INF FACULTY OF COMPUTER SCIENCE

Bachelor Thesis

Probabilistic Sensor Fusion for Lidar and Camera Data for Autonomous Vehicles

Malte Rost
Magdeburg, May 2, 2024

Supervisor: Dr.-Ing. Christoph Steup
Professor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Otto-von-Guericke-University Magdeburg
Chair of Computational Intelligence
Faculty of Computer Science

Abstract

The integration of multimodal sensor data for semantic segmentation in autonomous driving systems presents a significant challenge due to the inherent differences and limitations of each sensor modality as well as overconfident and untrustworthy classifier outputs. This thesis explores the use of probabilistic late fusion methods to enhance semantic segmentation demonstrated on Lidar and camera data from from the Waymo dataset. A a novel approach that leverages the advantages of late fusion allowing for the retention of modality-specific information is proposed. The methodology is centered around the development of a Confusion Likelihood Matrix (CLM) that quantifies the uncertainty and reliability of each mortality's predictions. By incorporating this matrix into the fusion process, the system dynamically adjusts the weighting of each mortality's input based on its performance in various scenarios. Experiment show that the effective combination of the strengths of different modalities is possible using advanced fusion methods such as CLM Fusion. The findings of this thesis underscore the potential, as well as the drawbacks, of probabilistic late fusion methods in overcoming the challenges of multimodal sensor fusion.

Contents

List of Figures

List of Tables

1 Motivation and Goals

The increasing popularity of autonomous driving technologies has brought machine perception to the forefront of automotive research and development. Among the various components of machine perception, semantic segmentation plays a pivotal role in providing a detailed and comprehensive understanding of the vehicle's surroundings. This capability is crucial for ensuring the safety and reliability of autonomous vehicles.
However, semantic segmentation presents unique challenges. Some classes are inherently more difficult to detect, or may be frequently confused with one another by the segmentation algorithms. Additionally, modality-specific characteristics, as well as environmental conditions such as lighting, weather, and seasonal changes can significantly affect the performance of the sensors and, by extension, the accuracy of the segmentation process.
To address these challenges, multimodal sensor fusion emerges as a promising solution. By leveraging the complementary strengths of various modalities, such as cameras and LiDAR, multimodal fusion aims to enhance the overall perception system. Late fusion, in particular, offers a straightforward and effective approach to integrate the predictions from existing unimodal classifiers without the need for extensive retraining or complex modifications.
The motivation for this research stems from the need to improve the robustness, trustworthiness, and accuracy of semantic segmentation in autonomous vehicles, especially under varied and adverse environmental conditions. By analyzing and quantifying the strengths and weaknesses of each modality, it becomes possible to devise a fusion strategy that combines their outputs using a weighting system based on calculated performance metrics.
The primary goals of this thesis are:

- The development of a Novel Fusion Technique. A new probabilistic fusion technique will be proposed. The technique should not only aim for high accuracy, but also ensure robustness against adverse conditions, such as poor weather or low lighting. It should be modular, allowing for easy integration and scalability, and should require minimal computational resources to facilitate real-time processing.

- To conduct testing and evaluation of the proposed fusion algorithm using real-world datasets. The analysis will focus on comparing the performance of the proposed method against existing techniques under various scenarios.

By achieving these goals, this thesis aims to contribute to the field of autonomous driving, particularly in enhancing the capability and reliability of semantic segmentation through advanced sensor fusion techniques.

2 Background

TBD: introduction to chapter

2.1 Modalities and their Characteristics

Related literature often uses the term "modality" interchangeably with the term "sensor", and other times it describes a specific type of data format provided by a sensor. For the purposes of this thesis, the term "modality" will denote any set of sensors of the same kind and their combined output. In automated driving, the most commonly used modalities are camera images and LiDAR point clouds. Modern cameras produce high resolution **images** that contain dense RGB-color information in the form of pixels. Because cameras are passive sensors reliant on incoming light, their performance decreases in low light conditions or scenes with extreme brightness, such as direct sunlight e.g. because of a lack of dynamic range that causes high- or low brightness cutoffs, or because of noise. The Field of View (FOV) is also different for Camera and LiDAR. It is often necessary to use multiple cameras to cover more of the surroundings, usually front-, rear-, and side facing cameras are used to create a surround view. Additional cameras are often added for better coverage. 360° LiDARs are often mounted on top of the vehicle and are able to scan in all directions, although directional LiDAR sensors are often used to increase point cloud density and range. Top mounted LiDAR systems are often used in research vehicles used for the creation of datasets such as the Waymo Open Dataset [SKD+20], but are currently not seen in standard factory model passenger cars. The mounting position of the main camera in automobiles is usually below the headliner, replicating the view of the driver. Thus, many areas might be occluded by the hood or pillars of the car, or objects in the scene that would not pose a problem with a higher mounting position [HSL+22]. **Points clouds**, on the other hand, consist of a large amount of scattered points in 3D space. Unlike with pixels, the number of points can vary between frames because not all light-beams are reflected back to the sensor. Each point contains 3D geometry information, as well as features such as reflectance [ZYJ+22]. Compared to typical images, the information in a point cloud is more spread out and thin. This is especially true for sensors that cover large scale scenes, such as those used for automated driving. Not only are the points more spread out because a 360° LiDAR sensor covers a larger area than a camera, but the overall number of points is also less than the number of pixels in a typical image.

2.2 Semantic Segmentation

Semantic segmentation plays a pivotal role in machine perception [SKD+20]. It stands alongside object detection and other techniques as a fundamental method for understanding complex scenes. Unlike object detection, which identifies and locates objects by using bounding boxes [OK17], semantic segmentation goes a step further by classifying each instance such as a specific pixel or point to differentiate between various object instances such as cars, pedestrians, and roads [FHSR+21]. Classification is usually achieved through the use of neural networks, which predict a probability for each label. The final label for each instance can then be determined, e.g. by selecting the label with the highest probability, a process known as Maximum A posteriori Probability (MAP) [CRD+19]. In the context of point clouds, points are often grouped into clusters, with each point in a cluster being assigned the same probability distribution [LMZ+21]. The choice of modality may significantly influence the label-specific segmentation accuracy. Each modality comes with its own set of strengths and limitations. For instance, cameras, which do not perceive depth, might incorrectly identify a person depicted on a billboard as a real person. Similarly, LiDAR sensors, which excel in capturing the shape and form of objects, might struggle to differentiate between objects with similar shapes, such as billboards and traffic signs [LMZ+21].

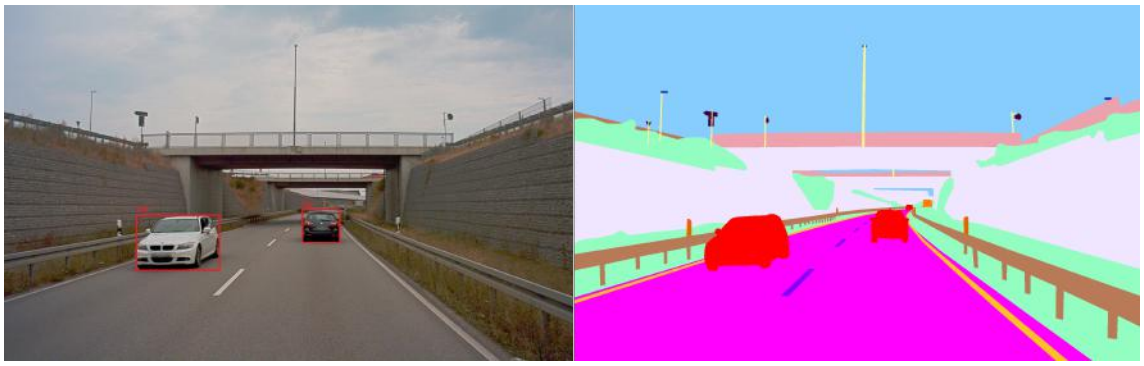

Figure 2.1: Comparison of Bounding Boxes and Semantic Segmentation [GKM+20]

2.3 Fusion

To mitigate specific limitations of a modality, increase the accuracy of predictions, and minimize errors, modern automated driving systems often use sensor fusion to combine multiple inputs and create a single, more comprehensive, and often more accurate output, and thus a better understanding of the vehicles' environment. This chapter outlines the different categories of fusion, their objectives, and the specific focus of this thesis on multimodal fusion.

Fusion categories:

- **Temporal Fusion** is a technique often used in motion prediction because it allows the system to track and anticipate the movement of objects over time, but it can

potentially be used for classification tasks. This technique refines predictions by incorporating data from previous timestamps [Thr02].

- **Spatial Fusion** uses data from different spatial locations and can enhance signal clarity and reduce noise. This approach often involves techniques like neighborhood filters [BCM06] or non-local spatial filtering [BCM05].
- **Multimodal Fusion** involves combining data from multiple sources. This can include the usage of multiple sensors of the same type to broaden the field of view or for redundancy purposes, as well as the crux of this thesis, namely, the combination of different sensor types involved in classification.

Multimodal fusion can be further categorized depending on the stage at which the data is combined, as can be seen in figure 2.2. The two categories modern research mostly focuses on are **early fusion** and **late fusion**. In **early fusion**, features of different modalities are combined before they are further processed. Let Δ^L be the output vector of size L i.e. the predicted probability distribution, and $\mathbb{R}^{N_{pc}}$ and $\mathbb{R}^{N_{pc}}$ be the numerical representation of features for images and point clouds respectively, then the classifier for early fusion can be expressed as [ZCVZ15]:

$$P_{early} : \mathbb{R}^{N_{pc+img}} \to \Delta^L$$

This means that the input for the NN is adjusted to integrate combined features of different modalities instead of just one. One example of early fusion for autonomous driving is the combination of color from an image and 3D coordinates from a point cloud. One way to combine these data types would be to map pixels onto the points, thus adding RGB color information to the points features. The information gain of this representation can then be utilized by a more enhanced neural network to create more comprehensive outputs compared to a classic point cloud. In **late fusion**, the data is first processed unimodal and the output of the NN is then combined. Thus, the classifier for late fusion can be defined as [ZCVZ15]:

$$P_{late} : \Delta^{2L} \to \Delta^L$$

The fusion of the outputs is usually done by heuristic methods. [RPM+22] discusses many different of these heuristic approaches, notably the product-, sum-, median-, and max-fusion. Some of these methods can also be expanded by assigning different weights to the modalities.

$$f_{prod}(X) = \gamma \prod_{i=1}^{N} x_i \qquad f_{wprod}(X) = \gamma \prod_{i=1}^{N} w_i x_i$$

$$f_{sum}(X) = \frac{1}{N} \sum_{i=1}^{N} x_i \qquad f_{wsum}(X) = \frac{1}{N} \sum_{i=1}^{N} w_i x_i$$

$$f_{med} = \Big[med(X_1), \dots med(X_d) \Big] \qquad f_{max} = \Big[max(X_1), \dots max(X_d) \Big]$$

This thesis will introduce a late fusion approach where an image and a point cloud is processed by a NN to create semantic segmentation for those inputs separately and then combines the resulting predictions with a probabilistic model.

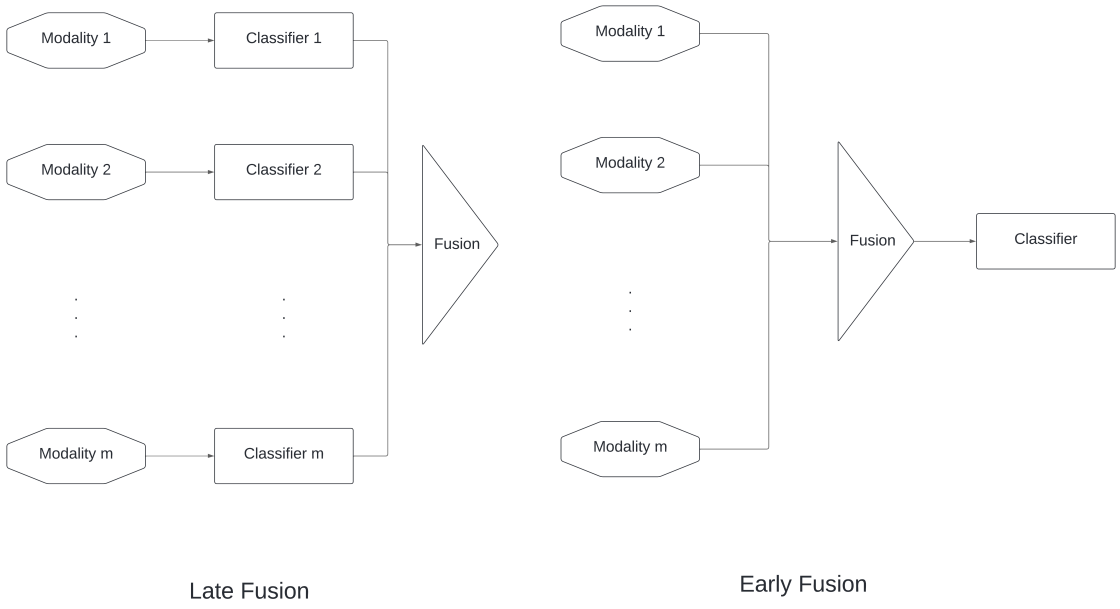

Figure 2.2: Early- and Late Fusion Illustration

2.3.1 Advantages of Late Fusion

A late fusion algorithm is comparatively easy to implement because it can be used on top of existing prediction pipelines. Unlike early fusion, where the input is altered and thus a NN used for prediction has to be constructed for the new input format specifically, late fusion is able to use existing NNs for each modality to create predictions. This is especially important considering that the focus of current research lies more on the development of unimodal classifiers [RPM+22]. It also means that a late fusion algorithm is very modular because it relies not on the data of the sensors, but solely on the output of classifiers. Thus, it is easy to exchange classifiers for the current state of the art to get performance gains without the need for retraining. It also means that adding or removing a modality does not require any alteration regarding other modalities that are already part of the system [RPM+22] and thus the algorithm is not only easy to expand, but also more robust in

situations where a sensor might become defective during operation. The retraining needed in early fusion is also a weak point because of possible performance loss. Training a NN gets more complicated the more input features are involved, mostly because more complex neural networks are more prone to over fitting [ZCVZ15], which means that adding more inputs, and thus more complexity, does not always result in better outputs. Because of the different characteristics of camera and LiDAR, as discussed in chapter 2.1, a large portion of the dataset cannot be regarded during training because there is no projection between the two modalities [ZLJ+21]. Training time can also increase by extreme amounts [KGJM18]. Well implemented late fusion algorithms, on the other hand, can easily add new modalities without decreasing accuracy [PWSR23]. Previous research also shows that late fusion might have an advantage against early fusion, especially when one sensor is dominating in terms of accuracy [PWSR23]. In conclusion, late fusion is more modular, allows a more efficient use of the dataset, and is more promising regarding possible performance gains.

