m_{jk} = \frac{1}{T} \sum_{i=1}^{T} |x_{ijk}|$$

for each $j = 1, 2, ..., m$ and $k = 1, 2, ..., n$, where
- x_{ijk} are the elements of the data matrix X_i - m and n are the dimensions of each data matrix X_i - T is the total number of time points - $|x_{ijk}|$ represents the absolute value of x_{ijk}

Applying this procedure assists in eliminating the static noise component, leading to a decrease in the amplitude of useful information. However, as Figure 4.8 illustrates, the dynamic noise is multiple orders of magnitude beneath the signal amplitude of interest, rendering this reduction insignificant.

The matrix generated from this operation might contain significantly negative elements. Fortunately, our targeted data are empirically always positive, enabling us to apply the ReLU function (as seen in Figure 4.10). The ReLU function effectively thresholds the negative signal components, thereby preserving only the pertinent aspects of the signal.

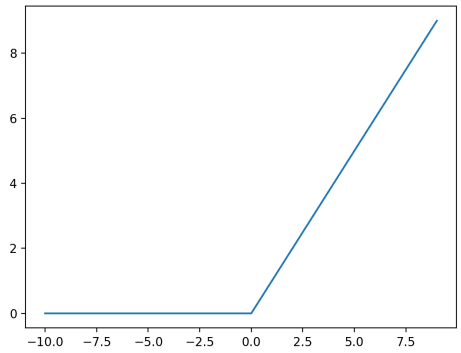


Figure 4.10: ReLU mathematical function

4.4.2 Investigating Convolution

Despite the preceding steps, residual noise may persist in the heatmap, as depicted in Figure 4.11 and Figure 4.12.

As observable in these figures, the radar profile of a vehicle can occasionally appear disjointed. This is likely due to the varied reflective properties of different parts of the vehicle – such as metal, glass, and rubber – that respond differently to the incident radar wave, thereby contributing independently to the backscattering. As a consequence, the vehicle might manifest as multiple distinct elements in the radar data, complicating the task of accurately identifying and counting vehicles from the heatmap alone.

To address this issue and enhance the coherence of vehicle profiles in the heatmap, a convolution operation was introduced. This mathematical technique, often

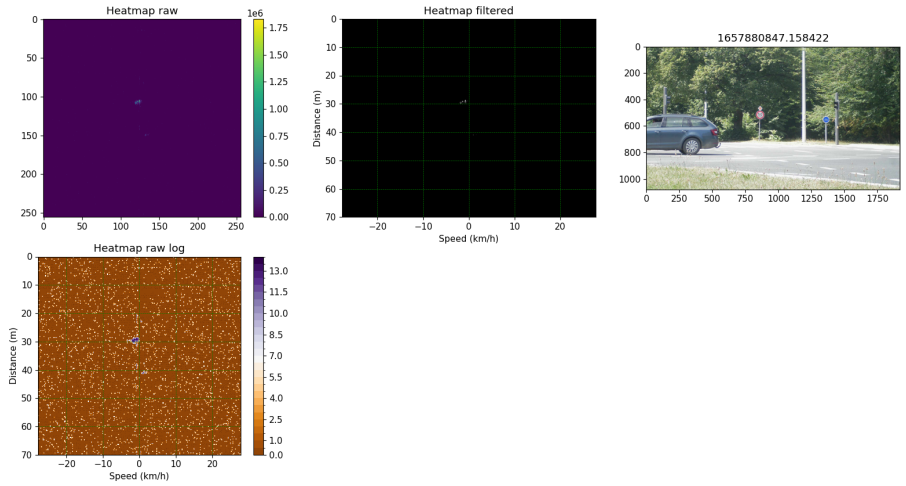


Figure 4.11: Data state example before convolution

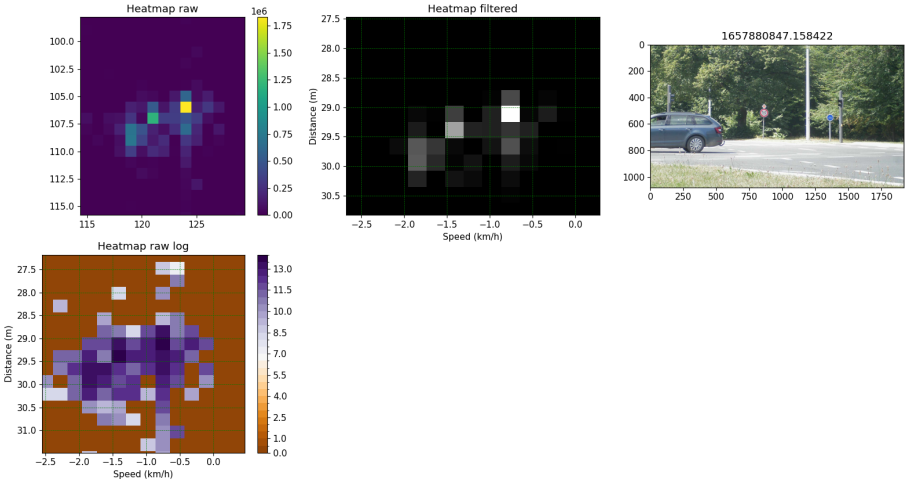


Figure 4.12: Data state example before convolution, zoomed

employed in signal processing and image analysis, can smooth the heatmap and unify the disjointed elements that are, in reality, part of the same vehicle. By applying the convolution operation, the radar reflections from various parts of a vehicle can be merged or smoothed, resulting in a more unified and recognizable vehicle profile. This not only facilitates the detection and counting of vehicles but also potentially enables a more detailed analysis of the vehicles' characteristics.

It's noteworthy that the choice of the convolution kernel – the specific pattern of weights applied in the convolution operation – is crucial in this process and may require careful tuning to achieve optimal results.

Two kernels were considered in our study, as suggested by François Ledent [15], specifically, the Triangular Kernel and the Gaussian Kernel. We focused our analysis on three scenarios:

- Raw data
- Triangular Kernel 3x3
- Gaussian Kernel 11x11, with a standard deviation, $\sigma = 0.5$

Triangular Kernel Consider k as a kernel of size $n \times n$ and r as a one-dimensional array of size n defined by:

$$r_i = \frac{|n + 1 - 2i|}{n^2 - n + 2} \quad \text{for} \quad i = 1, 2, ..., n$$

We then define the two-dimensional symmetric triangular kernel matrix K as:

$$K = r \otimes r$$

where $\otimes$ represents the outer product of r with itself. The kernel is normalized such that $\sum_{i=1}^{n} \sum_{j=1}^{n} K_{ij} = 1$ to maintain scale.

Gaussian Kernel Let m, n, and σ denoting kernel size and standard deviation respectively.

The Gaussian kernel G of size $(2m + 1) \times (2m + 1)$ can be derived as follows:

$$G_{ij} = \frac{1}{Z} \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right)$$

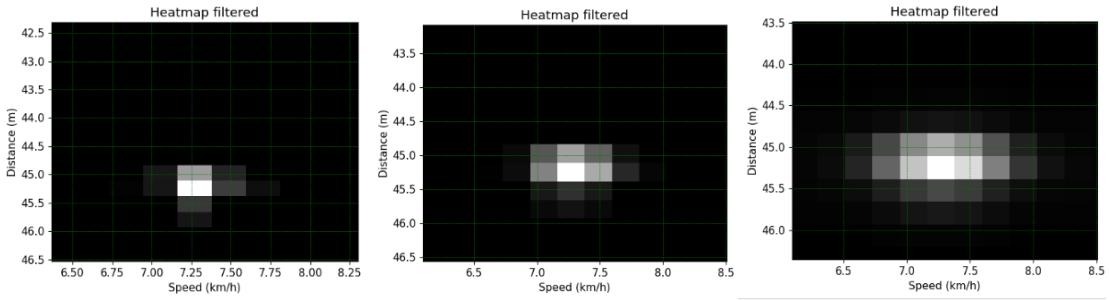
for each $i, j = -m, ..., m$, where Z is a normalization constant ensuring that $\sum_{i=-m}^{m} \sum_{j=-m}^{m} G_{ij} = 1$.

