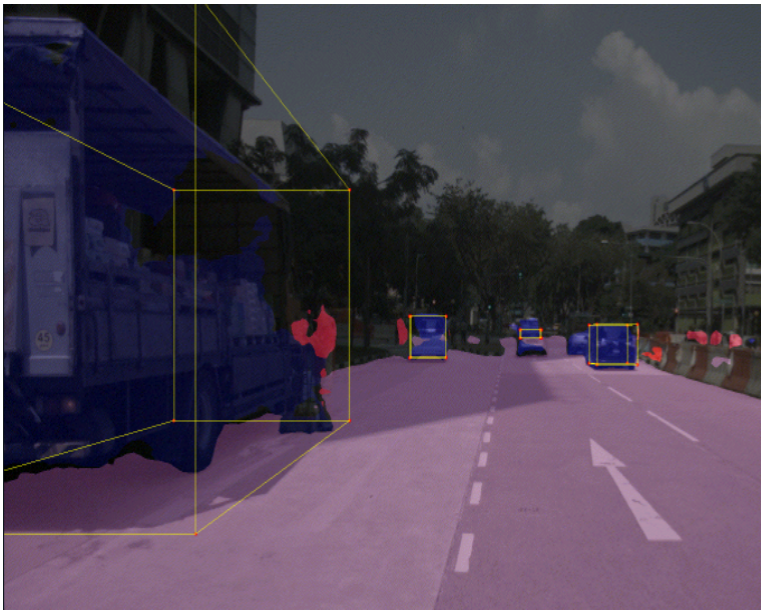
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA
Konputazio Ingeniaritza eta Sistema Adimentsuak Unibertsitate Masterra
Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

MASTER IN COMPUTATIONAL ENGINEERING AND INTELLIGENT SYSTEMS

MASTER'S THESIS

BEV2SEG_2: DRIVEABLE AREA SEGMENTATION FOR PREANNOTATION TASKS



Student: García Justel, Alan

Industrial Consultant: Sánchez Juanola, Martí
Supervisors: Barrena Orueechebarria, Nagore; Elordi Hidalgo, Unai

---

Academic Year: 2024-2025 **Date:** May 16, 2025

Abstract:

Keywords:

Contents

List of Figures

List of Tables

Acronyms

ADS Automated Driving Systems

ADAS Advanced Driving Assistance Systems

SAE Society of Automotive Engineers

V2I Vehicle-to-Infrastructure

V2X Vehicle-to-Everything

BEV Bird's-Eye-View

LiDAR Light Detection and Ranging

IPM Inverse Perspective Mapping

MLP Multilayer perceptron

CNN Convolutional Neural Network

FCN Fully Convolutional Network

VCD Video Content Descriptor

ViT Vision Transformer

AABB Axis-Aligned Bounding Box

IoU Intersection over Union

mIoU mean Intersection over Union

AP Average Precision

mAP mean Average Precision

1 Introduction

The development of Automated Driving Systems (ADS) has been a hot topic in the automotive industry for the last years. One of the fundamental aspects of an ADS is its perception system, as it is responsible for performing the obstacle detection and to provide a good environment representation for other systems, among others. This perception system can be divided into two main tasks: 3D object detection and local Bird's-Eye-View (BEV) map generation.

3D object detection is usually based on pointclouds obtained from LiDAR sensors, but as this sensors are costly a lot of research has been made in the field of camera 3D object detection. This approach has been gaining a lot of popularity carried by the improvements of the computer vision techniques with deep learning.

BEV segmentation aims to create a semantic representation of a vehicle's surroundings. This is a key component of an ADS perception system as BEV segmentation provides rich semantic information, precise localization, and absolute scales. This makes it useful for various tasks, including map reconstruction, prediction of agent intentions, and vehicle path planning.