2.4 Coordinate Systems and Projection

Measurements from different sensors cannot be directly fused due to their coordinates being expressed relative to each sensor's position. Thus, all measurements have to be converted to a common coordinate system before they can be effectively used together. The **Object Coordinate System** $P_O = [X_O, Y_O, Z_O]$, also called camera- or LiDAR Coordinate System, is a 3D Cartesian coordinate system with the origin usually at the center of the object (camera or LiDAR). We want to transform these into a common coordinate system, the 3D **World Coordinate System** $P_W = [X_W, Y_W, Z_W]$ that has its origin usually placed at the center of the vehicle. Because the camera is a 2D sensor, its coordinates can be further divided into the two-dimensional **Image Coordinate System** $P_I = [X_I, Y_I]$, which projects the 3D points of the camera coordinate system onto a 2D plane, and the **Pixel Coordinate System** $P_P = [u, v]$, which is a discretized form of the image coordinate system with a grid-structure that is defined by the size of the pixels. Once we defined a translation between the object- and world coordinate systems as shown in [Anw22], we can continue with projecting the 3D LiDAR points onto 2D camera coordinates. Because of FOV limitations, not all LiDAR points fall onto the plane during projection, and it is also very unlikely that all pixels will have an associated point. Thus, some (if not most) of the data will not be fusable. 3D to 2D Projection comes with a loss of depth-information. Multiple points can be projected onto the same pixel, but on the other hand, one point can only be associated with a single pixel.

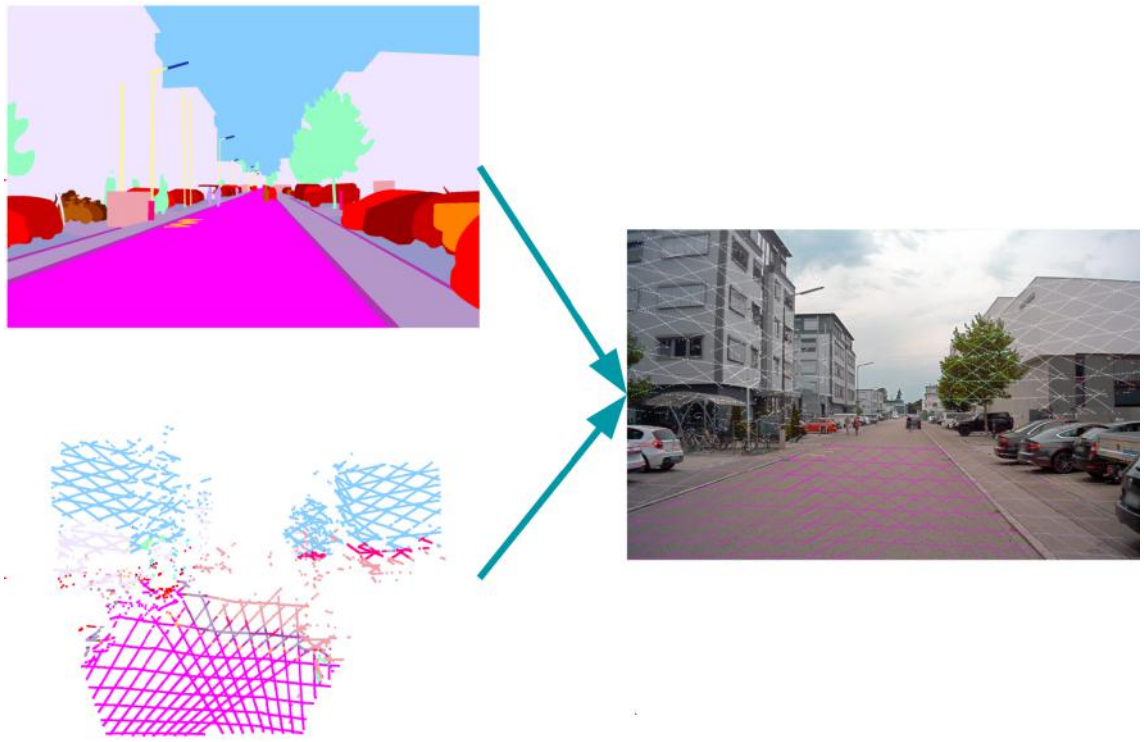

Figure 2.3: Illustration of Projection [GKM+20]

3 Related Work

This chapter reviews some foundational and recent advancements in sensor fusion and semantic segmentation technologies, highlighting their role in enhancing the perception capabilities of automated driving systems and their relation to this thesis.

In the study presented in [FHSR+21], a comprehensive comparison is made between various unimodal classifiers and their fused counterparts for the purpose of semantic segmentation in the context of automated driving. The research extends beyond the conventional use of cameras and LiDAR, but is highlighting these two as the most prevalent and effective sensor types in the domain. The findings suggest that, while LiDAR alone offers superior performance in an unimodal setup, the integration of multiple modalities is crucial for achieving robust and accurate scene interpretation. This is attributed to the ability of multi-modal fusion to leverage the complementary strengths of different sensor types. Another topic highlighted in the paper is the importance of diverse, extensive, and correctly labeled datasets for training classifiers. This will also be important for the late fusion models proposed in this thesis. The paper also discusses different fusion stages and -techniques as discussed in chapter 2.3.

Regarding late fusion techniques, [RPM+22] provides an analysis of several straightforward and commonly used methods and applies them to Driver Behavior Understanding. These methods are categorized into score-level fusion (including product-, sum-, and median-fusion) and rank-level fusion (including Majority Voting, Borda Count, and Reciprocal Rank). The study identifies score-level fusion as the more popular and generally more effective approach, showing the best results. Importantly, the results of the paper suggest that fusion performance increases with the number of modalities/sensors. The results for two modalities even show worse results than the best single modality alone. Because I am working with only two modalities, this knowledge is important when it comes to the analysis.

Further exploration of score-level fusion is undertaken in [WW19], where it is divided into heuristic and network fusion. Heuristic fusion refers to static approaches as discussed in [RPM+22], whereas network fusion involves the use of a neural network trained on classifier outputs. The proposed network model in this paper significantly outperforms heuristic methods in their application. Interestingly, their research also strongly suggests that simple heuristic methods might actually perform worse than top performing unimodal classifiers, although, for other use cases, as shown in [RPM+22] for example, the same issue does not appear as dramatic, only showing slightly worse performance for some classes but otherwise an overall performance gain with fusion. Regardless, the potential loss of

performance remains an issue that needs to be addressed, and it suggests that some kind of enhancement to these methods is necessary to make them competitive. One possible optimization that previously mentioned sources also covered is weighting. Assigning a low weight to an unreliable source could lead to a smaller chance of instance wise performance loss and also overall performance gains. The challenge is to accurately calculate meaningful weights.

In [SAS+22], different kinds of weighting methods are evaluated. Particle Swarm Optimization and Truncated Newton Algorithm are shown to yield the best results out of the tested methods, performing with a mean average precision at the cutoff 10 (MAP@10) of 0.109 compared to an equal weight distribution with a MAP@10 of 0.081. Although the methods tested in this paper are not directly applied in the thesis, they indicate the potential benefits of exploring alternative weighting strategies based on the CLM .

One of the fusion methods used in this thesis is based on the naive Bayes principle. In [SVHVP22], the naive Bayes approach is expanded by introducing environmental variables as a dependency for the likelihood. They use solar altitude as a dependency for an optical camera, and temperature for an infrared camera. A similar approach as described in chapter 4.6 will be used.

4 Methodology

In this chapter, the theoretical framework for the enhancement of classification systems through the use of CLM Fusion will be discussed. The concept of confusion matrices for multi-class scenarios will be enhanced through a new approach that offers a richer, more nuanced view of classifier performance. Then, the usage of this metric to refine classifier outputs and leverage the strengths of different sensor modalities to improve classification accuracy during the inference phase will be explained. Additionally, the chapter discusses alternative fusion methods as a baseline, as well as preprocessing requirements and the importance of scenario distinction in optimizing the performance of fusion methods under varying environmental conditions. This involves a discussion on how external variables such as weather or time of day can impact sensor performance, and how these factors can be integrated into the fusion process to enhance reliability and accuracy.

4.1 Confusion Likelihood Matrix

4.1.1 Confusion Matrices

A confusion matrix creates a representation of the performance of a classification algorithm by comparing the predicted classifications (S) against the actual, true conditions (X). For binary classification tasks, it categorizes predictions into four distinct outcomes:

- True Positives (TP) and True Negatives (TN), which represent accurate predictions where the predicted condition aligns with the actual condition ($S = X$).
- False Positives (FP) and False Negatives (FN), indicating errors where the predicted condition diverges from the actual condition $S \neq X$.

	X = Positive	**X= Negative**
S = Positive	True Positives	False Positives
S = Negative	False Negatives	True Negatives

Table 4.1: Example of a confusion matrix using positive and negative values

Normalization of the matrix values allows for the representation of probabilities, offering insights into the likelihood of each outcome.

Precision, Recall, and Accuracy:

Key metrics derived from the confusion matrix include precision, recall, and accuracy,

	X = Positive	**X= Negative**
S = Positive	$P(X = Positive \wedge S = Positive)$	$P(X = Negative \wedge S = Positive)$
S = Negative	$P(X = Positive \wedge S = Negative)$	$P(X = Negative \wedge S = Negative)$

Table 4.2: Example of a confusion matrix using positive and negative values

each providing different perspectives on the model's performance: Precision measures the accuracy of positive predictions, i.e. the likelihood that an instance detected as a positive is actually positive. Recall measures the model's ability to identify all positive instances, i.e. the likelihood that a positive instance is detected as a positive. Accuracy measures the overall correctness of the model's predictions. These metrics are calculated as follows:

$$precision = \frac{TP}{TP + FP}$$
$$recall = \frac{TP}{P}$$
$$accuracy = \frac{TP + TN}{P + N}$$

Generalization for n-Dimensional Classification:
For applications requiring the differentiation among multiple classes, such as semantic segmentation, confusion matrices can be extended to accommodate a broader set of labels. This expansion transforms the matrix into an array, where each cell represents the number of instances where a specific object was associated with a specific label by the classifier.

	Car	**Street**	**Pedestrian**
Car	Car detected as Car	Street detected as Car	Pedestrian detected as Car
Street	Car detected as Street	Street detected as Street	Pedestrian detected as Street
Pedestrian	Car detected as Pedestrian	Street detected as Pedestrian	Pedestrian detected as Pedestrian

Table 4.3: Example of a confusion matrix using the set of {Car, Street, Pedestrian} as a foundation

Notice that the main diagonal always contains values for the correctly identified instances, i.e. True instances, and outside values represent error (False instances). Thus, higher values on the main diagonal are desirable. We can adjust the equations for precision, recall and accuracy to get a general form for n-dimensional confusion matrices.

$$precision(class) = \frac{|\text{correctly detected class}|}{\text{total number of class detections}} = \frac{P(\text{correctly detected class})}{P(\text{class detection})}$$

$$recall(class) = \frac{|\text{correctly detected class}|}{\text{total number of class instances}} = \frac{P(\text{correctly detected class})}{P(\text{class instance})}$$

$$accuracy = \frac{\text{sum of main diagonal}}{\text{sum of matrix}}$$

Construction of a confusion matrix:

To construct a confusion matrix, each instance in the dataset is evaluated to determine the category it falls into based on its predicted and actual labels. This process involves the aggregation of outcomes across all instances to populate the matrix, providing a comprehensive overview of the model's predictive capabilities as long as the dataset is extensive and has enough variety. Assume we have a set of instances, I where each instance $I_k \in I$ has an associated real label X_j and detected label S_i. We can build a transitional matrix that contains a "1" in cell i, j

	Car	**Street**	**Pedestrian**
Car	0	1	0
Street	0	0	0
Pedestrian	0	0	0

Table 4.4: Example of a transitional matrix where $S_i = Car$ and $X_j = Street$

To create the final confusion matrix, we simply add all transitional matrices.

$$\text{confusion matrix} = \sum_{k=1}^{|I|} \text{transitional matrix}_k \tag{4.1}$$

4.1.2 Introducing the Confusion Likelihood Matrix

A classifier usually calculates a probability distribution instead of delivering a single condition as a prediction. To create a confusion matrix, all instances must first be classified as one possible conditions, e.g. by using MAP or other techniques. This, however, is associated with a loss of information because the uncertainty of the prediction is no longer apparent. For example, assuming we use MAP to obtain a class prediction S_i, the two predictions $\left(0.98, 0.01, 0.01\right)^T$ and $\left(0.34, 0.33, 0.33\right)^T$ both result in a classified output for the first label even though the latter prediction has much higher uncertainty. Afterward, the uncertainty of the prediction does no longer have an impact on the calculation. For this reason, A method is proposed to calculate a confusion matrix that takes the uncertainty of the classifiers into consideration. This altered version of a confusion matrix will be called "Confusion Likelihood Matrix" (CLM), named for its ability to quantify the likelihood of outputs in a classification system. By extending the traditional confusion matrix through the incorporation of the probabilities associated with each classification decision, it is providing a more nuanced understanding of the classifier's performance. This is particularly useful in scenarios where understanding the uncertainty and reliability of predictions is crucial. The output of the classifier for each instance I_k will be represented as a probability vector $P(S|I_k)$ where each element corresponds to the classifier's estimated probability for a given label:

$$P(S|I_k) = \begin{bmatrix} P(S_1|I_k) \\ P(S_2|I_k) \\ \vdots \\ P(S_n|I_k) \end{bmatrix}$$

Similarly, the true condition is represented as a vector, where the actual label has a probability of 1, and all others have a probability of 0.

$$P(X|I_k) = \begin{bmatrix} P(X_1|I_k) \\ P(X_2|I_k) \\ \vdots \\ P(X_n|I_k) \end{bmatrix}$$

with $P(X_j|I_k) = 1$ and $\forall l \in \{1, ...n\} \setminus j : P(X_l|I_k) = 0$