We then define the correlation kernel K of size $(2n + 1) \times (2n + 1)$:

$$K_{ij} = \begin{cases} G_{(i-n),(j-n)} & \text{if } |i - n| \leq m \text{ and } |j - n| \leq m \\ -\frac{1}{S} & \text{if } K_{ij} < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

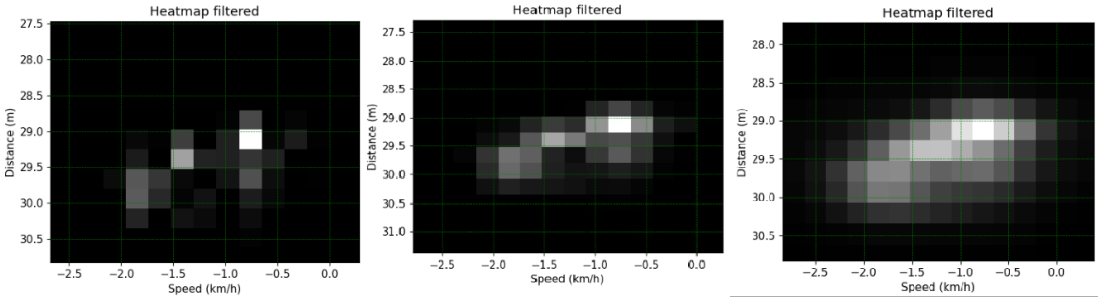
for each $i, j = 0, ..., 2n$, where $\epsilon = 1 \times 10^{-10}$ and S is the count of elements in K that are less than ϵ.

The parameters for these filters were chosen empirically at first, and subsequently refined via optimization (to be presented later). As demonstrated in Figure 4.13 and Figure 4.14, the triangular kernel may seem optimal in the first case. This would be beneficial since it would lessen the computational load. However, as depicted in Figure 4.14, the triangular kernel performs poorly in our situations due to the energy dispersion it induces, prompting us to narrow it down and reduce its robustness to vehicle with divided envelope. The Gaussian kernel, though more computationally intensive due to its size, provides more robust filtering.



((a)) Raw radar profile **((b))** Convoluted profile using triangular kernel **((c))** Convoluted profile using gaussian kernel

Figure 4.13: Comparison of kernels in good case



((a)) Raw radar profile **((b))** Convoluted profile using triangular kernel **((c))** Convoluted profile using gaussian kernel

Figure 4.14: Comparison of kernels in bad case

An exhaustive investigation of the error detection rate based on the kernel shape was conducted. An error is considered when the YOLO and radar pipeline results do not yield the same vehicle count for a given couple. Based on this metric, Figure 4.15 was produced. This figure illustrates the error rate as a function of the CFAR

threshold (explained in chapter 3.2.2) and the kernel shape. In the case where no vehicle is present in the camera image ("Error count 0 vehicle"), the radar heatmap pipeline is expected to detect none. While the kernel shape in this case slightly shifts the optimal threshold value, it doesn't affect the absolute minimum of error. However, when there is one vehicle in the image ("Error count 1 vehicle"), the optimal threshold value with the Gaussian kernel is lower than with the triangle, which fails to reduce the error compared to the raw data. This led us to select the Gaussian kernel.

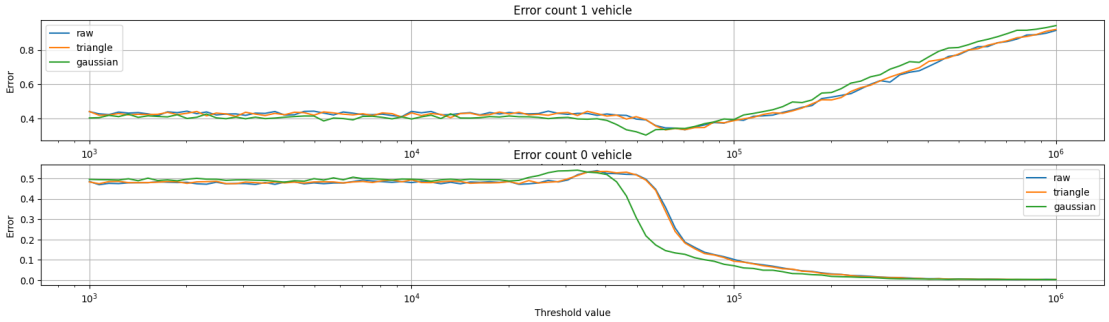


Figure 4.15: Comparison of the error rate per kernel shape

4.4.3 Optimizing the Gaussian Kernel

While the kernel parameter optimization was conducted considering the error rate as described in the preceding chapter, one significant limitation of this error metric is that it does not account for the pipeline's performance. Specifically, it does not reflect the fact that larger kernels result in larger processing times.

Figure 4.16 displays several features of the dataset and the data processing pipeline. It's notable that the curves for "0 vehicle" scenarios closely resemble those for "All samples." This observation indicates that a significant portion of the initial dataset consists of images in which YOLO fails to detect any vehicles, which are not beneficial to the pipeline's performance.

The "1 vehicle" scenario appears to reach a minimum error rate when using a 13x13 kernel size and a standard deviation (σ) of 0.5. However, this error metric doesn't account for the increase in processing time that comes with larger kernels. When only one vehicle appears in the heatmap, a larger kernel size might appear to fit the case well. Still, this approach doesn't take into account the associated computational costs, leading to a potential overfitting situation.

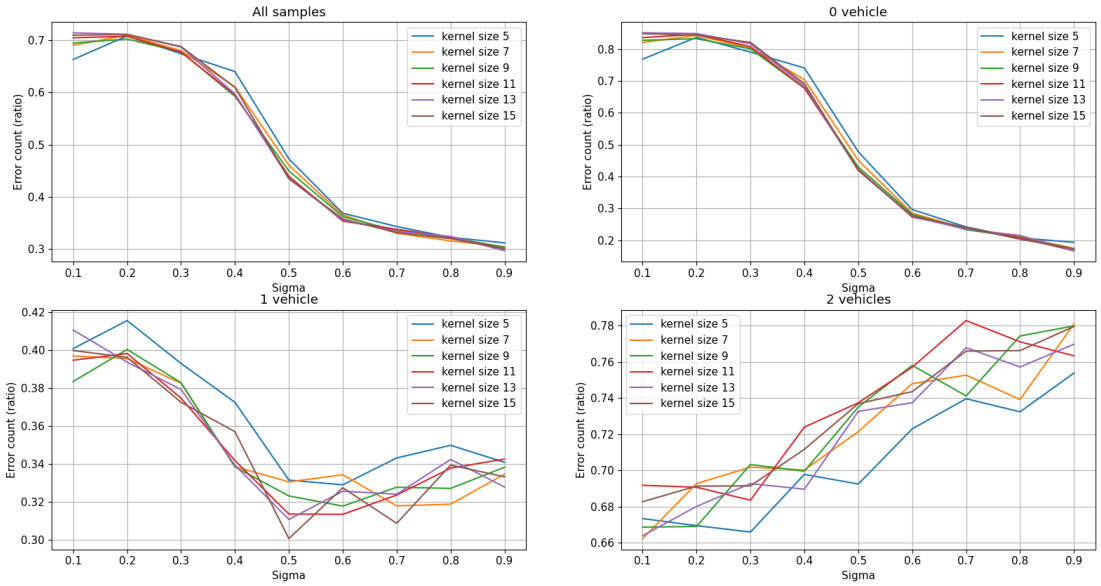


Figure 4.16: Impact of kernel size and standard deviation on the pipeline's performance in various vehicle count scenarios

Interestingly, we observe that a 5x5 kernel size tends to yield higher error counts across all standard deviations. To achieve a reasonable compromise between performance and accuracy, we selected an 11x11 kernel size. With this kernel size, the minimum error rate was also achieved with $\sigma = 0.5$.

Figure 4.16 also highlights the pipeline's inability to process situations involving two vehicles efficiently. The error count in such scenarios is significantly higher than in one-vehicle situations (over 66% versus approximately 35%). This disparity may arise because multiple vehicles entering the camera's field of view may follow one another closely (i.e., within the same lane), resulting in similar radar profiles and making it challenging to distinguish between them. As a result, the filter's performance characteristics differ significantly between one-vehicle and two-vehicle situations. However, as our focus is limited to one-vehicle situations, this discrepancy isn't a major concern in this study.