To obtain BEV semantic segmentation from cameras, traditional methods first generate semantic masks in image space and then transform them into BEV space using Inverse Perspective Mapping (IPM). Although simple, it requires accurate camera parameters and assumes a perfectly flat ground surface, which limits its effectiveness. Moreover, while planar or low-height objects such as road curbs, lane markings, and the drivable area retain a meaningful metric representation in BEV space, objects with height appear distorted after the transformation.

With the objective of addressing this limitations, recent methods leverage data-driven techniques for BEV representation. [24] [25] [26]. However, to the best of our knowledge, no previous work has explored training a standard semantic segmentation model directly on BEV images to measure the difference between the inference of planar elements.

This master's thesis seeks to answer the question: *Does a model directly trained on semantic BEV images outperforms a typical segmentation model for plannar elements?*. Additionally, this work explores a technical application of BEV semantic segmentation for annotating vehicular scenes with occupancy, occlusion, and drivable area masks, contributing to the field of monocular 3D object detection given 2D semantic masks.

2 State of the art

Currently, the automotive industry is driving the development of ADS with the promise of reducing road accidents and minimizing both the human and economic costs associated with them [1]. Although automated driving has recently gained increased attention from the industry due to the rise of deep learning and computer vision, it has actually been present for more than 20 years.

Some of the earliest automated driving competitions, such as the DARPA Challenges in 2003 and 2005 or the Grand DARPA Urban Challenge in 2007, significantly boosted the development of ADS, attracting the attention of both technology companies and the automotive sector [2]. This progress has been accompanied by the establishment of best practices and standardization processes to ensure the safety and reliability of ADS. In this context, the Society of Automotive Engineers (SAE) has defined a progressive scale of automation, ranging from Level 0 (no automation) to Level 5 (full automation), specifying the degree of driver intervention required at each stage [3].

Today, most vehicles incorporate Advanced Driver Assistance Systems (ADAS), which operate at SAE Levels 2 and 3. However, Level 4 ADS already exists, such as those developed by Waymo and Cruise for robotaxis, as well as autonomous buses deployed in some cities. These systems are designed to manage fallback autonomously, without the need for human intervention [4].

This development has led to the creation of various strategies and architectures for ADS. In recent years, significant progress has been made in End-to-End solutions, which combine deep learning and reinforcement learning techniques to derive vehicle control actions directly from sensor data [5]. However, most approaches favor more traditional modular solutions, which divide the automated driving problem into specific sub-tasks, integrating solutions from fields such as robotics, computer vision, deep learning, and automatic control.

In the context of modular architectures, the adoption of best practices has facilitated the categorization of these sub-tasks into three main groups [6][7]:

- **Perception**, which refers to the autonomous system's ability to gather environmental information and extract relevant knowledge, such as the location of obstacles, traffic signs, and the vehicle's position.
- **Behavior planning**, which involves making decisions to achieve the vehicle's objectives, such as reaching a destination while avoiding obstacles and optimizing the trajectory.
- **Motion execution**, which pertains to the vehicle's ability to carry out the

planned actions by controlling steering, speed, and necessary maneuvers.

Furthermore, these sub-tasks interact with each other, with the vehicle's hardware, and with communication systems such as Vehicle-to-Infrastructure (V2I) or Vehicle-to-Everything (V2X) in the case of connected vehicles.