Transition Matrix Using Probabilities: By employing matrix multiplication, we create a transitional matrix that captures the relationship between predicted probabilities and actual conditions:

$$P(S|I_k) \cdot P(X|I_k)^T = \begin{bmatrix} P(S_1 \wedge X_1|I_k) & P(S_1 \wedge X_2|I_k) & \cdots & P(S_1 \wedge X_n|I_k) \\ P(S_2 \wedge X_1|I_k) & P(S_2 \wedge X_2|I_k) & \cdots & P(S_2 \wedge X_n|I_k) \\ \vdots & \vdots & \ddots & \vdots \\ P(S_n \wedge X_1|I_k) & P(S_n \wedge X_2|I_k) & \cdots & P(S_n \wedge X_n|I_k) \end{bmatrix}$$

Since the actual condition is known, the transitional matrix simplifies to a column vector,

where the column corresponding to the true label is equal to the classifier's probability vector.

$$P(S|I_k) \cdot P(X|I_k)^T = \begin{bmatrix} 0 & \cdots & P(S_1 \wedge X_j|I_k) & \cdots & 0 \\ 0 & \cdots & P(S_2 \wedge X_j|I_k) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & P(S_n \wedge X_j|I_k) & \cdots & 0 \end{bmatrix}$$

With $P(X_j \wedge S_i|I_k) = P(S_i|I_k)$.
The final confusion matrix is obtained by summing the transitional matrices for all instances similar to (4.1):

$$\text{confusion matrix} = \begin{bmatrix} \sum_{k=1}^{|I|} P(S_1 \wedge X_1|I_k) & \sum_{k=1}^{|I|} P(S_1 \wedge X_1|I_k) & \cdots & \sum_{k=1}^{|I|} P(S_1 \wedge X_1|I_k) \\ \sum_{k=1}^{|I|} P(S_2 \wedge X_1|I_k) & \sum_{k=1}^{|I|} P(S_2 \wedge X_1|I_k) & \cdots & \sum_{k=1}^{|I|} P(S_2 \wedge X_1|I_k) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{|I|} P(S_n \wedge X_1|I_k) & \sum_{k=1}^{|I|} P(S_n \wedge X_1|I_k) & \cdots & \sum_{k=1}^{|I|} P(S_n \wedge X_1|I_k) \end{bmatrix}$$

This format is similar to a classic confusion matrix, with the difference being that each cell does not represent a number of classified instances, but rather the sum of all classification predictions in their respective column. notice that, if we assume $P(S)$ to be known, i.e. $P(S_i|I_k) = 1$ and $\forall l \in \{1, ...n\} \setminus i : P(S_l|I_k) = 0$, then this format is exactly the same as a classic confusion matrix counting instances as shown in Table 4.4.

4.1.3 Normalization

Using the accumulated sum of classification probabilities is inherently scalable and thus perfect for storage. For example, if we want to expand our dataset split by adding more instances, we can simply add all these new instances analogously to the already existing matrix. But for further calculation, it is necessary to convert the matrix into a probabilistic representation. Assuming that each instance has the same probability $P(I_1) = P(I_2) \cdots = P(I_{|I|}) = \frac{1}{|I|}$, it is possible to express the normalization of a confusion matrix as a derivation of the law of total probability $P(S \wedge X) = \sum_{k=1}^{|I|} P(S \wedge X|I_k) \cdot P(I_k)$.

$$\begin{aligned} CLM_{normed} &= \begin{bmatrix} \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_1 \wedge X_1|I_k) & \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_1 \wedge X_1|I_k) & \cdots & \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_1 \wedge X_1|I_k) \\ \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_2 \wedge X_1|I_k) & \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_2 \wedge X_1|I_k) & \cdots & \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_2 \wedge X_1|I_k) \\ \frac{1}{|I|}\vdots & \vdots & \ddots & \vdots \\ \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_n \wedge X_1|I_k) & \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_n \wedge X_1|I_k) & \cdots & \frac{1}{|I|}\sum_{k=1}^{|I|} P(S_n \wedge X_1|I_k) \end{bmatrix} \\ &= \begin{bmatrix} P(S_1 \wedge X_1) & P(S_1 \wedge X_2) & \cdots & P(S_1 \wedge X_n) \\ P(S_2 \wedge X_1) & P(S_2 \wedge X_2) & \cdots & P(S_2 \wedge X_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(S_n \wedge X_1) & P(S_n \wedge X_2) & \cdots & P(S_n \wedge X_n) \end{bmatrix}_I \end{aligned} \tag{4.2}$$

From the normalized matrix, we can derive the probability of detecting a condition (row sum) and the probability of encountering a condition (column sum).

$$P(S_i) = \sum_X P(S_i \wedge X_j)$$
$$P(X_j) = \sum_S P(S_i \wedge X_j)$$

These values are used to calculate the conditional probabilities of actual conditions given the predicted conditions, and vice versa:

$$P(X_j|S_i) = \frac{P(S_i \wedge X_j)}{P(S_i)}$$
$$P(S_i|X_j) = \frac{P(S_i \wedge X_j)}{P(X_j)}$$

For ease of use, normalization matrices, denoted as NS and NX are introduced, that facilitate the computation of conditional probabilities through element-wise multiplication, or Hadamard Product (denoted with $\odot$) with the confusion matrix:

$$P(X|S) = CLM \odot NS, \quad P(S|X) = CLM \odot NX$$

These matrices contain the reciprocals of the probabilities $P(S)$ and $P(X)$ respectively, allowing for the direct calculation of conditional probabilities.

$$NS = \begin{bmatrix} \frac{1}{P(S_1)} & \frac{1}{P(S_1)} & \cdots & \frac{1}{P(S_1)} \\ \frac{1}{P(S_2)} & \frac{1}{P(S_2)} & \cdots & \frac{1}{P(S_2)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{P(S_n)} & \frac{1}{P(S_n)} & \cdots & \frac{1}{P(S_n)} \end{bmatrix}$$
$$NX = \begin{bmatrix} \frac{1}{P(X_1)} & \frac{1}{P(X_2)} & \cdots & \frac{1}{P(X_n)} \\ \frac{1}{P(X_1)} & \frac{1}{P(X_2)} & \cdots & \frac{1}{P(X_n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{P(X_1)} & \frac{1}{P(X_2)} & \cdots & \frac{1}{P(X_n)} \end{bmatrix}$$

Calculating $P(X|S)$:

$$
\begin{aligned}
P(X|S) &= CLM \odot NS \\
&= \begin{bmatrix}
\frac{P(S_1 \wedge X_1)}{P(S_1)} & \frac{P(S_1 \wedge X_2)}{P(S_1)} & \cdots & \frac{P(S_1 \wedge X_n)}{P(S_1)} \\
\frac{P(S_2 \wedge X_1)}{P(S_2)} & \frac{P(S_2 \wedge X_2)}{P(S_2)} & \cdots & \frac{P(S_2 \wedge X_n)}{P(S_2)} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{P(S_n \wedge X_1)}{P(S_n)} & \frac{P(S_n \wedge X_2)}{P(S_n)} & \cdots & \frac{P(S_n \wedge X_n)}{P(S_n)}
\end{bmatrix} \\
&= \begin{bmatrix}
P(X_1|S_1) & P(X_2|S_1) & \cdots & P(X_n|S_1) \\
P(X_1|S_2) & P(X_2|S_2) & \cdots & P(X_n|S_2) \\
\vdots & \vdots & \ddots & \vdots \\
P(X_1|S_n) & P(X_2|S_n) & \cdots & P(X_n|S_n)
\end{bmatrix}
\end{aligned}
\tag{4.3}
$$

Analogously for $P(S|X)$:

$$
\begin{aligned}
P(X|S) &= CLM \odot NX \\
&= \begin{bmatrix}
P(S_1|X_1) & P(S_1|X_2) & \cdots & P(S_1|X_n) \\
P(S_2|X_1) & P(S_2|X_2) & \cdots & P(S_2|X_n) \\
\vdots & \vdots & \ddots & \vdots \\
P(S_n|X_1) & P(S_n|X_2) & \cdots & P(S_n|X_n)
\end{bmatrix}
\end{aligned}
\tag{4.4}
$$

4.1.4 Calculation Example

Let's define a set of classifier outputs. These outputs can be interpreted as the class-probabilities for a single pixel or point that were calculated by a classifier.

$$
\mathbf{P(S|I_1)} = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}
\mathbf{P(S|I_2)} = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.3 \end{bmatrix}
\mathbf{P(S|I_3)} = \begin{bmatrix} 0.1 \\ 0.6 \\ 0.3 \end{bmatrix}
\mathbf{P(S|I_4)} = \begin{bmatrix} 0.4 \\ 0.4 \\ 0.2 \end{bmatrix}
\mathbf{P(S|I_5)} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.6 \end{bmatrix}
$$

$$
\mathbf{P(S|I_6)} = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix}
\mathbf{P(S|I_7)} = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.2 \end{bmatrix}
\mathbf{P(S|I_8)} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.5 \end{bmatrix}
\mathbf{P(S|I_9)} = \begin{bmatrix} 0.4 \\ 0.5 \\ 0.1 \end{bmatrix}
\mathbf{P(S|I_{10})} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}
$$

Our dataset contains the labeled data that will be used for comparison. We can interpret the real class-label for each pixel as another probability distribution with a 100% probability for the given class.

$$
\mathbf{P(X|I_1)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}^T
\mathbf{P(X|I_2)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T
\mathbf{P(X|I_3)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}^T
\mathbf{P(X|I_4)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}^T
\mathbf{P(X|I_5)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T
$$

$$\mathbf{P(X|I_6)} = \begin{bmatrix}1\\0\\0\end{bmatrix}^T \quad \mathbf{P(X|I_7)} = \begin{bmatrix}0\\1\\0\end{bmatrix}^T \quad \mathbf{P(X|I_8)} = \begin{bmatrix}0\\0\\1\end{bmatrix}^T \quad \mathbf{P(X|I_9)} = \begin{bmatrix}1\\0\\0\end{bmatrix}^T \quad \mathbf{P(X|I_{10})} = \begin{bmatrix}0\\0\\1\end{bmatrix}^T$$

To calculate the CLM , we calculate the projected element wise product for each output and the corresponding real label distribution.

$$P(S|I_1) \odot P(X|I_1) = \begin{bmatrix}0 & 0.2 & 0\\0 & 0.5 & 0\\0 & 0.3 & 0\end{bmatrix}$$

The same calculation is done with all other vectors analogously, and then the sum of all these matrices creates the confusion matrix. To calculate the probabilistic representation of the matrix, we simply need to normalize it with the sum of all values, which is equal to the number of output vectors used (in this example 10).

$$\frac{1}{10}\text{CLM} = \frac{1}{10}\begin{bmatrix}1.3 & 0.8 & 0.7\\1.1 & 2.2 & 0.7\\0.6 & 1.0 & 1.6\end{bmatrix} = \begin{bmatrix}0.13 & 0.08 & 0.07\\0.11 & 0.22 & 0.07\\0.06 & 0.1 & 0.16\end{bmatrix}$$

To calculate $P(X)$ and $P(S)$ we simply take the sum of the columns and rows respectively.

$$P(X_1) = 0.3,\ P(X_2) = 0.4,\ P(X_3) = 0.3$$

$$P(S_1) = 0.28,\ P(S_2) = 0.4,\ P(S_3) = 0.32$$

And then use them to calculate $P(S|X)$ and $P(X|S)$.

$$\begin{aligned}P(X|S) &= CLM \cdot NS\\ &= \begin{bmatrix}\frac{0.13}{0.28} & \frac{0.08}{0.28} & \frac{0.07}{0.28}\\ \frac{0.11}{0.4} & \frac{0.22}{0.4} & \frac{0.07}{0.4}\\ \frac{0.06}{0.32} & \frac{0.1}{0.32} & \frac{0.16}{0.32}\end{bmatrix}\\ &= \begin{bmatrix}0.464 & 0.286 & 0.25\\0.275 & 0.55 & 0.175\\0.187 & 0.313 & 0.5\end{bmatrix}\\ P(S|X) &= CLM \cdot NX\\ &= \begin{bmatrix}\frac{0.13}{0.3} & \frac{0.08}{0.4} & \frac{0.07}{0.3}\\ \frac{0.11}{0.3} & \frac{0.22}{0.4} & \frac{0.07}{0.3}\\ \frac{0.06}{0.3} & \frac{0.1}{0.4} & \frac{0.16}{0.3}\end{bmatrix}\\ &= \begin{bmatrix}0.4333 & 0.2 & 0.2333\\0.3667 & 0.55 & 0.2333\\0.2 & 0.25 & 0.5337\end{bmatrix}\end{aligned}$$

Notice that for $P(X|S)$, the sum of each row equals 1 ($\sum SP(X_j|S) = 1$), and for $P(S|X)$, the sum of each column equals 1 ($\sum XP(S_i|X) = 1$). This is always true for the CLM. The $P(X|S)$ matrix provides insight about the precision, with the class dependent precision scores being contained in the main diagonal. But the matrix is not only indicating overall precision, but also exactly what classes the sensor likely confuses for others. Similarly, the $P(S|X)$ matrix provides insight about recall.