4.4.4 CFAR

As explained in the Litterature review, the CFAR is a widely used algorithm for extracting useful information in a radar heatmap. Our implementation is the GO-CFAR allowing better robustness of the system to the noise.

The implementation of this algorithm is as follow:

Let S be the input data matrix and K_G, K_D, K_H, and K_B be the kernels defined as:

$$K_{G_{ij}} = K_{D_{ij}} = K_{H_{ij}} = K_{B_{ij}} = \begin{cases} \frac{1}{(2m+1)(l-k)} & \text{if within the respective directional bands} \\ 0 & \text{otherwise} \end{cases}$$

for each $i, j = 0, ..., 2l$.

Then, calculate the convolution of S with each kernel:

$$M_G = S * K_G, \quad M_D = S * K_D, \quad M_H = S * K_H, \quad M_B = S * K_B$$

where $*$ denotes the 2D convolution operation.

Then, define the matrix of maximum means, $MaxM$, by taking the maximum value among M_G, M_D, M_H, and M_B at each pixel location.

Calculate the CFAR magnitude matrix, $Magn_{CFAR}$, by subtracting 1.1 times $MaxM$ from the convolution of S with a 4×4 averaging kernel:

$$Magn_{CFAR} = \left(S * \frac{1}{16} \mathbf{1}_{4\times4} \right) - 1.1 \times MaxM$$

where $\mathbf{1}_{4\times4}$ is a 4×4 matrix of ones.

Next, calculate the CFAR threshold, ϵ:

$$\epsilon = \max\left(\text{threshold}, 1 + 0.5 \times (\max(Magn_{CFAR}) - 1)\right)$$

Finally, generate the binary spotted matrix, *Spotted*, where each pixel is marked as 1 if its corresponding $Magn_{CFAR}$ value is greater than or equal to ϵ, and 0 otherwise.

From the *Spotted* matrix, we can determine which pixels in the heatmap are detected as vehicles. In this matrix, a value of 1 indicates that the corresponding pixel is detected as a vehicle, while a value of 0 indicates otherwise. A visual representation of the *Spotted* matrix can be seen in Figure 4.17, where white pixels represent a value of 1 and black pixels represent a value of 0.

4.4.5 Island Algorithm

The Island Algorithm problem arises in the context of image processing and computer vision, specifically when it comes to identifying and separating distinct components or "islands" in a binary image.

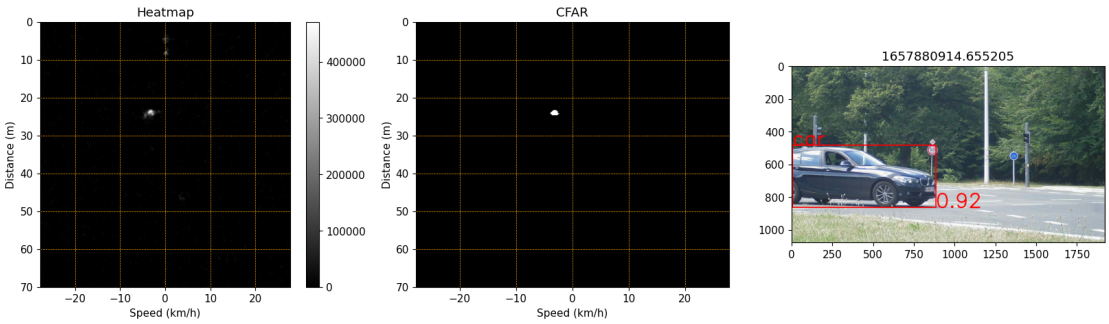


Figure 4.17: Example of the result of the CFAR

In the case of the Spotted matrix, we're interested in identifying distinct vehicles, which can be seen as distinct "islands" of pixels with a value of 1 in a "sea" of pixels with a value of 0.

An island in the context of a binary image is a group of connected pixels with the same value, typically 1. Two pixels are said to be connected if they are adjacent to each other either horizontally, vertically, or diagonally.

The Island Algorithm problem can be solved using various methods. One of the simplest methods involves a labeling algorithm, which can be described as follows:

1. Scan the Spotted matrix pixel by pixel (left to right, top to bottom).
2. When you find a pixel of value 1, check its neighbors (the pixels already visited). If none of its neighbors has a label (i.e., all are 0), assign a new label to this pixel. If one or more neighbors have labels, assign the smallest of these labels to the pixel being processed, and record equivalences between labels if there are neighbors with different labels.
3. After the whole image has been scanned and labeled, resolve the equivalences: for each pixel, if it has a label that is in the equivalence list of another label, replace its label with the smallest label of the equivalent labels.

After the application of the Island Algorithm, each detected vehicle corresponds to a uniquely labeled island in the Spotted matrix.

To locate each vehicle within the heatmap, we can determine the pixel with the highest intensity within each island. This position is considered the singular position representing the vehicle. Thus, the number of vehicles corresponds to the number of distinct islands detected by the Island Algorithm.

The corresponding speed and distance of each vehicle can then be derived from this singular position using hardware-specific calibrations and measurements.

4.5 Data merging

This part refer to the data merging part of the data pipeline (steps **3.** and **4.**). This part is particularly important as it is in charge to verify if the couple will be valid (ie. if the data of each modalities are associable regarding the context and limitation of the pipeline) or will be dropped. If the couple is flagged as valid, it will be labeled.

Multiple factor are here to ensure if the couple is valid. The first one is the vehicle count in the radar data and the camera data. If the count is different or different from 1 we do not keep the system. It would also be possible to drop the couple if a minimum energy in the heatmap is not reached, but this has not been implemented. This would allow to prevent false detection in case of additive interference. We choose not to implement it as the case seemed rare enough to be negligible (it has not been seen while manually going through the dataset).

An very important point of decision for keeping or not the couple is the calculation of a characteristic length of a detected vehicle. In the figure 4.18 both modalities detect one vehicle, but the vehicle detected by the radar is erroneous. The vehicle is obviously not at 4 meters in reality. To prevent theses detection to polute the final dataset a characteristic lenght metric has been implemented.

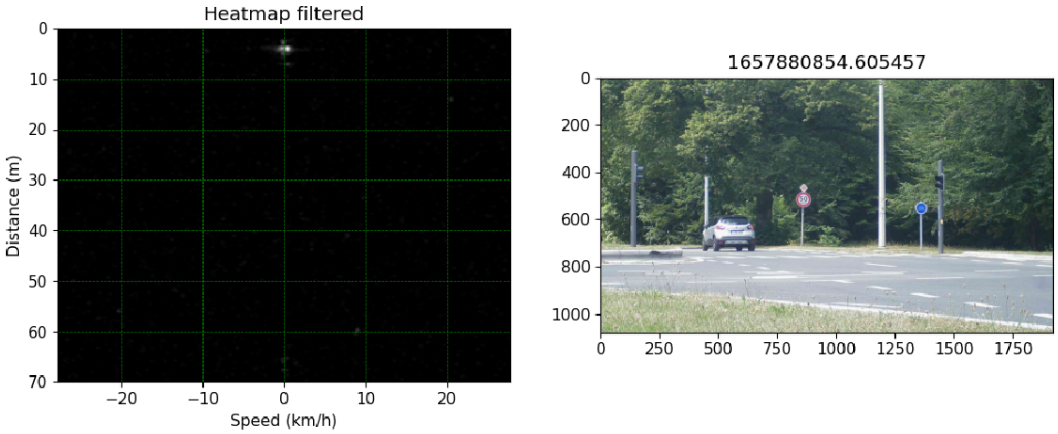


Figure 4.18: Wrong detection due to artefact

Given the 2D position $\mathbf{P}_{2D,i}$ of the ith corner of the bounding box in the image and the detected distance d_i from the radar, we can compute the 3D position $\mathbf{P}_{3D,i}$ of each corner of the bounding box. If we define $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ as these 3D positions, we can form a parallelogram. The area of this parallelogram is computed as:

$$Area = \|(\mathbf{A} - \mathbf{B}) \times (\mathbf{C} - \mathbf{A})\|$$

From the area, we can derive a characteristic length L of the vehicle as the square root of the area:

$$L = \sqrt{Area}$$

This calculation can be interpreted intuitively in the figure 4.19.