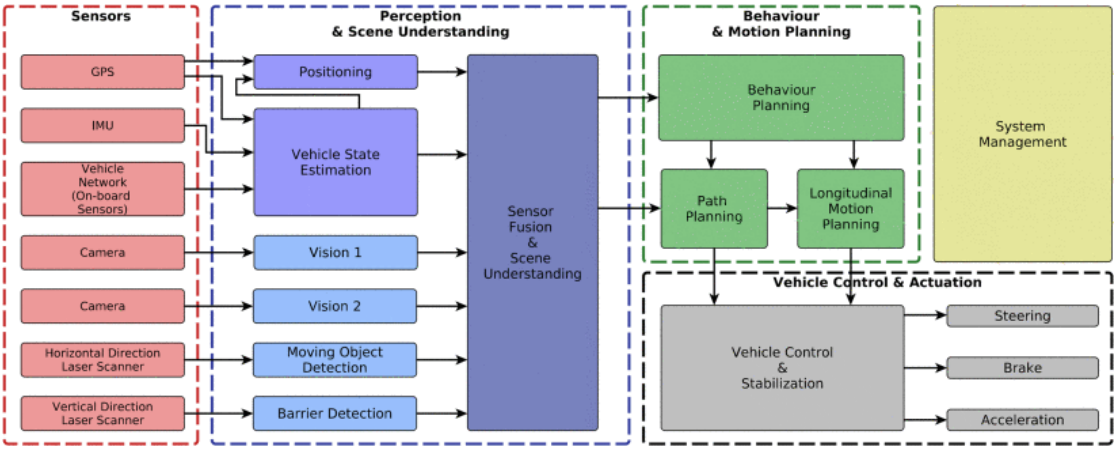


Figure 1: Typical ADS modular architecture.

In these types of architectures, an error in one sub-task can propagate and affect the performance of others, potentially compromising the overall system functionality. This is particularly critical in the perception module, as the quality of the obtained information directly impacts subsequent tasks such as localization, mapping, and planning. Therefore, ensuring robust perception systems is essential for the performance and safety of ADS.

Two of the main perception tasks in ADS are 3D object detection and BEV segmentation. 3D object detection is one of the most crucial tasks and is commonly based on point clouds obtained from LiDAR sensors. In the absence of LiDAR, an alternative is camera-based 3D detection, which aims to predict 3D bounding boxes in a common coordinate system using images [8].

On the other hand, BEV segmentation focuses on performing semantic segmentation of the environment, identifying drivable areas and lane boundaries in the vehicle's reference frame. Unlike object detection, BEV segmentation enables dense prediction of static environment classes, which is essential for local map construction, agent behavior estimation, and downstream tasks such as behavior planning in ADS [9].

This thesis is developed within the context of BEV semantic segmentation and

explores several key aspects and tasks related to perception systems in ADS.

2.1 Semantic segmentation

There are multiple tasks in computer vision, each addressing different levels of scene understanding. Image classification assigns a single label to an entire image, while semantic segmentation works in the pixel-level of the image assigning a category to each pixel based on a predefined label set. Instance segmentation extends this by detecting multiple objects in an image and segmenting them to delineate their boundaries. The fusion of semantic and instance segmentation is known as panoptic segmentation, which not only provides pixel-level semantic labels but also differentiates between instances of the same class. This thesis focuses on the semantic segmentation task, a well stablished research topic in computer vision.

Before the rise of deep learning based methods, semantic segmentation techniques relied heavily on hand-crafted features. The introduction of Fully Convolutional Networks (FCN) [10] boosted the field, as their convolutional layers enabled data-driven feature extraction directly from images, allowing for pixel-wise predictions well-suited for segmentation tasks. Additionaly, U-Net [11] further impoved this by incorporating an encoder-decoder architecture with skip connections to refine segmented object boundaries and preserve spatial information.

Furthermore, the introduction of image classification backbones like VGG [12] and ResNet [13] into segmentation models further boosted performance. This backbones are known by their strong feature extraction capabilities, and also the residual connections in ResNets helped mitigate the vanishing gradient problem, improving the learning of deep features and enhancing segmentation results.

From that point, semantic segmentation started to been framed as a structured prediction problem and the development of specific modules and architectures for this task was made. That's the case of dilated convolution [14], which increase the receptive field by "inflating" the Convolutional Neural Network (CNN) kernels with holes allowing models to capture more spatial context in their prediction.

Also, transformers have been adapted from Natural Language Processing (NLP) to vision tasks, starting with Vision Transformers (ViT) [15], which models images as patch sequences using self-attention. For semantic segmentation, architectures like SETR [16] replace CNN backbones with transformer for a better global understangin, while hybrid models like Swin Transformer [17] and MaskFormer balance efficiency and context modeling. These advancements improved segmentation performance, particularly in tasks requiring both fine details and long-range dependencies.