4.2 CLM Fusion

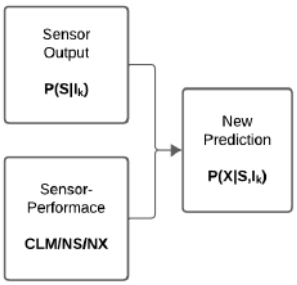

Figure 4.1: Unimodal CLM Fusion

Using the classifier outputs directly as an estimation of the labels is a valid option that is commonly used in most related work, given that classifiers are designed to predict labels accurately. However, acknowledging that classifiers may make observable systematic errors, we can use metrics like the CLM gained from analyzing the classifier to refine the classifier output during the inference phase to make it represent the real-world better based on the derived knowledge. This chapter is dedicated to showing how to combine the Sensor Output $P(S|I_k)$ with the CLM by using the Law of Total Probability to calculate an optimized estimation $P(X|I_k)$.

$$P(X_j|I_k) = \sum_S P(X_j \wedge S_i|I_k) = \sum_S P(X_j|S_i, I_k) \cdot P(S_i|I_k)$$

The CLM is designed to capture average probabilities that accurately represent our classifier's performance. Consequently, we can make an educated approximation for the conditional probability $P(X_j|S_i, I_k) \approx P(X_j|S_i)$.

$$P(X_j|I_k) \approx \sum_S P(X_j|S_i) \cdot P(S_i|I_k) \tag{4.5}$$

The accuracy of this approximation is dependent on the data used to create the CLM, and the refinement of this approximation will be discussed in the following chapters.
It is possible to express equation (4.5) as a matrix multiplication of the CLM, NS, and $P(S|I_k)$ This allows us to calculate all values from the probability distribution for $P(X|I_k)$ simultaneously.

$$
\begin{aligned}
\hat{P}(X|I_k) &\propto P(S|I_k)^T \cdot (CLM \cdot NS) \\
&\propto \begin{bmatrix} P(S_1|I_k) & \cdots & P(S_n|I_k) \end{bmatrix} \cdot \begin{bmatrix} P(X_1|S_1) & P(X_2|S_1) & \cdots & P(X_n|S_1) \\ P(X_1|S_2) & P(X_2|S_2) & \cdots & P(X_n|S_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(X_1|S_n) & P(X_2|S_n) & \cdots & P(X_n|S_n) \end{bmatrix} \\
&\propto \begin{bmatrix} \sum_S P(X_1|S_i) \cdot P(S_i|I_k) \\ \sum_S P(X_2|S_i) \cdot P(S_i|I_k) \\ \vdots \\ \sum_S P(X_n|S_i) \cdot P(S_i|I_k) \end{bmatrix}
\end{aligned} \tag{4.6}
$$

Because these values only represent proportions, a normalization is needed to obtain the final result.

$$
\hat{P}(X|I_k) = \frac{P(S|I_k)^T \cdot (CLM \cdot NS)}{\sum_X P(S|I_k)^T \cdot (CLM \cdot NS)} \tag{4.7}
$$

However, using the CLM to refine an unimodal output will not be the goal of this thesis. The Method will instead be expanded to utilize it for multimodal fusion.

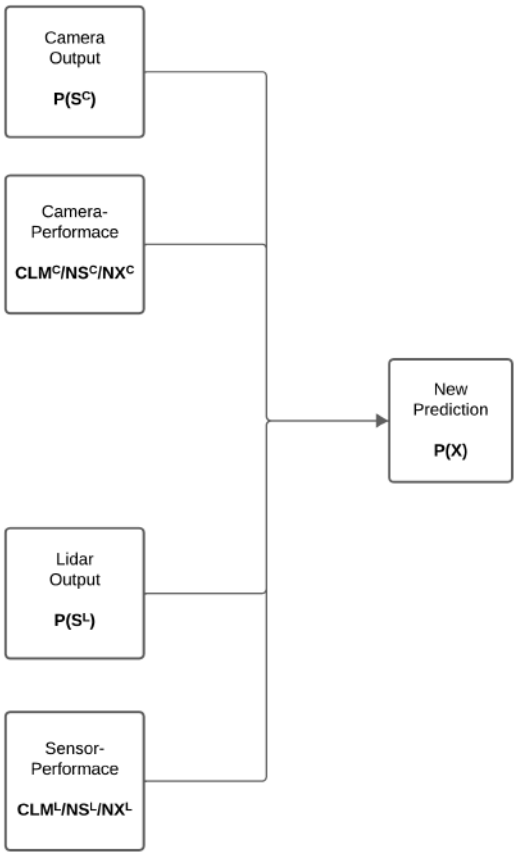

Figure 4.2: Multimodal CLM Fusion

Using the same approach as before, we can expand the formula for multiple modalities. In our case, we have the two modalities: a LiDAR with classifier output S^L and a camera with classifier output S^C. Similar to [SVHVP22], the classifier outputs of all modalities, in our case S^C and S^L, are assumed independent.

$$\begin{aligned} P(X|I_k) &\propto \sum_{S^C}\sum_{S^L} P(X|S^C, S^L, I_k) \cdot P(S^C|S^L, I_k) \cdot P(S^L|I_k) \\ &\propto \sum_{S^C}\sum_{S^L} P(X|S^C, S^L, I_k) \cdot P(S^C|I_k) \cdot P(S^L|I_k) \end{aligned} \tag{4.8}$$

The conditional probability $P(X|S^C, S^L, I_k)$ will be approximated similar as before with $P(X|S^C, S^L)$. For multimodal fusion, the CLM has to be constructed differently than before by using only the part of the dataset where all modalities are fusable, i.e. the data where a point is projectable onto a pixel, so that $X^C = X^L = X$. To calculate $P(X|S^C, S^L)$ with the given assumptions, the formula can be derived as shown in [SVHVP22].

$$P(X|S^C, S^L) = \frac{P(S^C|X) \cdot P(S^L|X) \cdot P(X)}{\sum_X P(S^C|X) \cdot P(S^L|X) \cdot P(X)} \tag{4.9}$$

This conditional probability can be expressed as a 3-dimensional tensor.

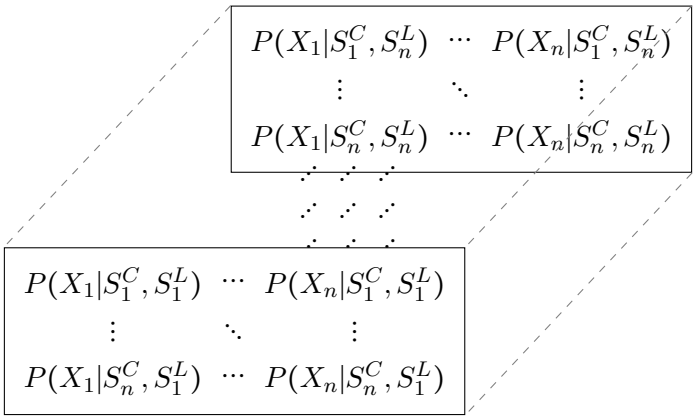

Figure 4.3: 3-Dimensional Tensor

This approach allows for the computation of $P(X)$ using matrix multiplication, similar to the single modality case (4.6) and results in the algorithm's complexity scaling exponentially relative to the quantity of modalities. The implementation will be demonstrated in chapter 5.3

4.3 Properties of the CLM Fusion

Before testing the approach in practice and comparing it to other fusion methods, let us first analyze some factors that influence how a modality affects the outcome of the fusion, and get a better understanding of how the CLM is expected to change this behavior. After a general explanation of CLMs fusion properties, Four different corner cases will be illustrated, detailing the impact of the CLM in these cases, along with examples for illustration purposes.

- How can two sensors complement each other to gain new information?
- The consequences of a sensor producing random or uniformly distributed outputs on the fusion
- The consequences of having a sensor with flawless output accuracy.

4.3.1 Combining Strengths and Mitigating Overconfidence

As discussed in Chapter 2.2, the unique attributes of each modality can lead to distinct advantages and disadvantages in object detection.
Late fusion techniques can harmonize the strengths of various sensors, thereby mitigating their individual weaknesses. Consider, for example, a scenario involving two sensors (a Camera and a LiDAR) tasked with identifying three distinct objects: bicycles, pedestrians, and motorcycles. Suppose the camera frequently confuses bicycles and pedestrians with each other, but has a better track record with motorcycles. Conversely, the LiDAR may confuse bicycles and motorcycles but demonstrates superior accuracy in detecting pedestrians. Let's call the set of classes that are frequently confused by a modality Confusion Classes.

$$C^C = \{bicycle, pedestrian\}$$
$$C^L = \{bicycle, motorcycle\}$$

Note that the examples provided here are hypothetical and not derived from empirical data.
If we assume that the classifier outputs are representative of their confidence, we can expect them to show lower, almost evenly distributed probabilities for their Confusion Classes.

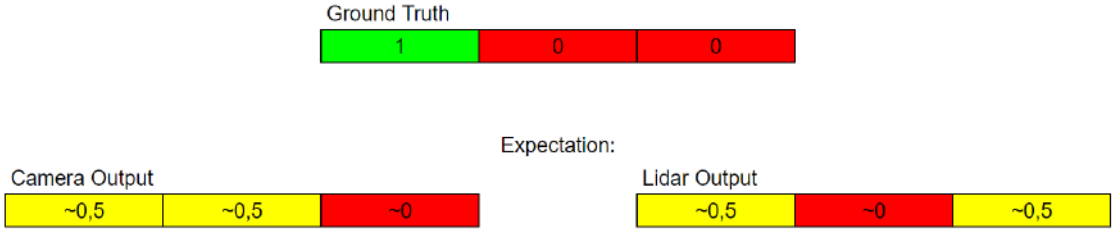

Figure 4.4: Illustration of an Expected Classifier Output

Unfortunately, in real-world scenarios, we have to consider the trustworthiness of the classifier. Classifiers tend to be overconfident in their predictions, meaning that classes with high probabilities are further boosted in their score. This often results in very confident predictions with low entropy outputs. The problem with this can be seen when the overconfident estimation of one modality is wrong. If both modalities disagree with a high confidence, it is difficult to decide.
There is a high chance that at least one of the classifiers delivers unreliable data, which will have a negative impact on the fusion result if we aren't altering the classifier output in any way to accommodate for the known problems of each modality. Here is an example to illustrate this. Suppose that we have a motorcycle and the camera, being better at detecting this class, correctly provides a very high probability for it. Meanwhile, the LiDAR misclassifies it as a bicycle this time. Using unweighted product fusion will result in a high probability for both the bicycle and motorcycle class.

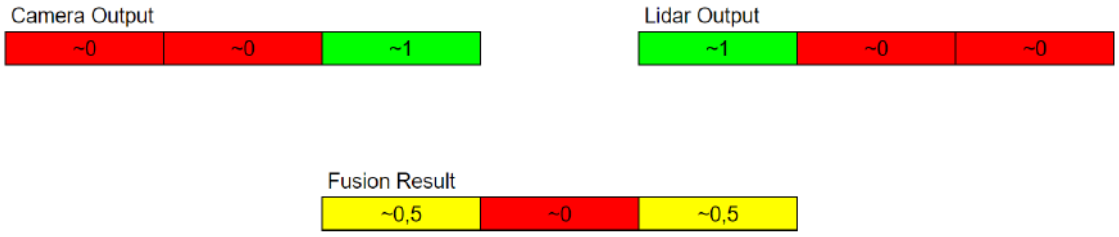

Figure 4.5: Illustration of a Naive Fusion Method

To alleviate this problem to some extent, weighting methods can be used to differentiate between more or less trustworthy outputs, taking more influence from trustworthy sources. The CLM Fusion at its core follows the same idea, using the CLM to alter the classifier outputs according to their trustworthiness. We can see this by taking a look at the unimodal CLM Fusion. The CLMs for each sensor, as illustrated in Figure 4.6, would reflect the specified tendencies of the camera and LiDAR.

Figure 4.6: CLM Example

When a modality indicates a high probability of the presence of a label it can detect reliably, the CLM fused output will mirror this confidence and the entropy will only increase slightly. However, if it suggests a high probability for a member of a Confusion Class, the fused output would yield a moderately high probability for both, reflecting the modalities' uncertainty.

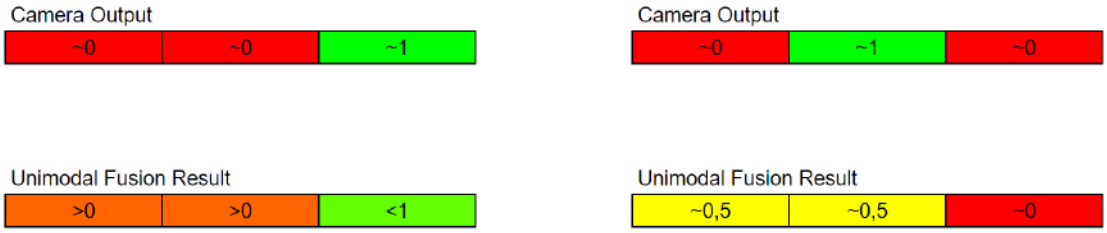

Figure 4.7: Illustration of Unimodal CLM Fusion

Now let's take a look at the same situation as before. Given our understanding of each sensor's reliability, we would intuitively favor the camera's assessment, leading us to a fused prediction with a relatively high probability for the motorcycle label close to the cameras output, a relatively low probability for the bicycle label close to the cameras output, and an even lower probability for the pedestrian label as demonstrated in figure 4.8.

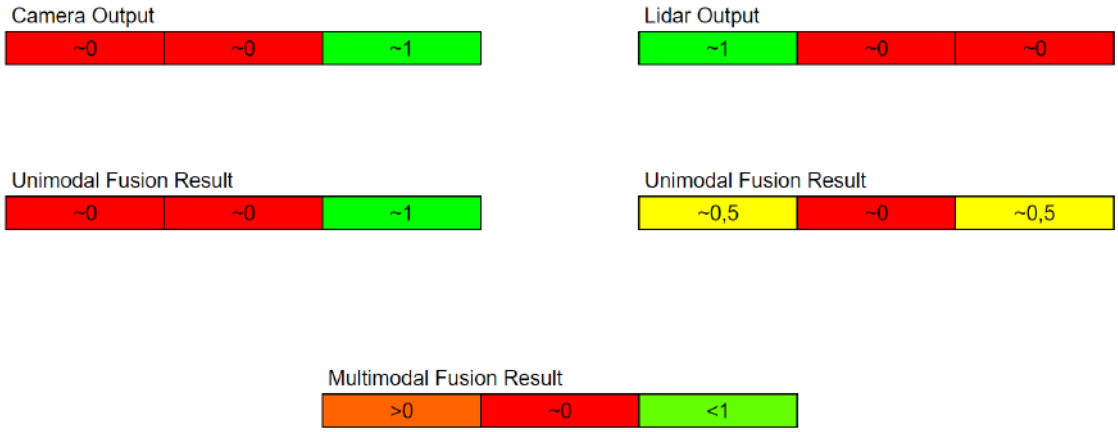

Figure 4.8: Illustration of Multimodal CLM Fusion 1. Case

The proposed fusion method aims to emulate this intuitive logic in practice. Moreover, sensor fusion can provide additional insights when both sensors yield wrong estimations. For instance, if the camera incorrectly detects a pedestrian and the LiDAR incorrectly detects a motorcycle, the fusion algorithm would adjust the probabilities, resulting in a heightened likelihood for a bicycle, although the confidence will be lower than in the previous example.