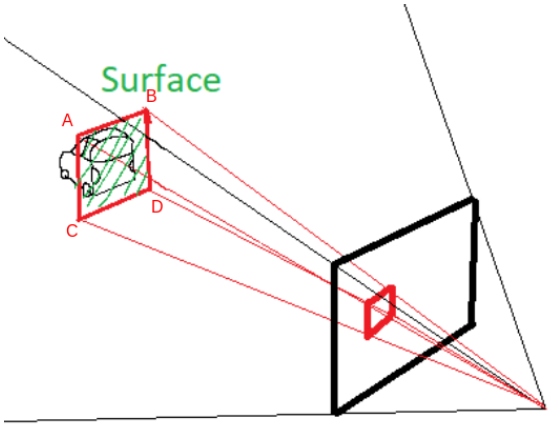


Figure 4.19: Characteristic length calculation schema

The analysis of this metric across a significant subpart of the dataset would allow to choose a threshold. This analysis can be seen in fig 4.20.

This study revealed another phenomenon not taken into account at that moment. For distance around 10-15 meters and 20-25 meters, we can see a concentration of widely different characteristic length. This happens when vehicle enter (or leave) the field of view of the camera depicted in Figure 4.21. Filtering theses case is also a good idea as the center of the vehicle is assumed to be the center of the bounding box, thus implying a rapid (virtual) acceleration of the vehicle entering the field of view which is not wanted. We chose a value of 1.5 for this threshold to filter theses cases.

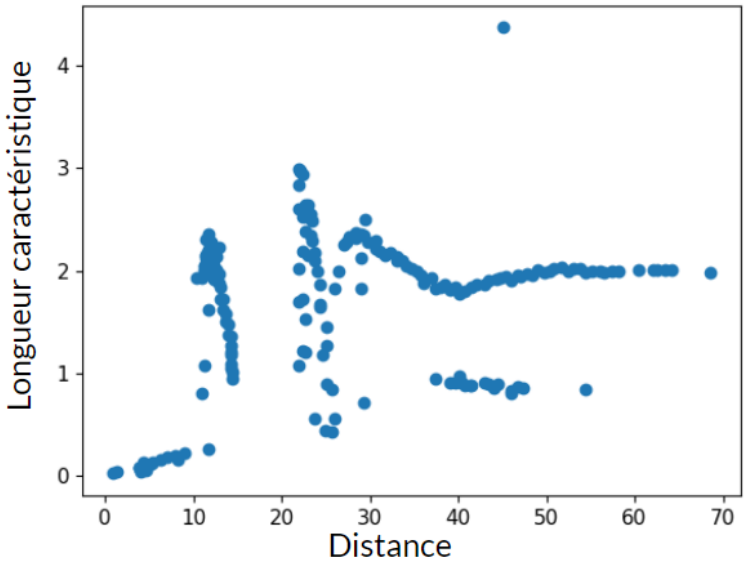


Figure 4.20: Characteristic length representation across 10% of the dataset

Figure 4.21: Vehicle entering the camera field of view

If a couple pass all of these filter, it is considered as valid and the 3D position of the vehicle is calculated using the following equation as described in the literature review part.

$$yaw = \tan^{-1}\left(\frac{x - c_x}{f}\right) \tag{4.6}$$

$$pitch = \tan^{-1}\left(\frac{y - c_y}{f}\right) \tag{4.7}$$

$$\begin{bmatrix} x_{abs} \\ y_{abs} \\ z_{abs} \end{bmatrix} = \begin{bmatrix} distance \cdot \cos(pitch) \cdot \cos(yaw) \\ -distance \cdot \sin(yaw) \\ distance \cdot \sin(pitch) \end{bmatrix} \tag{4.8}$$

4.6 Sequential Data Analysis

The data merging phase of our pipeline is followed by the sequential analysis, a key component that ensures consistent vehicle identification across diverse data sets. The role of this procedure is to maintain the precision and pertinence of our labels across multiple captured instances of the same vehicle.

The main input data for this phase consist of several pairs of camera RGB images and radar heatmaps, each pair featuring a labeled vehicle as per our previous discussions. This vehicle label offers an extensive set of data concerning the vehicle's state, such as its position in 3D space, velocity, distance from the observer, among other parameters. In addition, the position of the vehicle in the image is precisely located and annotated.

The sequential pipeline is an extension of the parallel labeling pipeline, built to handle these labeled data pairs in a sequential manner. It aims to ascertain if the vehicle identified in a particular data set (designated as "N") corresponds to the vehicle identified in the previous data set (denoted as "N-1"). This continuity is crucial for monitoring the same vehicle across multiple instances.

To achieve this identification, we use a state machine that operates on a pool of vehicle IDs (refer Figure 4.22). Whenever a vehicle is detected in a dataset, its ID is stored in this pool, along with metadata such as the timestamp of the last observation, the vehicle's position, and bounding box details.

As each data set is processed, the state machine cycles through a set of metrics. For each metric, the machine compares the current vehicle data with each ID in the pool, utilizing a chosen function (acting as the metric) to generate a 'score'. This score reflects the likelihood of a match between the current vehicle data and an existing ID, with lower scores indicating higher probability.