2.2 BEV semantic segmentation

Traditional methods [18] estimate local BEV maps using camera inputs under the flat-ground assumption applying IPM. However, these methods require accurate camera parameters which has led to research focusing on camera parameter estimation for BEV transformation [19] [20]. Another key challenge in BEV map generation is handling object occupancy and occlusion. ADS must be aware of objects dimensions and should account for uncertainty in vehicular scenes. However, estimating vehicle occupancy is non-trivial as the necessary perspective views of objects are often unavailable. In this context, there are many researchs that addresses the local semantic map estimation with different approaches.

Cam2BEV [21] applies IPM to transform multi-camera segmented input images into the BEV domain which is fed into the model to refine the BEV representation. To handle occlusions, Cam2BEV introduces an additional semantic class that explicitly marks occluded areas from all vehicle-mounted cameras. As the input of the model are already segmented images, an extra CNN is employed to test the method in real-world data.

HDMapNet [22] generates high-definition semantic maps from multi-camera input by employing a feature projection module that transforms image features into BEV space. The model first extracts image features and transforms them into the camera coordinate system with a shared Multilayer perceptron (MLP) backbone, and then projects them into BEV using camera extrinsics. Finally a semantic segmentation decoder is used.

PYVA [23] introduces a cross-view transformer that projects features from the front-view domain to the BEV domain. While similar to HDMapNet, PYVA differs in that it does not rely on camera parameters in the second transformation stage as the model is cappable of learning this transformation. Different to other methods, this work uses a GAN-based framework to manage occluions by estimating the vehicle's top view masks. However, this method is not suitable for multi-camera fusion.

Other approaches propose different architectures for BEV semantic segmentation. VPN [24] introduces a two-layer MLP module for multi-camera feature fusion, followed by a decoder for semantic segmentation in indoor scenes. LSS [25] proposes a unified framework that lifts 2D images into a 3D space by learning an implicit depth distribution and shows that their method is suitable for end-to-end motion plannig. M²BEV [26] transforms 2D image features into 3D voxels along projection rays and obtains an efficient BEV representation which supports multiple end tasks such as semantic segmentation or object 3D detection.

MonoLayout [27] tackles occlusion estimation by employing a standard encoder-decoder framework combined with adversarial training, making it suittable for predicting amodal layouts of the driving scene. BEVFormer [28] similarly enhances occlusion reasoning by leveraging attention mechanisms to fuse multi-view spatial-temporal features from historical BEV maps.

There are also methods that combines information form multiple sensor such as FishingNet [29] which extends VPN to use multiple sensors, or HDMapNet which is also cappable of using LiDAR sensors.

In summary, existing approaches to local BEV map estimation typically follow one of two strategies:

1. Performing early-stage segmentation on input images before refining the BEV representation.
2. Learning to embed image features into BEV space before passing them through a semantic segmentation decoder.

However, to the best of our knowledge, no previous work directly trains a model on already-reprojected BEV images.

Instead of applying semantic segmentation before transforming images into BEV space, we study whether training a segmentation model directly on top-view images improves the representation of planar elements compared to the traditional segmentation-first-then-IPM approach. Furthermore, we treat occupancy and occlusion mask generation as a post-processing step applied to BEV semantic segmentation, where 3D object detection is performed. This is integrated into a pre-annotation pipeline for vehicle scenes, contributing to advancements in monocular 3D object detection.

3 Methodology

This section details the experiments and implementations made to address the problems described in Section 1. On the first hand, the experimental design is introduced, tackling the model and dataset selection and the training and validation processes. Afterward, the implementation of the occupancy and occlusion masks preannotation pipeline is presented. Finally, the evaluation strategy of the pipeline is discussed to measure the quality of the resulting semantic masks and the monocular 3D detections for estimating the object's dimensions.

Thus, this project can be divided into two main blocks: BEV semantic segmentation experimentation and the design and implementation of the preannotation pipeline. Finally, all experiments for evaluating the models and the resulting framework are presented.