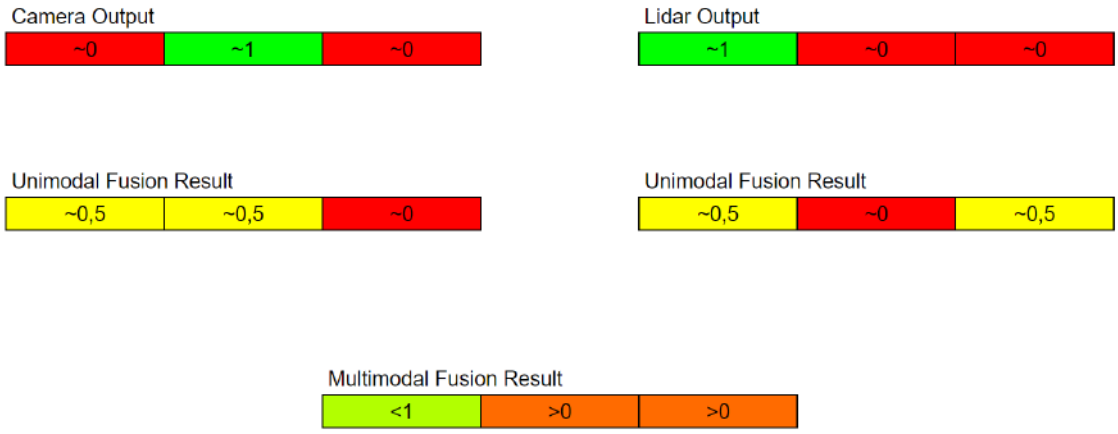

Figure 4.9: Illustration of Multimodal CLM Fusion 2. Case

4.3.2 Entropy

The entropy of a probability distribution serves as a measure of uncertainty or randomness. When examining the influence of entropy on the output of the fusion result, we can see a clear relationship. High entropy input distributions, characterized by a balanced and diverse distribution of values, tend to contribute less to the final output. Conversely, low entropy input distributions, where one value dominates while others are rare, exert a more pronounced influence on the output. This means that less trustworthy sensors with low accuracy generally have a smaller influence on the fusion result than high accuracy sensors. Let's take a look at two corner cases to illustrate this correlation. If a sensor has an **evenly distributed output** (i.e. and entropy of 1), the CLM will have evenly distributed columns and the sum of each column will be equal to $P(X)$. The same is true if the sensor outputs **random values** because in this case, the average of all instances will be evenly distributed.

$$CLM = \begin{bmatrix} \frac{P(X_1)}{n} & \cdots & \frac{P(X_n)}{n} \\ \vdots & \ddots & \vdots \\ \frac{P(X_1)}{n} & \cdots & \frac{P(X_n)}{n} \end{bmatrix}$$

$P(S)$, on the other hand, will have the same value for each label. $P(S_1) = P(S_2) \cdots = P(S_n) = \frac{1}{n}$. During the inference phase, the distribution of the output has no influence on the result if each column of the CLM consists of the same values.

$$\begin{aligned} P(S|I_k)^T \cdot (CLM \odot NS) &= \begin{bmatrix} P(S_1|I_k) & \cdots & P(S_n|I_k) \end{bmatrix} \cdot \begin{bmatrix} \frac{\hat{P}(X_1)}{n} & \cdots & \frac{\hat{P}(X_n)}{n} \\ \vdots & \ddots & \vdots \\ \frac{\hat{P}(X_1)}{n} & \cdots & \frac{\hat{P}(X_n)}{n} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\hat{P}(X_1)}{n} \cdot \sum SP(S_i|I_k) & \cdots & \frac{\hat{P}(X_n)}{n} \cdot \sum SP(S_i|I_k) \end{bmatrix} \\ &\propto \begin{bmatrix} \hat{P}(X_1) & \cdots & \hat{P}(X_n) \end{bmatrix} \end{aligned}$$

As we can see, in this case, the result is always equal to the average label distribution of the modality.
Looking at the multimodal fusion, we can see that this correlation holds there too. Furthermore, if only one modality is random, then it will have no effect on the calculation. Let's look at an example where S^L is a classifier with random output:

$$\begin{aligned}
P(X) &= \sum_{S^C}\sum_{S^L} \frac{P(S^C|X)\cdot P(S^L|X)\cdot P(X)}{\sum_X P(S^C|X)\cdot P(S^L|X)\cdot P(X)} \cdot P(S^C)\cdot P(S^L) \\
&= \sum_{S^C}\sum_{S^L} \frac{P(S^C|X)\cdot P(X)}{\sum_X P(S^C|X)\cdot P(X)} \cdot P(S^C)\cdot P(S^L) \\
&= \sum_{S^C} P(S^C)\cdot \frac{P(S^C|X)\cdot P(X)}{\sum_X P(S^C|X)\cdot P(X)} \cdot \sum_{S^L} P(S^L) \\
&= \sum_{S^C} P(S^C)\cdot P(X|S^C)
\end{aligned}$$

An evenly distributed output acts as a neutral element during fusion. Thus, we have effectively eliminated the randomness of the output to create a result that represents no knowledge and will have no influence on the fusion. whether this is more useful than a result that represents the average probabilities for X is up to debate. Knowing that two different modalities most likely have different average probabilities, we can assume that they might provide a useful information gain during fusion. At the same time, treating a random sensor as a viable information source seems contradictory.
On the other hand, if we assume a **perfect sensor** where $\forall k : P(S|I_k) = P(X|I_k)$, the CLM will only have values grater zero on the main diagonal and $P(S) = P(X)$. In fact, the CLM will look as follows:

$$CLM = \begin{bmatrix} P(X_1) & 0 & \cdots & 0 \\ 0 & P(X_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P(X_n) \end{bmatrix}$$

Thus, $CLM \cdot NX = CLM \cdot NS = I$ and $CLM \cdot NX \cdot NS$ results in $\frac{1}{P(X)}$ on the main diagonal. during the inference phase because there are no values greater than zero outside the main diagonal and the sensor output is a unit vector, the fusion result will also be a unit vector representing the correct label. Using the fusion formula will not change the result because all false positives from other sensors will be multiplied with zero and thus ignored.

4.4 Score Level Fusion

To compare the performance of the CLM Fusion, we can take a look at some of the approaches shown in [RPM+22]. Choosing the approach that fits our use case the best is important to achieve the best possible performance. in this thesis, the product and sum fusion methods will be examined, as they have been shown to yield the best results in general.

$$\hat{P}(S) \propto \hat{P}_C(S) \cdot \hat{P}_L(S)$$

or

$$\hat{P}(S) \propto \hat{P}_C(S) + \hat{P}_L(S)$$

Both methods come with advantages and disadvantages, with the Product Fusion showing more similarity to the Probabilistic approach. Product fusion emphasizes modality consensus. It results in increased scores for a label if both sensors concurrently show a high probability, as multiplying two high values increases the result even further in relation to other scores compared to summation. This can be beneficial when both sensors provide **trustworthy information**. It also decreases the score in case of discrepancies. If one sensor has a very low probability for a certain label, the resulting fusion output for that label will also be low, even if the other modality provides higher outputs. This helps in reducing the impact of outliers or unreliable sensor readings if both sensors provide trustworthy results. But to effectively use Product Fusion, the trustworthiness has to be maintained, otherwise, the fusion will Over-penalize disagreements in case of over-confident scores.
Sum Fusion, on the other hand, preserves the diversity of information from both sensors. While this leads to an inaccurate representation of each modalities' confidence, because the entropy does no longer have the effect described in chapter 4.3.2, it can be useful if the outputs aren't trustworthy. Sum Fusion doesn't excessively penalize one sensor's high confidence in a label, even if the other sensor falsely disagrees. This makes it robust to untrustworthy scores. If one modality provides unreliable readings for certain labels, Sum Fusion can compensate by relying more heavily on the other modality's information with less influence from other modalities compared to product fusion.
Different weighting methods for the Product and Sum Fusion will be proposed to alleviate some associated problems. The Fusion formula for Sum Fusion can be adjusted by adding a modality-specific weight w^M. We can express the weighted Sum fusion as follows:

$$P(S) \propto w^C \cdot P(S^C) + w^L \cdot P(S^L) \tag{4.10}$$

The weight can be expressed in many different forms. Traditionally, each modality has a specific weight based on their performance, such as accuracy. The unweighted Fusion can

be expressed by using the same weights for each modality:

$$w^C = w^L = 1$$

It has the most potential for errors because each modality has the same influence regardless of its performance and thus the most potential for bad performing classifiers to negatively influence the better performing ones. This was also supported by multiple different sources, such as [RPM+22].
Using a scalar weight such as the accuracy:

$$w^C = \frac{acc^C}{acc^C + acc^L}$$

lowers the potential risk for negative influences, especially in cases where one modality vastly outperforms another by relying more on the higher performing modality. At the same time, this approach still has room for improvement, having a big information loss because the confusion matrix is compressed into a single scalar weight.
Another approach that utilizes the information of the CLM better would be the calculation of individual weights for each label. F1 Scores of the individual label as an indication of how good the sensor is in correctly detecting this specific label can be used to utilize class specific performance better:

$$w_i^C = \frac{F1_i^C}{F1_i^C + F1_i^L}$$

(analogously for w^L)
Weighted Product fusion requires a different approach because the influence of a classifier output on the fusion result is dependent on the entropy of the output. We can alter the entropy by adding evenly distributed noise proportional to the modality weight.

$$P(S^C) \to w^C \cdot P(S^C) + \left(1 - w^C\right) \cdot \mathbb{1}$$
$$P(S^L) \to w^L \cdot P(S^L) + \left(1 - w^L\right) \cdot \mathbb{1}$$

This method maintains the ranking order of a modalities output while increasing entropy, i.e. the order of the highest probability class to lowest does not change, but the differences in probability become smaller proportional to $1 - w$. As presented here, this method is only applicable to scalar weights and will be tested using accuracy weighting.

4.5 Preprocessing

To perform the fusion step correctly, our data has to satisfy several conditions. Firstly, each modality has to deliver data at the same timeframe. This might be achieved by aligning the frame rate, or, if this isn't possible, by using temporal interpolation [ZLW+22]. We need to use projection to create a tuple of matching data. In our case, each point is projected into image space to create a tuple of pixel- and point estimation $(P(S^C), P(S^L))$.
By fusing both estimations, we create a new, more comprehensive estimation:

$$\vec{P}(S^L) \cdot \vec{P}(S^C) \rightarrow \vec{P}(S)$$

Points and pixels that are not projectable cannot be used during the multimodal fusion shown in this thesis, but they might be used for further calculations, as will be discussed in chapter 6. The projection can potentially cause problems when there are more than two matching instances. Because a pixel can be matched to multiple points but not the other way around, three different strategies to deal with this scenario are proposed.
(1) weight all estimations equally and fuse them into a new prediction:

$$\vec{P}(S^{L1}) \cdot \vec{P}(S^{L2}) \cdot \vec{P}(S^C) \rightarrow \vec{P}(S)$$

(2) fuse the estimations from the point cloud first, e.g. by using one of the simple fusion methods shown in [RPM+22], then fuse the pixel and points as before. Note that the pixel estimation will have a higher weight than the individual points:

$$\vec{P}(S^{L1}) \cdot \vec{P}(S^{L2}) \rightarrow \vec{P}(S^L)$$

$$\vec{P}(S^L) \cdot \vec{P}(S^C) \rightarrow \vec{P}(S)$$

(3) fuse both points separately, resulting in multiple estimations:

$$\vec{P}(S^{L1}) \cdot \vec{P}(S^C) \rightarrow \vec{P}(S^1)$$
$$\vec{P}(S^{L2}) \cdot \vec{P}(S^C) \rightarrow \vec{P}(S^2)$$

Which strategy works the best depends on the use case and whether further calculations will be done in pixel- or 3D space. (1) and (2) are suitable options for pixel space because there is one resulting estimation for each fusable pixel, thus the total number of pixel estimations is preserved. (2) assigns equal weights to each modality, and thus might be a better option than (1). (3) is a suitable option for 3D space because the total number of point estimations are preserved.

4.6 Extension with Scenarios

- each scenario must be independent for the formula to work
- scenarios are further discussed in 5.2

For the best possible performance, it is important that the confusion matrix represents the actual performance of the sensor as accurately as possible. The problem is that sensor performance is influenced by external variables such as weather or time of day. Thus, a confusion matrix formed over a diverse data set would only represent average performance, and furthermore, the distribution of "good" and "bad" data would have a large influence on the final result. To avoid these problems and at the same time increase the meaningfulness of the confusion matrix, the data set can be split into different parts, with each part depicting a different scenario. Each part of the data set can then be used to calculate its own confusion matrix that represents the performance of the sensor in the respective scenario. During the inference phase, the existing scenario can then be determined, and the appropriate confusion matrix can be used for the calculation. We can alter our fusion formula to incorporate scenarios by adding an environmental dependency G for each modality, similar to the approach shown in [SVHVP22].

$$P(X|S^C, S^L, G) = \frac{P(S^C|X,G) \cdot P(S^L|X,G) \cdot P(X|G)}{\sum_X P(S^C|X,G) \cdot P(S^L|X,G) \cdot P(X|G)}$$

Firstly, it is important to find a definition for a scenario. Typical definitions for scenarios in automated driving involve the action that the autonomous driving system is performing, e.g. lane changes, emergency braking etc. [ZdR17] but for our use case we need to find scenarios that are the most distinct regarding sensor performance. For this purpose, different environmental variables to define a scenario will be used. Each environmental variable is composed of different conditions, e.g. $daytime \in \{day, twilight, night\}$. Every possible combination of environmental conditions is a distinct scenario (see fig. 4.10)

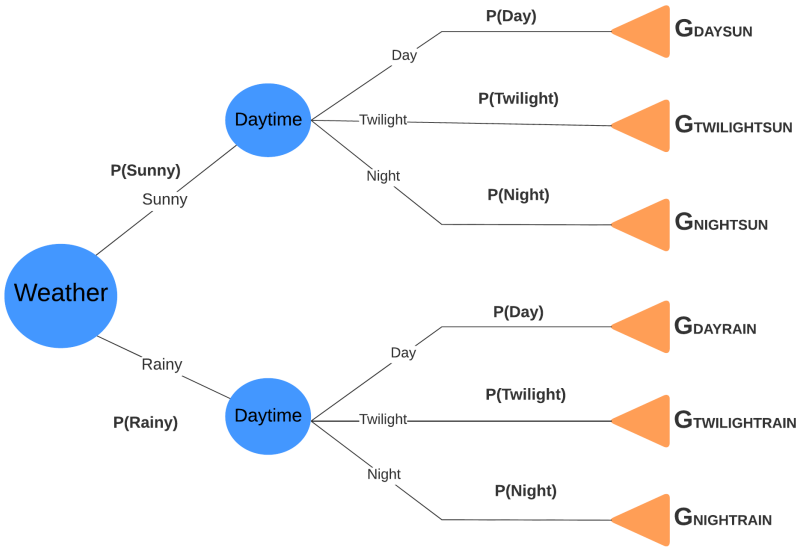

Figure 4.10: Scenario Tree for Environmental Conditions Weather and Daytime

The Scenario distinction can also be applied to the fusion methods presented in chapter 4.4 by using the scenario-specific accuracy or F1 scores.
In case a single scenario cannot be precisely recognized for a given frame, multiple CLMs for different scenarios could be combined by adding them weighted on the scenarios' probability.