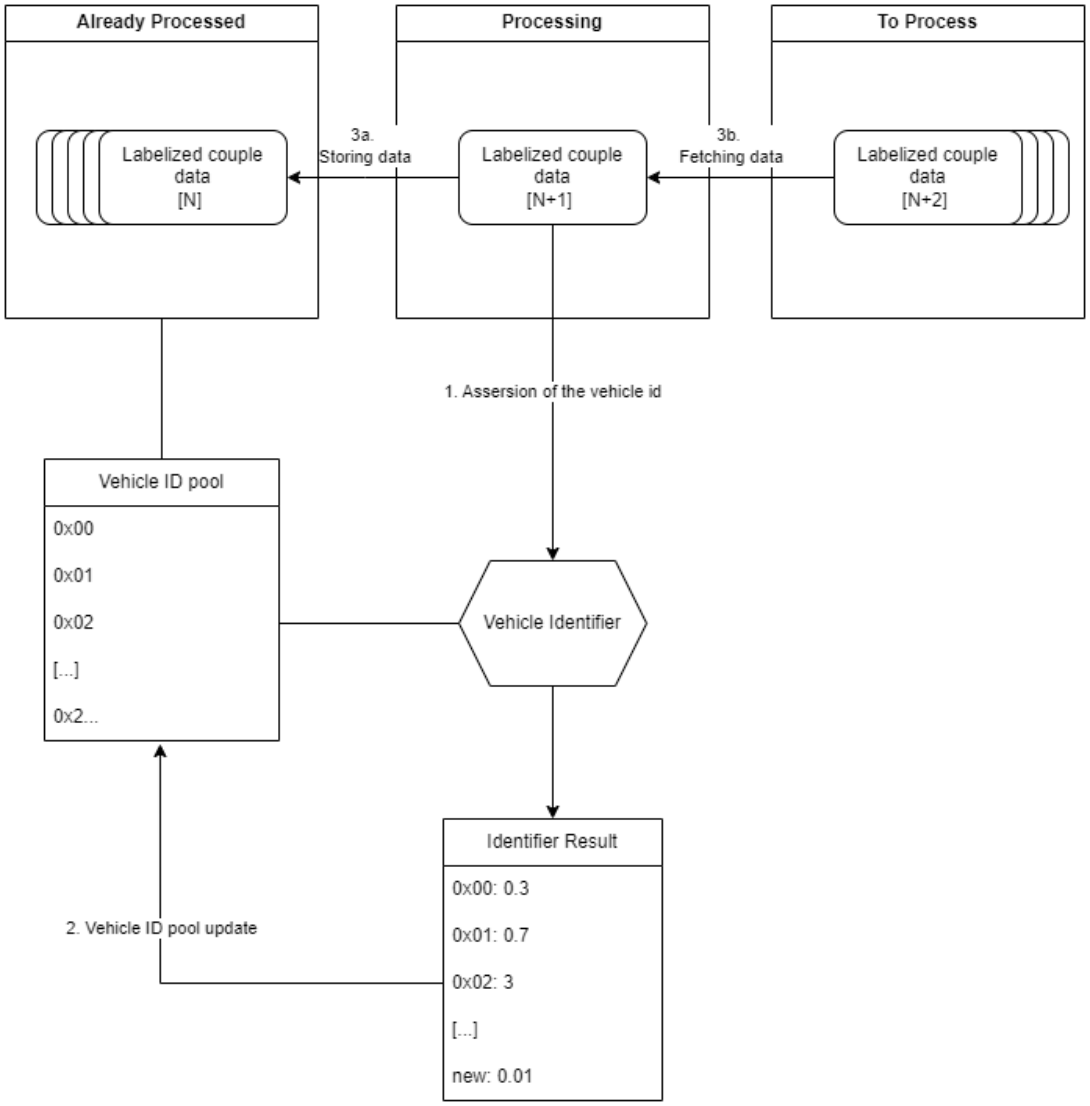


Figure 4.22: Sequential Pipeline

These scores are compiled for each ID through a weighted average. The weights of these averages were determined via trial and error to optimize performance. If the score for a new ID is the lowest, it implies that the vehicle has not been observed before, and this new ID is added to the pool. Conversely, if an existing ID gives the lowest score, the corresponding metadata are updated to mirror the current observation.

The process then continues to the next data set until the end of the recording session. This iterative and efficient system allows for consistent vehicle tracking across multiple data sets, thereby enhancing the accuracy and utility of our labels.

The different metrics used are:

- **Radar prediction:** knowing the previous position in the radar heatmap we can estimate knowing the time delay the next position approximately.
- **3D Euclidean distance:** knowing the previous position in 3d the euclidean distance help to know if the vehicle has moved a plausible distance given its expected velocity.
- **Overlap area:**With the previous bounding box associated with the ID and the current bounding box, a high overlap suggests continuity and supports the hypothesis that the two observations represent the same vehicle.
- **Time delay:** comparing the timestamp of the current observation with the time the vehicle associated with the ID was last seen. A longer delay reduces the likelihood that the current observation pertains to the same vehicle, helping account for scenarios where a vehicle may have exited and then re-entered the sensor's range.

Our sequential analysis is a resilient system that allows us to track vehicles accurately across multiple data sets. This enhances the consistency of our vehicle labeling and significantly improves the overall accuracy and utility of the resulting dataset. Further refinements and additional metrics can be considered to enhance this process based on the specific characteristics and requirements of the dataset.

The different parameters of these metrics were manually optimized through trial and error using tools specifically designed for this task, such as the overlay that illustrates the metric score for a particular image (refer Figure 4.23). This overlay aids the user in understanding which metric seems too strong or weak in specific cases. However, choosing the correct parameters was challenging and likely suboptimal.

The qualitative metric used to assess the overall performance of the identification pipeline was the characteristic length introduced in the previous chapter. This length should follow approximately a concave curve. When a vehicle enters the camera's field of view, its characteristic length will increase and decrease when it leaves. This allows for a quick visual assessment of the pipeline's integrity, as

Figure 4.23: Overlay example demonstrating the identifier's decision-making process by explaining the score of each metric for the current couple.

shown in figure 4.24 and figure 4.25. The first figure clearly shows inadequate reuse of identifiers (a continuous line represents the same vehicle).

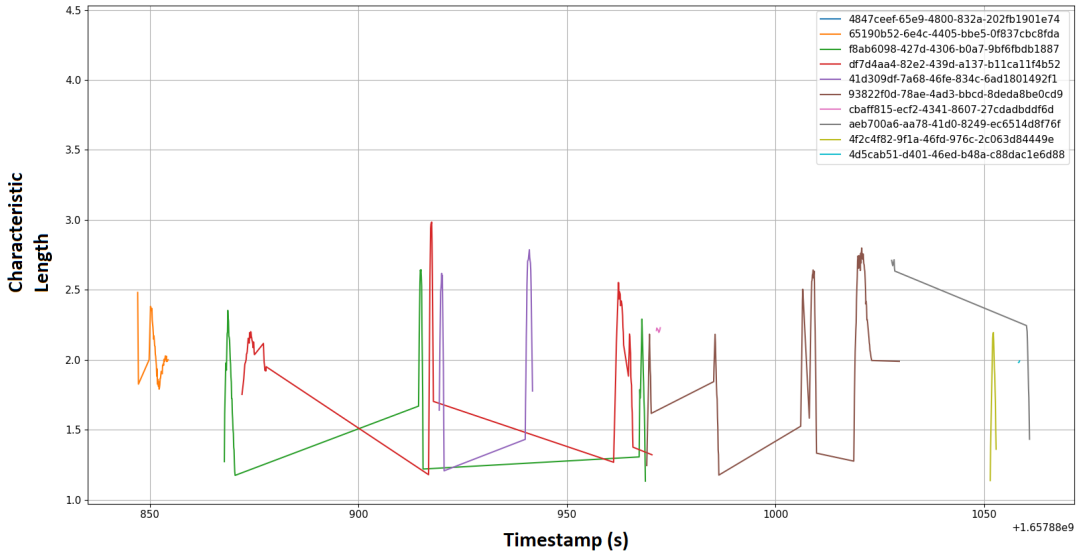


Figure 4.24: Example of poor vehicle separation

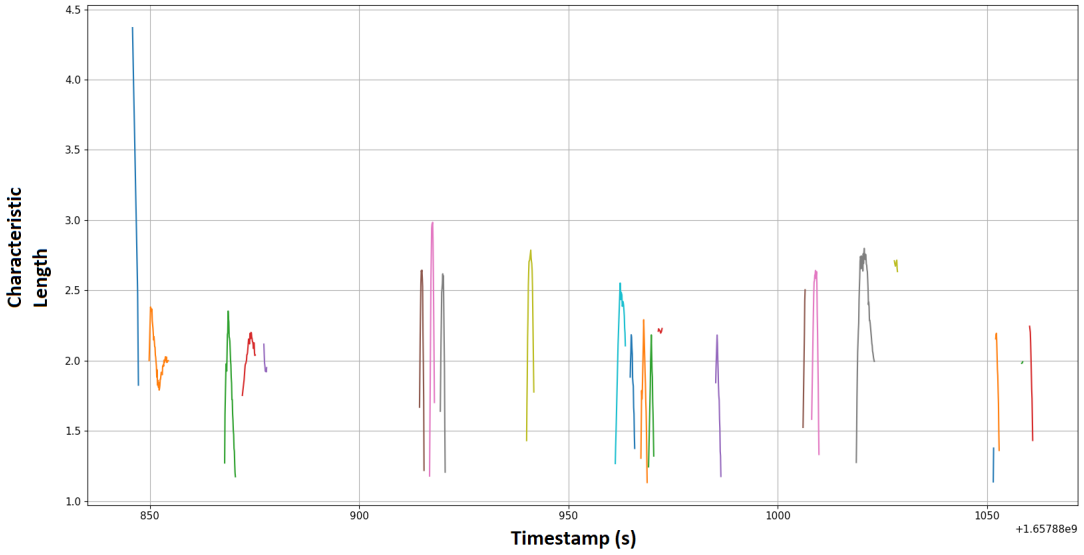


Figure 4.25: Example of effective vehicle separation using final parameters

4.7 Integration of the Transformer Model

While an implementation of the transformer model may not be feasible due to time constraints, this chapter will explore possible methods to incorporate it, particularly in relation to trajectory forecasting as outlined in the literature review [9]. Two approaches may be employed, each varying based on their input.

4.7.1 Employing Labels as Input

The first approach involves using labels as the input. This places the responsibility of integrating multimodal data on the existing pipeline. If actual implementation were to occur, this pipeline would need to operate in real time, necessitating adjustments to enhance performance. Despite the directness and expedited implementation of this approach, it does shift the burden of adequate modality information integration onto the operator and does not fully utilize the transformer's ability to analyze and learn from the full context.

Nevertheless, given its limitations, this approach appears viable as an initial step. Providing sufficient context is crucial for the effective operation of the transformer. An input of at least five or more samples for each vehicle before attempting to forecast their positions could be a suitable starting point, to be refined based on empirical results [9]. Additional features, such as the bounding box area, may also be included in the input to enhance the transformer's forecasting ability [7].

4.7.2 Camera and Filtered Heatmap as Input

The second approach entails using camera data and filtered (or raw) radar heatmap as inputs. This could potentially enable a transformer network to extract the positions of multiple vehicles from a single image, overcoming the limitations of the current pipeline. Given that a human could associate specific radar profiles with corresponding vehicles using common sense and depth perception, a suitably designed transformer architecture could potentially achieve the same.

The most straightforward implementation would entail creating suitable embeddings for the camera image and radar data (or potentially the raw data signal instead of the heatmap). With these embeddings, various multimodal merging strategies can be tested to find the optimal approach [18]. However, a significant amount of training data would be required to develop a reliable model, potentially exceeding the size of the dataset used to train GPT-3, which is estimated at least 500 GB of text data [17]. Given the larger file size of image data, the feasibility of this approach may be limited.

As a potential solution, a layered model could be considered to distribute the computational load and improve model performance. For instance, utilizing YOLO for initial image analysis may be beneficial, and could potentially be applied to the radar heatmap as well. While these considerations fall outside the scope of this thesis, they present promising avenues for future research.

4.7.3 Utilizing Pretrained Models

The principle of transfer learning, introduced earlier in this thesis, could be a valuable tool in the integration of a transformer network. In 2020, BERT demonstrated promising results in trajectory forecasting tasks through transfer learning [9]. By 2023, superior models such as GPT-4 (closed source) and LLaMA (open source with leaked weights) became available [19]. Leveraging these models with either the label as input or raw modality as input approach could potentially yield satisfactory results, whilst being less time-consuming and resource-intensive. Considering the limited availability of training samples (as discussed in the experimental results chapter), this approach may prove especially beneficial.

Chapter 5

Experimental Results

5.1 Dataset Characteristics

This chapter provides an examination of the dataset created for this study. The dataset comprises 992 distinct vehicles, with a total of 10504 valid couples.

However, the count of usable vehicles is significantly lower. If a minimum of 10 samples per vehicle is required to provide ample context to the transformer network for trajectory prediction, only 409 vehicles are deemed usable. Figure 5.1 illustrates this phenomenon, highlighting the high number of vehicles with fewer than 10 samples. The average sample count per vehicle across the dataset is 10.58.

Figure 5.2 presents a statistical analysis of the sample count per vehicle per session. As observed, several sessions, such as 2,3,5,7, have a small number of vehicles with more than 10 samples. Detailed statistical information is provided in table 5.1.

Session	Mean	Count	Min	Max	Q10	Q25	Q50	Q75	Q90
0	13.3	389	1	64	2.0	4.0	12.0	19.0	28.2
1	9.3	215	1	31	2.0	4.0	7.0	13.0	19.0
2	5.1	61	1	17	1.0	2.0	3.0	7.0	12.0
3	3.1	29	1	20	1.0	1.0	2.0	3.0	4.8
4	9.6	190	1	61	2.0	4.0	8.0	14.0	19.0
5	5.4	31	1	36	2.0	3.0	3.0	5.0	10.0
6	12.8	70	1	48	2.0	4.0	10.5	18.0	26.0
7	2.0	7	1	5	1.0	1.0	1.0	2.5	3.8

Table 5.1: Statistics of the distribution of number of sample per vehicle dataset

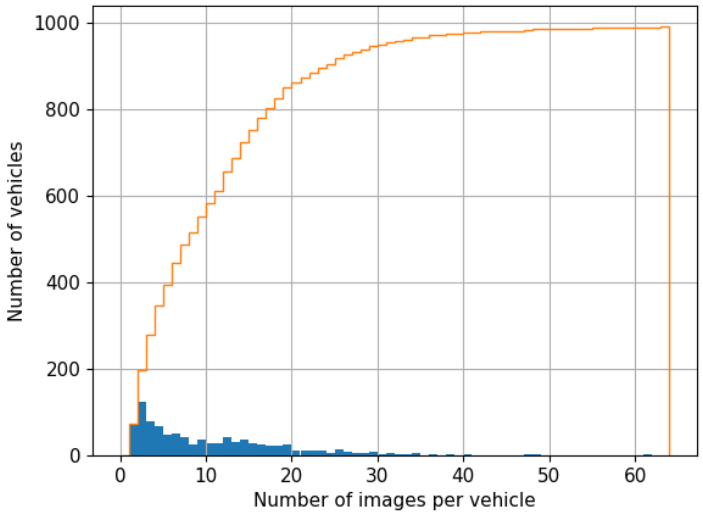


Figure 5.1: Accumulative histogram illustrating the number of vehicles having at least a certain quantity of samples

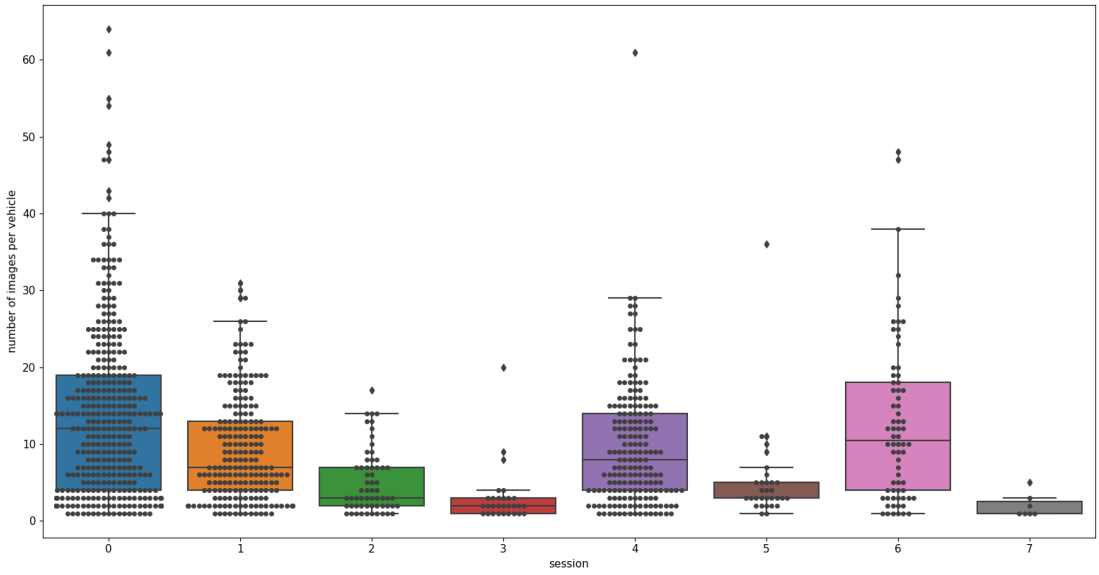


Figure 5.2: Statistical distribution of sample count per vehicle per data capture session

Figure 5.3 depicts a reverse cumulative histogram per session. This representation helps evaluate the potential usefulness of a given session depending on the number of samples required per vehicle.

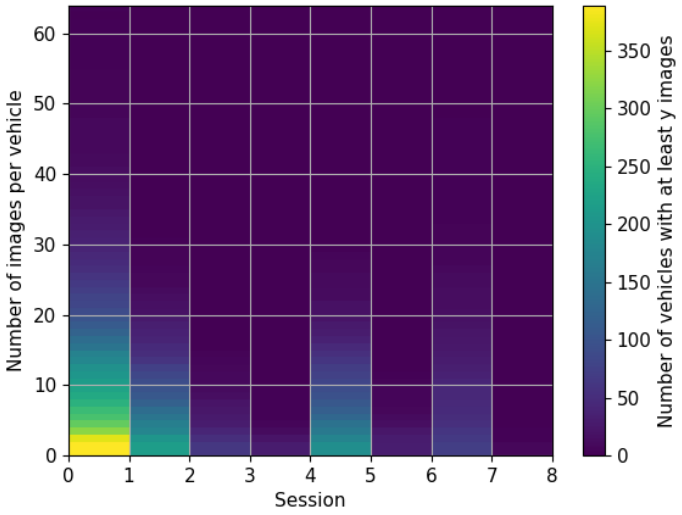


Figure 5.3: Accumulative histogram showing the number of vehicles having at least a certain quantity of samples per session

5.2 Interpretation of Results

The statistical analysis reveals significant discrepancies in the distribution of samples across sessions and vehicles. This uneven distribution may affect the efficiency of the proposed model, as it relies heavily on sample count per vehicle to predict trajectories accurately.