5 Experiments

In this chapter, the different fusion methods presented in chapter 4 are tested using a real-world dataset, specifically the accuracy-, F1-, and unweighted Sum Fusion, the accuracy and unweighted Product fusion, as well as the CLM Fusion methods. Other weighting or fusion methods are not considered in this thesis. The usage of a database constructed for the testing of classifiers and their fusion will be explained, as well details about the implementation in code.

5.1 Database

The AulaKI database will be used as a basis for the analysis conducted in this chapter. In the context of this study, all data stored within the AulaKI Database is presumed trustworthy, with any potential discrepancies falling outside the scope of this analysis. The database is organized to encompass images, point clouds, weather conditions, and timestamps for each frame, as well as the ground truth labels, and pre-calculated classifier outputs for the image and point cloud. The image classifier used is based on [HSC+19] but using ResNet [HZRS15] instead of Mobilenet. The LiDAR classifier is based on [LCL+23]. Image classifier outputs are stored as a 3-dimensional tensor containing a probability distribution for each pixel, and point cloud classifier outputs are stored as a sparse 3D tensor containing the probability distribution for each projected point projected onto the image plain. The ground truth is stored in image format, containing the index of the real label for each pixel. Due to timestamp errors, a large amount of data is not usable for fusion. The dataset is split into two distinct parts.

- 80% are the training split, which contains data that was used to train the classifiers.
- 20% the validation split, which was used to validate the classifiers. (57% of which are not usable due to timestamp deficits)

To avoid the risk of overfitting, the training split is excluded from the CLM calculations and testing. Instead, the validation split is used to construct the CLM and to simulate the inference phase fusion. The classifiers are trained to detect 27 different labeled classes grouped into 11 categories (see Table 5.1). It is noteworthy that certain classes are absent in real-world scenarios and will be disregarded during the analysis to ensure the relevance and accuracy of the experimental outcomes. Table 5.1 also contains the amount of labeled instances that are used for testing.

Label ID	Name	Category	Number of Labeled Instances
20	terrain	nature	0
24	vegetation	nature	1123573
10	lane marking	lane marking	285832
0	animal	other	37
2	bird	other	0
6	dynamic	other	32713
7	ego vehicle	other	605
9	ground	other	419539
12	other vehicle	other	11142
19	static	other	508570
13	person	person	218868
15	rider	person	6281
14	pole	pole	123637
16	road	road and sidewalk	5397364
17	sidewalk	road and sidewalk	662168
18	sky	sky	16155
3	building	structure	1597152
8	fence	structure	0
26	wall	structure	0
21	traffic light	traffic signal	12366
22	traffic sign	traffic signal	44646
1	bicycle	vehicle	3669
4	bus	vehicle	3890
5	car	vehicle	1468829
11	motorcycle	vehicle	1972
23	truck	vehicle	87487
25	void	void	60789

Table 5.1: Labels ordered by category

5.2 Scenarios

The selection of scenarios for the analysis in this thesis has been chosen dependent on the dataset's available information, specifically time and weather data. To categorize the data, it has been divided into three time-based categories: DAY, NIGHT, and TWILIGHT, and two weather-based categories: SUNNY and RAIN. This categorization yields six distinct scenarios:

$$\{DAYSUN, DAYRAIN, NIGHTSUN, NIGHTRAIN,$$
$$TWILIGHTSUN, TWILIGHTRAIN\}$$

However, the dataset exhibits a significant disparity in the volume of data available for each scenario. The following table summarizes the number of pixel/point pairs (i.e. the number of pixels that are projectable) for each scenario, illustrating the distribution of data:

Scenario	Relative Frequency
DAYSUN	85.8%
DAYRAIN	0%
NIGHTSUN	7.9%
NIGHTRAIN	0%
TWILIGHTSUN	6.3%
TWILIGHTRAIN	0%

Table 5.2: Data Availability by Scenario

The table reveals that scenarios involving RAIN, i.e. DAYRAIN, TWILIGHTRAIN, and NIGHTRAIN lack any data, rendering them unsuitable for comprehensive analysis. Conversely, the SUNNY scenarios DAYSUN, TWILIGHTSUN, and NIGHTSUN present a spectrum of data availability, with DAYSUN being the most populated scenario. The uneven distribution of data across scenarios necessitates careful consideration when evaluating the performance of fusion methods. The difference in data volume may influence the generalizability of the findings. Consequently, the analysis will mainly focus on the DAYSUN scenario.

5.3 Code

The implementation of the CLM calculation and the experimental inference phase fusion simulation is carried out using Python 3.10.12 and PyTorch 2.1.2 [PGM+19] with CUDA 12.1. The process begins by iterating over all entries in the database, filtered according to the specified scenario. From these entries, the real labels X are extracted from the ground truth, and the predictions S from the classifier outputs. Initially, each LiDAR point is associated to a camera pixel. If multiple LiDAR points are associated with the same pixel, they are pre-fused according to chapter 4.5 using Product Fusion. If no LiDAR point exists for a pixel, the pixel is discarded. The output is a $n \times 27$ tensor, with n being the number of projectable pixels. Each pixel at index i of the camera tensor corresponds to a point at index i of the LiDAR tensor. A key function utilized in this process is torch.einsum, which uses a notation based on the Einstein summation convention.

```
torch.einsum("s,x->sx", S, X)
```

The function is particularly useful for calculating transitional matrices, which are essential for creating the CLM. The syntax “s,x->sx” in the torch.einsum function specifies the dimensions of the input tensors, where “s” represents the dimension of the predictions and “x” the dimension of the ground truth. S and X are 27×1 tensors for the predictions and ground truth, respectively. The Einstein Summation Convention states that explicitly stated dimensions are kept in the output and not stated dimensions are summed over. Consequently, “s,x->sx” for the tensors S and X states that the product of each combination of values of the two input tensors are put into a 27×27 matrix. This is equivalent to $S \cdot X^T$.

To compute a CLM more efficiently, the transitional matrices can be computed for an entire frame at once, the following code demonstrated this:

```
clm = torch.zeros((27,27))
for X_frame, S_frame in iterator:
    transitional_matrix += torch.einsum("nc,nx->cx", (S_frame, X_frame))

# normalize to get probabilities
clm = torch.div(clm, clm.sum())
```

During the inference phase, the CLM fusion also uses the torch.einsum function. This phase involves creating a three-dimensional tensor (`likelihood`) as shown in figure 4.3. The probabilities $P(S^C|X)$ (`clm_c * nx_c`), $P(S^L|X)$ (`clm_c * nx_c`) and $P(X)$ (`X`) are derived from the CLMs, while $P(S^C|I_k)$ (`S_c`) and $P(S^L|I_k)$ (`S_l`) are obtained from the classifier outputs.

```
likelihood = torch.einsum('x,cx,lx->xcl',X,clm_c * nx_c,clm_l * nx_l)
likelihood = likelihood / likelyhood.sum(0)
```

The fusion is then executed as follows:

```
CLM_fused = torch.einsum('ic, il, xcl -> ix', S_c, S_l, likelihood)
```

5.4 Results

The performance evaluation process in this chapter is done using the validation split of the dataset by executing the inference phase of the fusion. To analyze and compare the performance of both unimodal classifiers and the different fusion methods, their respective F1 scores as well as the overall accuracy and macro-averaged F1 scores, grouped by scenario will be compared. This way, the comparison of modality-specific strengths and weaknesses will be made obvious, as well as overall performance. To explain the impact of certain unimodal behaviors on the fusion result, the classes will be grouped into well performing and low performing classes for each modality, based on their F1 scores. A modality is deemed well-performing for a class if the F1 score for that class is at least 0.5, indicating a relatively high level of accuracy in classification. Conversely, a modality is considered low-performing for a class if the F1 score is below 0.5, suggesting a lower accuracy. It is important to note that the 0.5 threshold is used for a basic categorization, and there may still be significant performance differences within one category. The class performance will then be compared before and after fusion to analyze the impact when:

- both modalities are well performing
- only one modality is well performing
- both modalities are low performing

- the F1 score of one modality is close to zero

To compare the fusion methods, the focus will be on the DAYSUN Scenario because it contains the largest amount of data and shows good overall performance for both camera and LiDAR. For the F1 score analysis, labels “animal” and “ego vehicle” will be omitted due to a severe lack of data. This also affects the calculation of the macro weighted F1 score.

5.4.1 Differences Between Modalities and Scenarios

First, the performance of the unfused modalities alone will be analyzed. The LiDAR sensor demonstrates generally lower accuracy and F1 scores than the camera, outperforming it in only slightly in the classes “bicycle” and “rider”. This observation is contrary to the findings of [FHSR+21] and may partially be due to the inability of the LiDAR to detect certain classes. The LiDAR particularly struggles with classes “dynamic,” “ground,” “other vehicle,” “sky,” and “static,” showing F1 Scores close to zero. other classes such as “lane marking” also have much lower F1 Scores than the camera, likely due to an indistinguishable shape, while classes such “pole” and “traffic light” might have bad performance due to the small size making it hard for the LiDAR to pick up distinct and consistent geometric features.

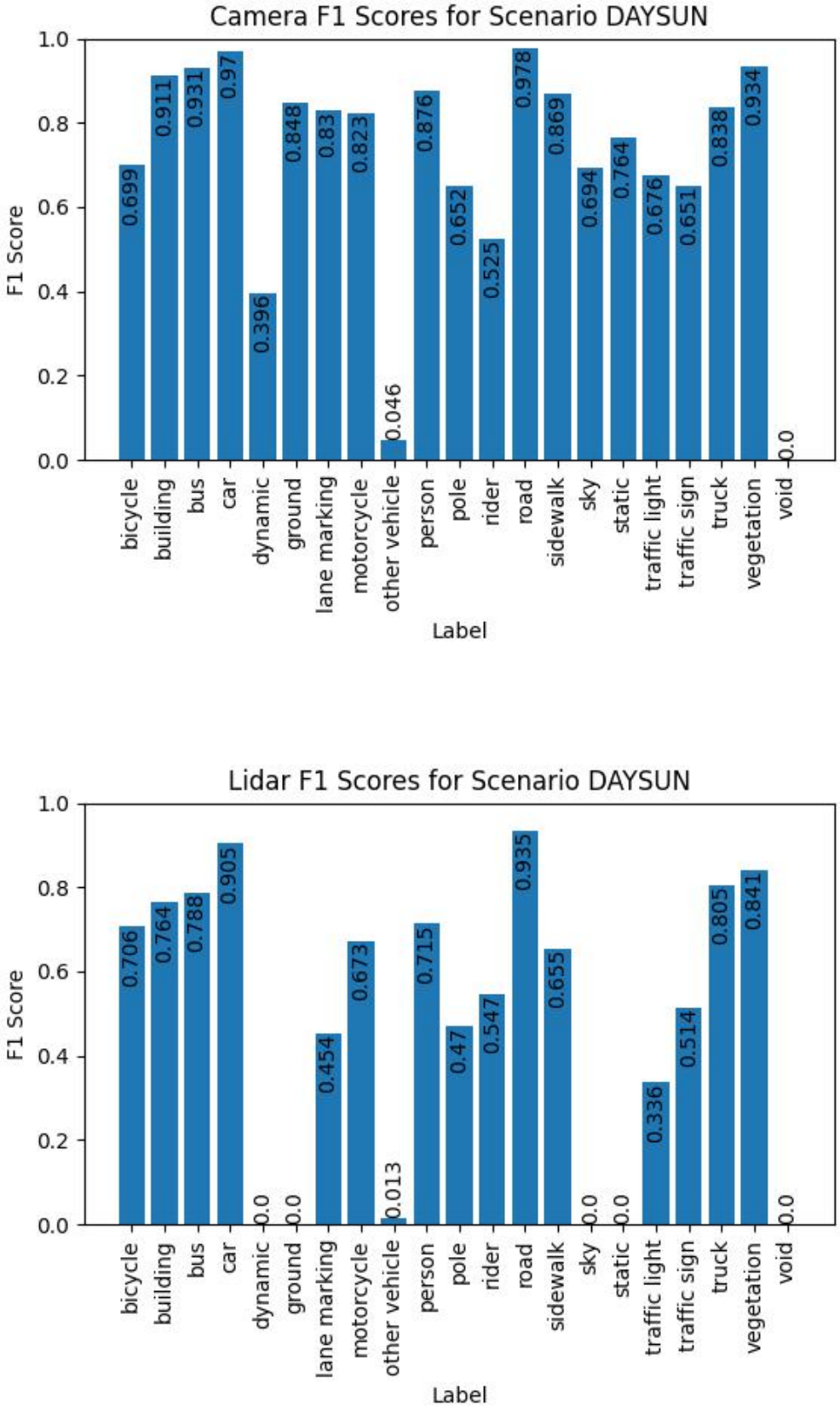

Figure 5.1: Unimodal F1 Scores for Scenario DAYSUN

Now we will compare the differences between scenarios. For this purpose, only classes that appear in all SUNNY scenarios are considered and compared. Most classes demonstrate a noticeable difference, with the DAYSUN scenario being the best performing and NIGHTSUN the worst performing in most cases. This could indicate that the daytime does have a considerable effect on sensor performance, most likely due to the lighting conditions as explained in chapter 2.1. Although it is important to note that some performance differences will likely be attributed to the lack of data for certain scenarios and are thus not completely representative for a comprehensive scenario analysis. This theory is supported by the fact that LiDAR scores also show a distinction when it comes to daytime. This is not expected behavior and might indicate that other unknown factors influence the scenario performance, or that the analysis of more data is necessary for a comparison as explained.

Sensor	Scenario	Accuracy (%)	Macro F1 Score (%)
Camera	DAYSUN	92.96	71.00
	NIGHTSUN	80.93	50.00
	TWILIGHTSUN	92.57	65.93
LiDAR	DAYSUN	80.81	48.20
	NIGHTSUN	74.61	33.83
	TWILIGHTSUN	79.58	40.53

Table 5.3: Unimodal Performance

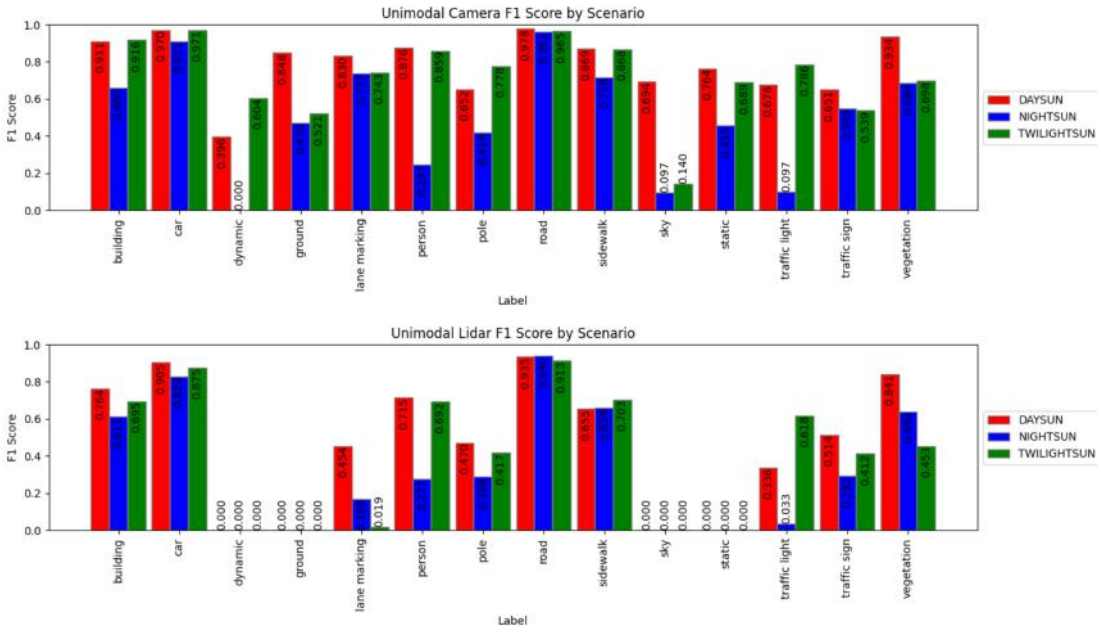

Figure 5.2: Unimodal F1 Score Comparison

5.4.2 LiDAR and Camera Fusion

Comparing the Sum Fusion techniques first, notable differences among the various weighting schemes are observed. The accuracy weighted Sum Fusion shows a clear improvement in both Accuracy and F1 scores over the unweighted method. This discrepancy under-

scores the limitations of the unweighted method, exemplified particularly in its handling of classes such as “sky,” “static,” and “ground,” where poor LiDAR performance negatively influences the fusion outcome immensely. This suggests that the absence of a weighting mechanism in the unweighted method allows less reliable outputs to disproportionately affect the results. The introduction of accuracy weighting in Sum Fusion mitigates these issues to some extent by preferring inputs from the more accurate camera classifier, thereby enhancing the scores of under performing LiDAR classes. However, this strategy is not without drawbacks, as evidenced by the reduced performance in the “rider” class, where LiDAR outperforms the camera in terms of F1 score. Both methods did not demonstrate the ability to effectively combine the classifier outputs, achieving overall lower performance than the camera classifier alone. The ability to improve the detection of single classes was only demonstrated for a few cases, while most individual class scores decreased compared to the camera. Transitioning to the F1-weighted Sum Fusion, the negative influence on low-performing classes is somewhat alleviated, though not entirely eliminated. The overall accuracy slightly increased compared to the other weighting methods, closely matching the camera’s Accuracy, and the macro-averaged F1 score also improved significantly. This enhancement suggests that F1 weighting offers particular benefits for the fusion of classes with disparate performance between sensors.

Fusion Method	**Weighting**	**Accuracy (%)**	**Macro F1 Scores (%)**
Product Fusion	None	85.62	56.66
	Accuracy Weighted	91.47	69.48
Sum Fusion	None	85.92	59.00
	Accuracy Weighted	89.67	65.74
	F1 Weighted	91.60	69.75
CLM Fusion	-	92.82	69.38

Table 5.4: Fusion Methods Performance

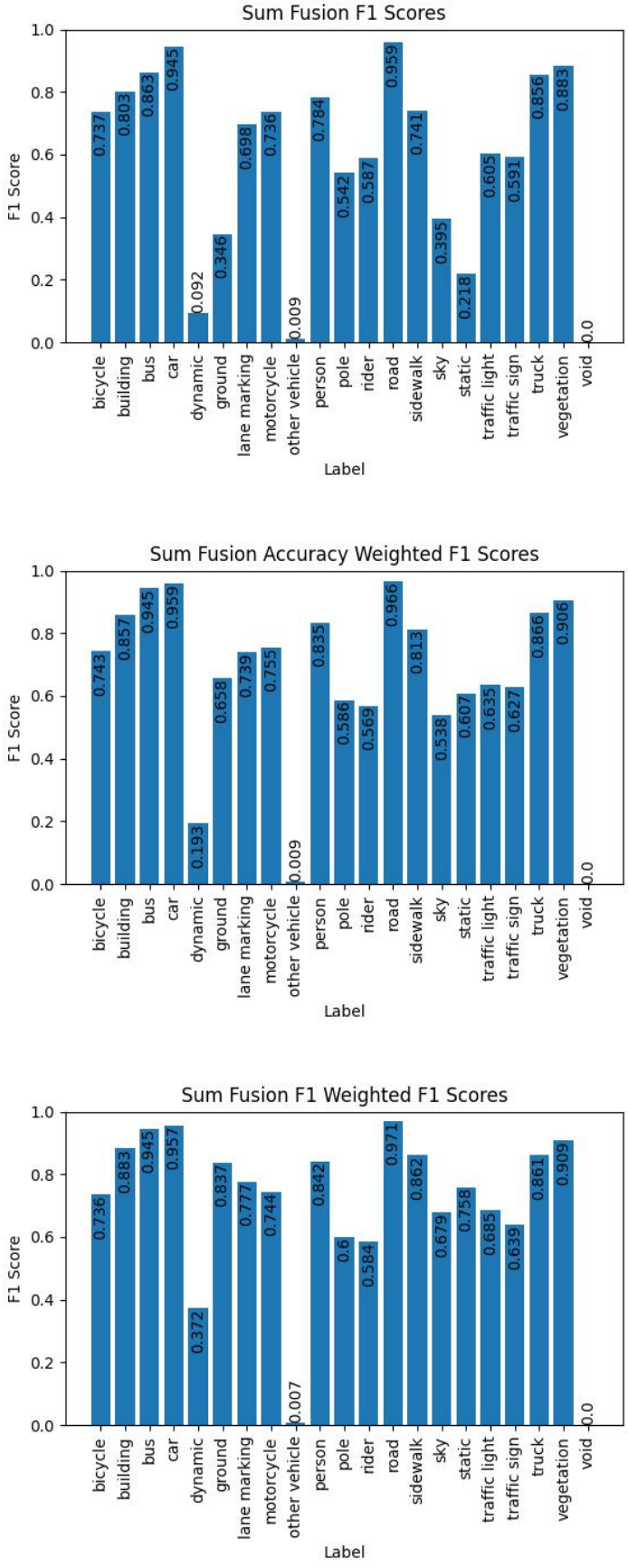

Figure 5.3: Sum Fusion F1 Scores for Scenario DAYSUN

Comparing the Product Fusion methods, the unweighted approach appears to have notable challenges related to overconfidence, particularly evident in the F1 scores for the classes "dynamic," "ground," "other vehicle," "sky," and "static." Each of these categories records an F1 score of close to 0% for the LiDAR classifier alone. Analysis of the individual classifier outputs reveals that this issue stems from the LiDAR classifier's extremely low probability outputs (often approaching zero) for these classes. Such overconfidently low probability outputs significantly influence the fusion process, as discussed in chapter 4.4. Conversely, the accuracy-weighted Product Fusion demonstrates improved performance across these problematic classes. This enhancement is likely attributed to the method's ability to increase entropy, thereby mitigating the detrimental effects of overconfidence. Additionally, this weighting strategy leads to improved performance for other classes as well, indicating a more robust and reliable fusion approach compared to the unweighted Product Fusion.

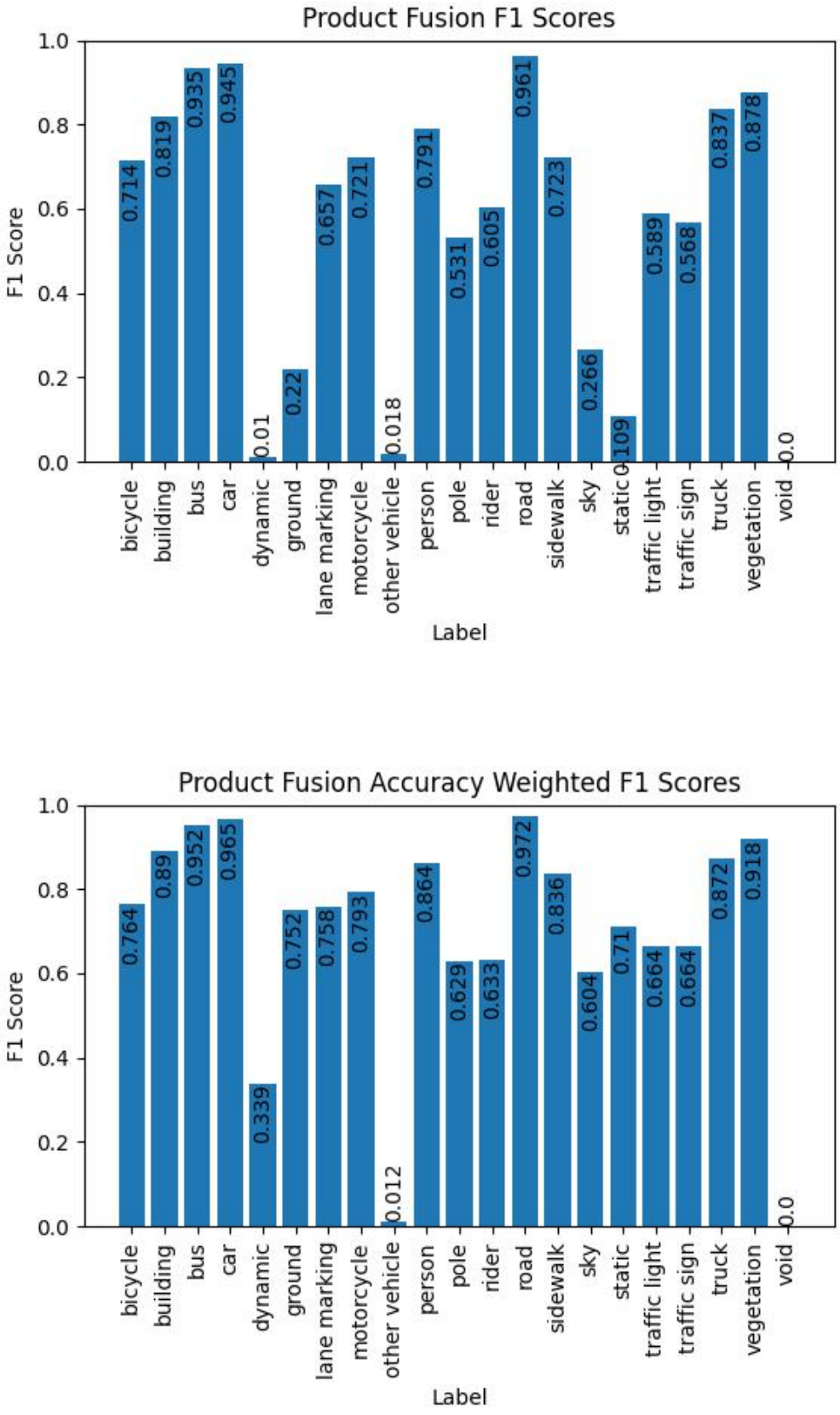

Figure 5.4: Product Fusion F1 Scores for Scenario DAYSUN

The CLM Fusion method, with an overall accuracy of 92.82%, stands out as the most effective among the fusion techniques evaluated, slightly surpassing both the F1-weighted Sum Fusion and the Accuracy-weighted Product Fusion, while its macro F1 score of 69.38% is about the same as achieved by those methods. A detailed analysis of performance across different classes reveals that CLM Fusion excels in scenarios where one modality performs poorly while the other performs well. In such cases, the F1 scores closely align with those of the better-performing modality, demonstrating the method's ability to leverage the strengths of each sensor effectively. This advantage is also observed in classes where both modalities perform well, often enhancing the results beyond the highest unimodal score. For instance, in classes such as "rider" the performance not only matches but exceeds that of the top-performing unimodal classifier, showcasing an improvement especially over the F1-weighted Sum Fusion. However, the CLM Fusion method exhibits a notable shortcoming in classes where both modalities perform poorly, such as "dynamic" and "other vehicle." In these instances, the fusion is resulting in diminished performance compared to the unimodal top output, as well as Sum- and accuracy weighted Product Fusion, indicating a limitation in the method's ability to compensate for simultaneous low performance across modalities. However, this behavior could potentially be beneficial for automated driving applications, as unreliable outputs have further decreased reliability, which indicates to higher level decision-making components that the information is not trustworthy. Although this is not possible to measure using the performance metrics used in this analysis. An interesting anomaly is observed in the class "sky," where the F1 score is unexpectedly lower than that of the camera. This reduction is likely attributable to a labeling error, as LiDAR technology typically does not detect the sky. Instances labeled as "sky" may involve corner cases where LiDAR points, positioned at the edge of an object, are erroneously projected onto pixels classified as "sky." This misclassification highlights the complexities of projecting a point onto an image. The discretization into pixels means a single pixel may encompass multiple classes but can only be labeled as one.