As depicted in Figure 5.1, many vehicles have fewer than 10 samples. Given the requirement for a minimum of 10 samples per vehicle, more than half of the vehicles in the dataset are considered unsuitable for training the model. This lack of data may result in inadequate learning and potentially poorer model performance.

Table 5.1 and Figure 5.2 further show that sessions 2, 3, 5, and 7 contain few vehicles with more than 10 samples, which could further limit the scope of training and testing data.

In conclusion, these results underscore the need for more comprehensive data collection that covers a more extensive range of vehicles and provides a larger sample count per vehicle. Nevertheless, the existing data could still serve as a starting point for initial model development and refinement, and further data could be collected iteratively as the model evolves.

5.3 Establishing Approximate Ground Truth Annotations

To qualitatively assess the quality of the dataset, it was necessary to approximate the ground truth annotations. Two methods were employed for this purpose: using a laser rangefinder and leveraging Google Maps' top-down view.

The laser rangefinder was used to gauge the distance of specific points within the static context. This method allows us to reconstruct the scene and compare the road boundaries with the computed vehicle trajectories. Figure 5.4 presents various trajectories from the initial data capture session, with the road sides depicted in black.

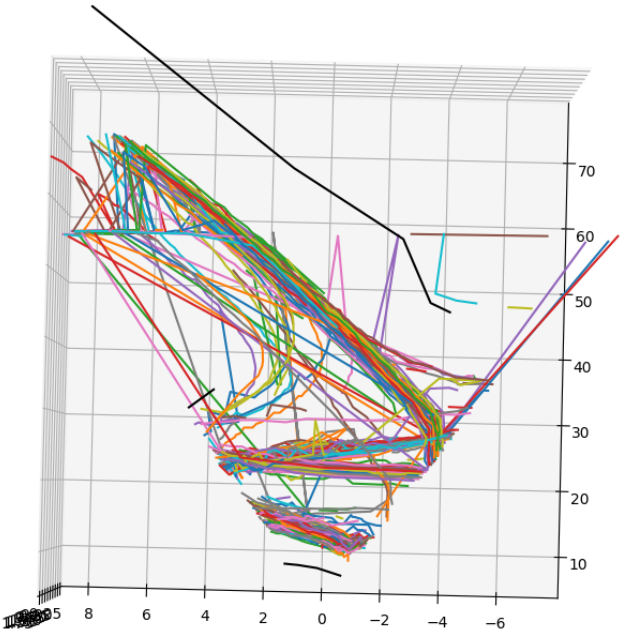


Figure 5.4: Trajectories from the first data capture session, with static objects (in black) in a top-down view

However, when comparing the black road outlines and trajectories with the Google Maps top-down view, a lack of precise alignment is evident. Several factors may contribute to this discrepancy, including the measurement approximations, the

uncertain position of the hardware during data recording, and the difficulty in pinpointing precise points with a rangefinder in low-light conditions at approximately 50 meters without a tripod. The chosen rangefinder position is approximately accurate within a 3-meter radius. Also, the device's height was not precisely defined, which could lead to inaccuracies when calculating the closest distance using the Pythagorean theorem.

A more reliable quantification method involves utilizing Google Maps' top-down view and aligning vehicle trajectories with different road lanes. Figure 5.5 demonstrates this approach. Although this method remains qualitative, it illustrates that the trajectories appear logical. Some anomalies can be observed, such as certain color-coded lines (each representing the same vehicle) appearing to "jump" across the map. This issue might be attributed to challenges in vehicle identification, with the jumps seemingly occurring along the radar axis, suggesting a potential issue with the 2D bounding box area metric.

Figure 5.5: Trajectories overlaid on Google Maps' top-down view

5.4 Interpretation and Discussion

The discrepancies observed in ground truth approximation methods underscore the inherent challenges in verifying the dataset's quality. The use of a laser rangefinder, despite its apparent precision, is prone to errors due to measurement approximations, operational challenges, and lack of precise information about the device's placement. On the other hand, overlaying vehicle trajectories on Google Maps offers a more intuitive visualization of the data but isn't devoid of anomalies, likely due to identification difficulties.

Nevertheless, these methods provide a crucial qualitative assessment of the dataset's quality and reveal potential issues that could be addressed in future data collection efforts. For instance, ensuring precise device positioning and improved measurement techniques during data recording can enhance data quality. Also, refining the vehicle identification method, potentially by adjusting the 2D bounding box area metric, could reduce anomalies observed in the trajectory data.

5.5 Analysis of Processing Time and Computational Resources

The entire processing pipeline executed for approximately 46,000 seconds (around 13 hours) on an Intel 13600K paired with an NVIDIA 3070 graphics card. The breakdown of time spent on each session and processing stage is provided in Table 5.2. As indicated in Figure 5.6, the labelization stage accounted for the most substantial portion of the pipeline's processing time.

Sess.	Total	Extr.	Verif.	Align.	Heat.	Label.	Comp.
0	9577.27	726.27	1019.74	62.17	1497.28	5535.46	736.34
1	6636.40	498.91	654.03	28.24	1099.53	3395.17	960.52
2	3350.28	178.58	326.29	4.11	584.05	1728.99	528.26
3	6189.57	375.92	663.27	33.68	1238.00	3532.75	345.95
4	8276.73	633.85	848.37	64.24	1657.07	4546.66	526.53
5	3401.98	198.29	314.29	4.62	564.34	1710.70	609.74
6	5743.96	395.99	607.89	30.29	1147.31	3299.91	262.56
7	2886.07	167.40	283.75	11.10	491.44	1480.14	452.23

Table 5.2: Computational performance of each stage in the data processing pipeline, measured in seconds

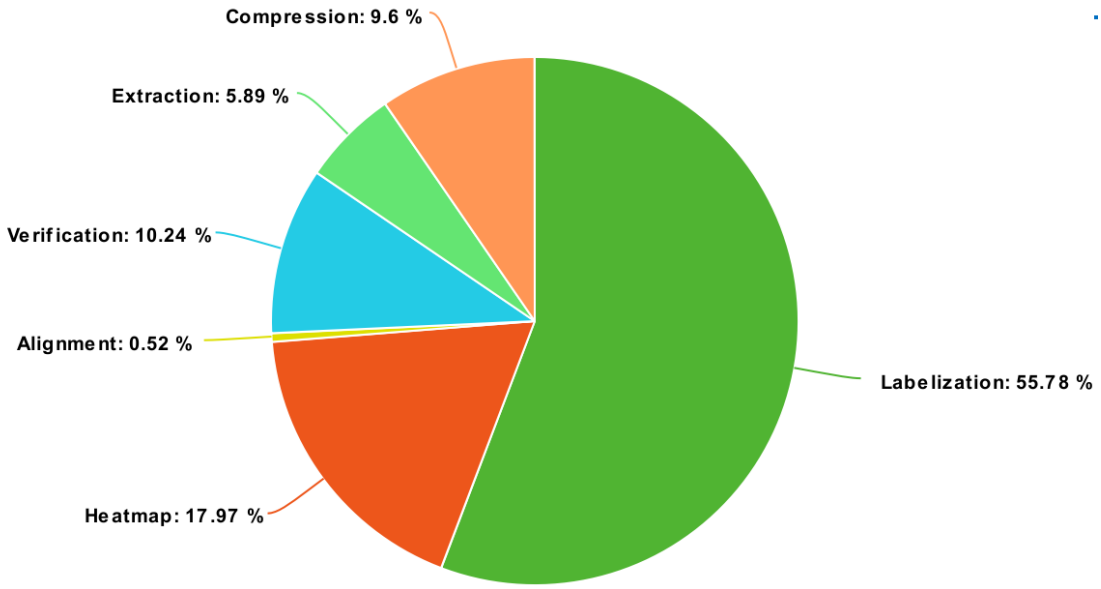


Figure 5.6: Distribution of processing time across stages

Analyzing the table and figure, we can see that labelization accounted for the most significant portion of the total processing time across all sessions. This stage's time-intensity underscores the computational demand of this process and could be a focal point for optimization efforts to improve overall pipeline efficiency. The relatively high time consumption during the verification, heatmap creation, and compression stages also suggest potential areas for process enhancement. In contrast, the extraction and alignment stages required less processing time, indicating they were relatively more efficient or less computationally demanding. These insights can guide future enhancements to the data processing pipeline, potentially focusing on improving labelization and other high-time stages' computational efficiency.

5.6 Discussion of the Results

The experimental results have offered significant insights into the dataset characteristics, their implications on the proposed model's performance, the dataset's verification strategies, and the time and resources consumed in the data processing pipeline.

The investigation of the dataset's characteristics highlighted a few challenges, the most critical being the uneven distribution of samples across vehicles and sessions. The high number of vehicles with fewer than 10 samples, which is the

prerequisite minimum for effectively training the proposed transformer model, suggests potential inadequacies in the learning process and the subsequent model's performance. The limited number of usable vehicles in sessions 2, 3, 5, and 7 further constrain the scope of training and testing data, thereby accentuating the need for more comprehensive data collection.

Despite these challenges, the dataset can serve as a starting point for initial model development and can be refined iteratively as more data gets collected. It emphasizes the importance of data management in shaping model performance, underscoring the need for effective data gathering, filtering, and usage strategies in machine learning.

Verification of the dataset's quality was performed through two methods: a laser rangefinder and overlaying trajectories on Google Maps. Despite each method's inherent limitations, they collectively served to assess the dataset's qualitative validity. This qualitative evaluation revealed potential areas of improvement for future data collection efforts, such as the precision of the rangefinder's placement, improved measurement techniques, and refinements in vehicle identification methods. The process further indicated the importance of integrating multiple data validation strategies to identify and address potential dataset issues.

The analysis of processing time and computational resources revealed the significant computational demand of the labelization stage. It accounted for the largest portion of the total processing time across all sessions. The verification, heatmap creation, and compression stages also consumed substantial time, suggesting potential areas for optimization to improve the overall efficiency of the data processing pipeline.

In contrast, extraction and alignment stages consumed relatively less time, indicating their efficiency or less computational demand. The insights obtained can guide future enhancements to the data processing pipeline, possibly focusing on improving the computational efficiency of labelization and other time-consuming stages.

In conclusion, the experimental results provide critical insights into the dataset and the data processing pipeline, shedding light on potential challenges, areas of optimization, and strategies for future enhancements. They underscore the intricate relationships between data quality, data management, data validation, computational efficiency, and the ultimate performance of machine learning models. Further studies may explore potential solutions to the identified challenges and

incorporate the lessons learned into the development and refinement of machine learning models and data processing pipelines.

Chapter 6

Conclusion

6.1 Identified Limitations and Potential Improvements

The results presented in the experimental section make it clear that our processing pipeline has room for improvement. It is evident that some vehicles are incorrectly tracked across samples, a concern that could be addressed by fine-tuning the identifier's parameters. The generated dataset's size limits its utility for training a model to accurately predict vehicle trajectories in unknown situations. Given the repetitive nature of our data - with most recording sessions sharing similar camera positions and angles - a model trained on this dataset is highly prone to overfitting. To mitigate this risk, we could diversify the recording settings by including additional crossroads.

A significant limitation of the current pipeline is its inability to track multiple vehicles simultaneously. While it is feasible to modify the pipeline to address this shortcoming, the complexity of the task would significantly increase, as detailed in [15]. However, this enhancement is crucial if the goal is to train a robust model that continues to operate efficiently even in heavy traffic conditions.

The pipeline parameters have been fine-tuned based on this specific dataset, primarily using data from the first capture session. If the pipeline were to be used with a substantially different dataset, it would be crucial to reevaluate these metrics to ensure optimal data processing.

The code created during this master's thesis is available under the MIT License [2] at this URL https://github.com/Ery4z/MasterThesis. Detailed information about the code can be found in this repository.

Possible future directions for this work include (but are not limited to):

- Expanding the dataset through additional data recording.
- Implementing a proposed Transformer model to gain insights into data quality and limitations.
- Enhancing the Vehicle Identifier for more accurate tracking of vehicles across data samples.
- Adjusting the data pipeline to accommodate multi-vehicle recognition.
- Improving code to automate the search for optimal processing pipeline parameters.

6.2 Summarizing the Contributions

This master's thesis has proposed a structure for a data processing pipeline that handles bimodal radar-camera data. This pipeline was used to generate a dataset containing bimodal data for various vehicles with their 3D positions and velocities (relative to the acquiring device) in a crossroad context. The dataset includes 992 different vehicles, with 409 of them appearing in more than 10 bimodal pairs. The study also proposes an integration strategy for a transformer model designed for vehicle trajectory prediction, although the model is not yet implemented.

This research lays the groundwork for developing a robust multimodal data processing pipeline to generate a comprehensive, fully labeled bimodal dataset. It offers a code base that anyone can build upon to continue this project, promoting open-source contributions by providing a non-restrictive license. This pipeline will be used to process new bimodal data acquired in April 2023 and likely during the summer months of July-August 2023. The dataset will serve as a valuable resource for Gauthier de Hertaing's PhD research and, more broadly, for the laboratories at École Polytechnique de Louvain. I would like to extend my gratitude to everyone who contributed to this research endeavor.

Bibliography

[1] Explainable artificial intelligence. https://paperswithcode.com/task/explainable-artificial-intelligence. Accessed: 2023-05-21.

[2] The mit license. https://opensource.org/license/mit/.

[3] F. Studer A. Farina. A review of cfar detection techniques in radar systems, 1987.

[4] Alexander Kolesnikov Dirk Weissenborn Xiaohua Zhai Thomas Unterthiner Mostafa Dehghani Matthias Minderer Georg Heigold Sylvain Gelly Jakob Uszkoreit Neil Houlsby Alexey Dosovitskiy, Lucas Beyer. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[5] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need, 2017.

[6] I-Hau Yeh Yueh-Hua Wu Ping-Yang Chen Jun-Wei Hsieh Chien-Yao Wang, Hong-Yuan Mark Liao. Cspnet: A new backbone that can enhance learning capability of cnn, 2019.

[7] Gauthier De Sousa, Kevin ; Rotsart de Hertaing. Conception d'un senseur intégré multimodal pour l'observation des routes. https://dial.uclouvain.be/memoire/ucl/en/object/thesis%3A30593, 2021.

[8] G. M Hatem et a. Comparative study of various cfar algorithms for non-homogenous environments, 2018.

[9] Marco Cristani Fabio Galasso Francesco Giuliari, Irtiza Hasan. Transformer networks for trajectory forecasting, 2020.

[10] Sepp Hochreiter. Long short-term memory, 1997.

[11] Ali Farhadi Joseph Redmon. Yolov3: An incremental improvement, 2018.

[12] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection, 2015.

[13] R. E. KALMAN. A new approach to linear filtering and prediction problems, 1960.

[14] Christophe De Vleeschouwer Laurent Jacques. Lelec2885 - image processing and computer vision, 2023.

[15] François Ledent. Synchronization of multimodal data flows for real-time ai analysis, 2022.

[16] Francesco Giuliari Irtiza Hasan-Marco Cristani Fabio Galasso Luca Franco, Leonardo Placidi. Under the hood of transformer networks for trajectory forecasting, 2023.

[17] Openai. Gpt-3 model card.

[18] David A. Clifton Peng Xu, Xiatian Zhu. Multimodal learning with transformers: A survey, 2022.

[19] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[20] Ross Girshick Kaiming He Bharath Hariharan Serge Belongie Tsung-Yi Lin, Piotr Dollár. Feature pyramid networks for object detection, 2016.

[21] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017.

[22] LIN Haifeng LU Kangjie et al. XU, Renjie. A forest fire detection system based on ensemble learning. *Forests*, 12:217, 2021.

[23] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813, 2017.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**