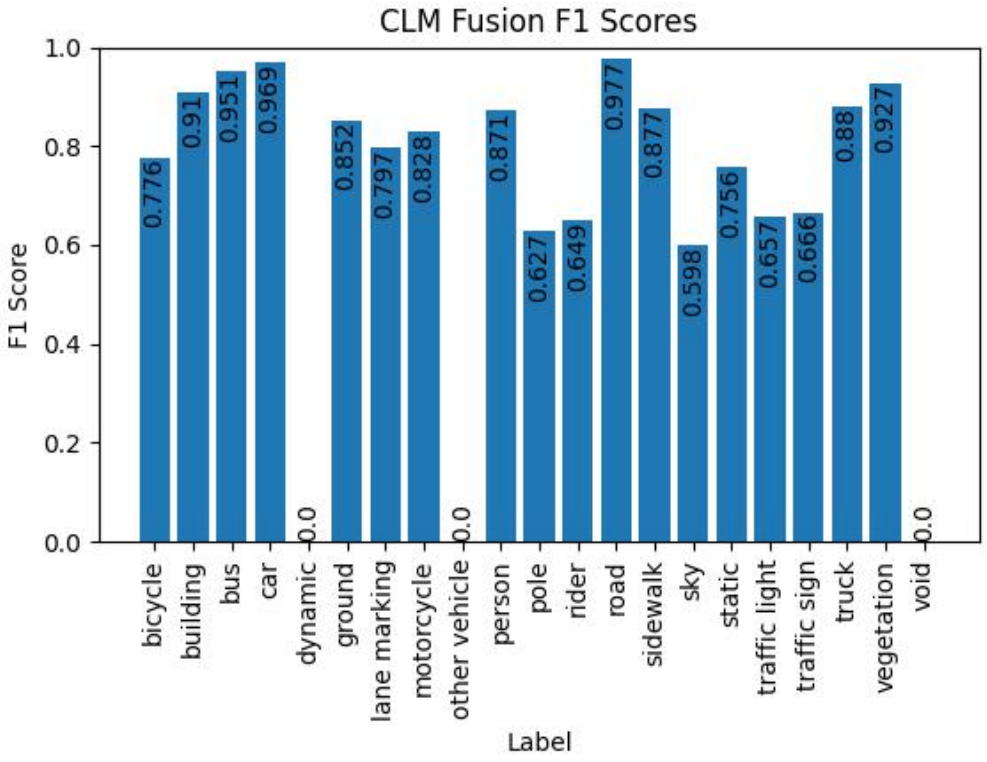

Figure 5.5: CLM Fusion F1 Scores for Scenario DAYSUN

5.4.3 Runtime

The analysis of runtime is crucial for assessing the practicality of implementing various fusion methods in real-world autonomous driving systems. The CLM Fusion necessitates a detailed examination of its runtime implications to gauge its feasibility. For this purpose, runtime evaluations were conducted using simulated data, represented by randomized tensors of dimensions 1920×1020×28, to mimic the classifier outputs assuming an ideal projection between modalities. The simulations were performed on an NVIDIA A40 graphics card with 48 GB GDDR6 graphics memory.

It is important to clarify that the runtime figures obtained are solely for comparative analysis between the different fusion methods and are not indicative of real-world operational scenarios. These figures should not be used to draw conclusions about the practical implementation of these methods in actual systems. Instead, they serve to illustrate the relative increase in computational demand when employing the CLM Fusion with an increasing number of modalities.

In the tests, the Product Fusion method was established as a baseline, with the fusion stage requiring an average time of 1.19 milliseconds. In contrast, the CLM Fusion exhibited a significantly higher average fusion time of 24.01 milliseconds, or a factor of 20.18. Another test simulating the fusion of three modalities resulted in a further runtime increased by a factor of 26.81, underscoring the exponential growth in computational load. This increase

in runtime for the CLM Fusion appears to be proportional to $n^m + s$ with n being the number of classes, m the number of modalities, and s an additional setup effort. This scaling is anticipated as the dimensions of the tensor shown in figure 4.3 expand with each added modality. These findings suggest a practical limit to the number of modalities that can be efficiently integrated using CLM Fusion, highlighting the trade-offs between enhanced fusion accuracy and computational efficiency.

6 Conclusion and Future Work

The CLM Fusion Algorithm has demonstrated improvements in accuracy when compared to baseline methods, and macro-averaged F1 scores are similar. Unlike the baseline methods, CLM Fusion was designed based on stochastic properties providing a solid mathematical foundation to weight and redistribute the output of classifiers based on their individual behavior, thereby increasing the trustworthiness of the output. As detailed in Chapter 4.3, the CLM Fusion approach offers several advantages over traditional weighting methods, particularly in its ability to effectively fuse modalities that exhibit considerable variance in class-wise detection performance. While the test results also suggest this theory, it appears that the F1 scores of classes that have no well performing modality drop. However, this may be caused by a more uniformly distributed fused output, indicating higher entropy. This can indicate higher level components to handle the output more critically, increasing reliability of the overall system. To utilize the CLM to its full potential, scenarios have been defined to distinguish between multiple different CLMs for a modality, each of which represents a more accurate approximation for the specific scenario. This can be utilized to enhance performance, although this has to be tested thoroughly in future work. This capability underscores the potential of CLM Fusion to leverage the strengths of diverse modalities, thereby enhancing the overall system performance. Looking ahead, there are opportunities to further enhance the algorithm's effectiveness by integrating additional modalities. This integration could exploit the diversity of sensor outputs to achieve a more robust complementary effect, potentially leading to superior performance metrics. However, it is important to recognize that increasing the number of modalities also introduces greater complexity to the fusion process. This complexity could limit the potential for further advancements, necessitating careful consideration in the design and implementation of extended multimodal fusion systems.

Another advantage of the CLM Fusion is the transparency. The CLM offers an intuitive and easy to understand weighting method based on sensor performance, which makes the comparison of different modalities based on their specific strengths and weaknesses simple. A disadvantage, on the other hand, is the need for an extra dataset-split used to create the CLM , because this makes it necessary to use a more extensive dataset for optimal results, especially if multiple CLMs for different scenarios are calculated, in which case the dataset also has to be more diverse regarding coverage of different scenarios.

The scope of this thesis can be seen as a proof of concept for the algorithm, and there are many different parts of it that can be optimized or expanded.

The Waymo Dataset used in this thesis unfortunately did not possess enough data recorded

for weather conditions, thus, only the daytime could be used for distinction. Moreover, additional information beyond time and weather, such as traffic situation, locality, speed, etc. would be useful to define different scenarios. This would offer the opportunity to select optimized scenario definitions based on the performance impact on the sensor, as well as the availability of related data. Fusion performance could be increased by using a different definition for scenarios that allows for a finer distinction. Selecting scenarios based on possible sensor performance impact could lead to a more precise estimation and thus to a more accurate CLM for each individual situation. Choosing scenarios that are roughly equally common could also lead to more precise estimations by avoiding comparatively uncommon scenarios with bad representation in the dataset.
For a more comprehensive analysis, it would be advisable to repeat the experiments using a more diverse dataset and test different scenario definitions.
Another problem is the loss of data due to projection. As discussed in chapter 4.2, the CLM calculation can only be done on data that is projectable to all other modalities involved in the fusion. This Problem can be amplified with the addition of modalities. Exploring methods to include more data in a meaningful way would help to use the dataset more effectively, and might be necessary to include more modalities.
An inherent challenge in multimodal fusion, particularly when employing the CLM, is the requirement for a comprehensive dataset that encompasses all modalities involved. The projection process, necessary for aligning different sensor outputs, often results in a significant reduction of usable data. As discussed in chapter 4.2, this is because the CLM calculations are contingent upon the availability of corresponding data points across all modalities, which may not always be feasible, especially with an increase in the number of modalities. To mitigate this issue, it is crucial to explore methodologies that can effectively utilize the available dataset without compromising the integrity of the fusion process. This might include the development of advanced projection techniques that maximize the overlap of data points across modalities, or the adoption of synthetic data generation techniques to augment the dataset where real data is insufficient.
Though, this does not mean that the incorporation of an additional modality into a fusion system necessarily diminishes the total number of points that can be fused during the inference phase. The modularity of the CLM Fusion is crucial, as it allows for selective inclusion or exclusion of modalities at specific data points based on their availability. Specifically, when modalities overlap only in certain regions, it is feasible to compute a CLM exclusively for these intersecting areas. Consequently, multiple CLMs can be established, each corresponding to a different combination of overlapping modalities. This approach ensures that all available modalities are utilized optimally, without the need to disregard regions where only a subset of modalities is present. By applying the appropriate CLM to each set of overlapping modalities, the fusion process becomes both flexible and efficient, maximizing the use of available data while maintaining the integrity of the fusion results across varied sensor coverage areas. Though, the problem of increased computational complexity with an increasing number of modalities remains for regions with multiple overlapping modalities.

While this thesis focuses exclusively on the development and evaluation of a multimodal fusion approach for semantic segmentation in automated driving, it is important to recognize the potential of other fusion techniques that, although not directly explored here, could further enhance detection performance when combined with the strategies proposed. Temporal fusion, for instance, involves integrating data across different time frames, which could be achieved by mapping sensor data from a previous timestamp to the current one and incorporating these predictions into the prior. However, this method presents challenges, such as accurately aligning data from consecutive timestamps in a dynamic environment where the sensor's position may change. Similarly, spatial fusion, which leverages data from neighboring points or pixels, could be implemented using neighborhood filters to refine the prior with spatially contextual information. While promising, these techniques also introduce additional computational complexity, as they require processing a larger volume of data for each point or pixel.

A Appendix

Symbol	Description
X	Random variable representing the true labels (ground truth).
X_j	A specific true label with index j.
$P(X)$	Probability distribution representing the likelihood of the true labels, represented as a vector for practical reasons.
$P(X_j)$	The probability of a specific true label j (value between 0 and 1).
S	A random variable representing a label estimate by the classifier.
S_i	A specific estimated label with index i.
$P(S)$, $P(S_i)$	Follow the same conventions as $P(X)$ and $P(X_j)$.
I	Random variable representing data points (pixel or points).
I_k	A specific data point from the dataset.
$P(I_k)$	All data points I_k have the same probability $P(I_1) = P(I_2) \cdots = \frac{1}{\lvert I \rvert}$.
$P(X_j \lvert S_i)$	The probability that a specific true label j is correct, given that the classifier has estimated a specific label i.
$P(X \lvert S)$	A matrix that provides the values for all X_j given all S_i.
$P(S \lvert I_k)$	The classifier's output for a specific data point, a probability distribution for this data point.
$P(X \lvert I_k)$	The probability distribution representing the ground truth for a given data point.
$\hat{P}(X)$	An estimate of the true labels based on made assumptions. Usually used for the fusion result.

Table A.1: Mathematical Notation Conventions

Bibliography

[Anw22] Aqeel Anwar. What are intrinsic and extrinsic camera parameters in computer vision?, Mar 2022.

[BCM05] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, 2005.

[BCM06] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Neighborhood filters and pde's. *Numerische Mathematik*, 105:1–34, 2006.

[CRD+19] Robin Chan, Matthias Rottmann, Radin Dardashti, Fabian Huger, Peter Schlicht, and Hanno Gottschalk. The ethical dilemma when (not) setting up cost-based decision rules in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[FHSR+21] Di Feng, Christian Haase-Schuetz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multimodal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2021.

[GKM+20] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, et al. A2d2: Audi autonomous driving dataset. *arXiv preprint arXiv:2004.06320*, 2020.

[HSC+19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[HSL+22] Keli Huang, Botian Shi, Xiang Li, Xin Li, Siyuan Huang, and Yikang Li. Multi-modal sensor fusion for auto driving perception: A survey, 2022.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[KGJM18] Douwe Kiela, Edouard Grave, Armand Joulin, and Tomas Mikolov. Efficient large-scale multi-modal classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[LCL+23] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17545–17555, 2023.

[LMZ+21] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A. Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2021.

[OK17] Sang-Il Oh and Hang-Bong Kang. Object detection and classification by decision-level fusion for intelligent vehicle systems. *Sensors*, 17(1), 2017.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[PWSR23] Maciej Pawłowski, Anna Wróblewska, and Sylwia Sysko-Romańczuk. Effective techniques for multimodal data fusion: A comparative analysis. *Sensors*, 23(5), 2023.

[RPM+22] Alina Roitberg, Kunyu Peng, Zdravko Marinov, Constantin Seibold, David Schneider, and Rainer Stiefelhagen. A comparative analysis of decision-level fusion for multimodal driver behaviour understanding. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1438–1444, 2022.

[SAS+22] Maria Shoukat, Khubaib Ahmad, Naina Said, Nasir Ahmad, Mohammed Hassanuzaman, and Kashif Ahmad. A late fusion framework with multiple optimization methods for media interestingness, 07 2022.

[SKD+20] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[SVHVP22] Zuhaib Ahmed Shaikh, David Van Hamme, Peter Veelaert, and Wilfried Philips. Probabilistic fusion for pedestrian detection from thermal and colour images. *Sensors*, 22(22), 2022.

[Thr02] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[WW19] Pasawee Wirojwatanakul and Artit Wangperawong. Multi-label product categorization using multi-modal fusion models. *CoRR*, abs/1907.00420, 2019.

[ZCVZ15] Richard Zhang, Stefan A. Candra, Kai Vetter, and Avideh Zakhor. Sensor fusion for semantic segmentation of urban scenes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1850–1857, 2015.

[ZdR17] Jinwei Zhou and Luigi del Re. Reduced complexity safety testing for adas and adf. *IFAC-PapersOnLine*, 50(1):5985–5990, 2017. 20th IFAC World Congress.

[ZLJ+21] Zhuangwei Zhuang, Rong Li, Kui Jia, Qicheng Wang, Yuanqing Li, and Mingkui Tan. Perception-aware multi-sensor fusion for 3d lidar semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16280–16290, October 2021.

[ZLW+22] Lili Zhao, Xuhu Lin, Wenyi Wang, Kai-Kuang Ma, and Jianwen Chen. Rangeinet: Fast lidar point cloud temporal interpolation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2584–2588, 2022.

[ZYJ+22] Wei Zhou, Qi Yang, Qiuping Jiang, Guangtao Zhai, and Weisi Lin. Blind quality assessment of 3d dense point clouds with structure guided resampling, 08 2022.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den ....