3.1 Segmentation experiment design: BEV2Seg_2

To address the hypotesis that *A semantic segmentation model trained with BEV images better segments planar elements* the process represented in Figure 2 has been designed. It follows two main approaches: first, performing image segmentation on regular RGB images and then reprojecting them to BEV; second, applying IPM to the original images and then segmenting them with the model.

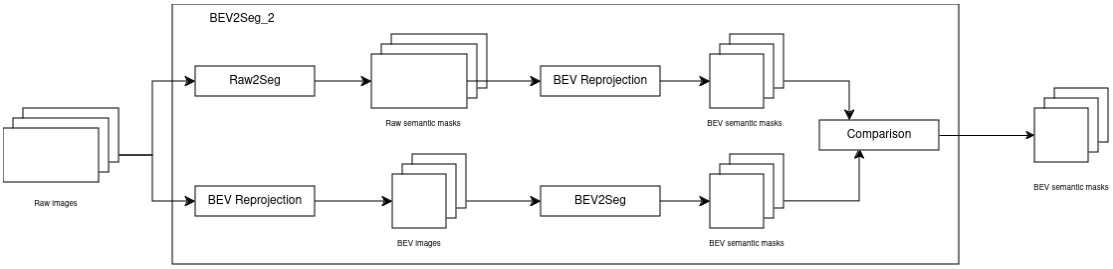


Figure 2: bev2seg_2 flow diagram.

Considering this flowchart, three main questions arises: (1) which segmentation model to use, (2) which dataset to train the models with, and (3) how to compare the train models and select the optimal output of the pipeline.

3.1.1 Segformer

There are multiple techniques and strategies for tackling semantic segmentation in both regular and BEV images (see Section 2).

Several models could be chosen to address the proposed hypothesis. The current state of the art includes both CNN and ViT-based models that achieve competitive results. As shown in Figure 1, there is no significant difference in accuracy and inference speed among the top-performing models. Moreover, many of these models have already been applied in the context of ADS. In this work, Segformer [30] has been selected as the semantic segmentation model due to its balance between performance and efficiency. Additionally, it is integrated into the Huggingface [31] ecosystem, which provides an optimized, parallelized implementation, facilitating distributed training.

Model Name	**Encoder**	**Params (M)** ↓	**FPS** ↑	**Cityscapes test mIoU (%)** ↑
DDRNet-39	-	32.3	-	80.4
PIDNet-L	-	36.9	-	80.6
DeeplabV3+	ResNet-101	62.7	1.2	80.9
SETR	ViT-Large	318.3	0.5	82.2
Segformer	MiT-B4	64.1	3.0	83.8

Table 1: Comparison of different models. Results are obtained from [32] [33] [30].

The Segformer model consists on an hierarchical Transformer encoder, which extract coarse and fine features, and a lightweight MLP decoder to directly fuse these multiscale features and predict the segmentation mask (Figure 3). Segformer comes with a series of Mix Transformer encoders (MiT) that share the same architecture but have different sizes: from MiT-B0 as the lightweigtest encoder for realtime inference, to MiT-B5 for best performance.

The training strategy involves using pretrained encoders from ImageNet-1K, attaching an untrained segmentation head decoder, and fine-tuning the entire model for semantic segmentation of vehicular scenes. Since the objective is to train the models directly on BEV images, which differ significantly from standard perspective images, the encoder layers will remain trainable rather than being frozen.

3.1.2 BEVDataset

In order to train the model on the segmentation task, a valid dataset must be selected. There are several semantic segmentation datasets for the ADS context such as Cityscapes [34], that defines 30 semantic classes and provides 5.000 frames with pixel-level high-quality annotations and 20.000 weakly annotated images; KITTI [35], which provices 400 annotated images with a 0.5 split for training and validation following the Cityscapes annotation format; ApolloScape [36], that provides 146.997 frames with corresponding pixel-level annonations and pose information for 25 labels; or NuImages, a subset of NuScenes [37], that contains annotated

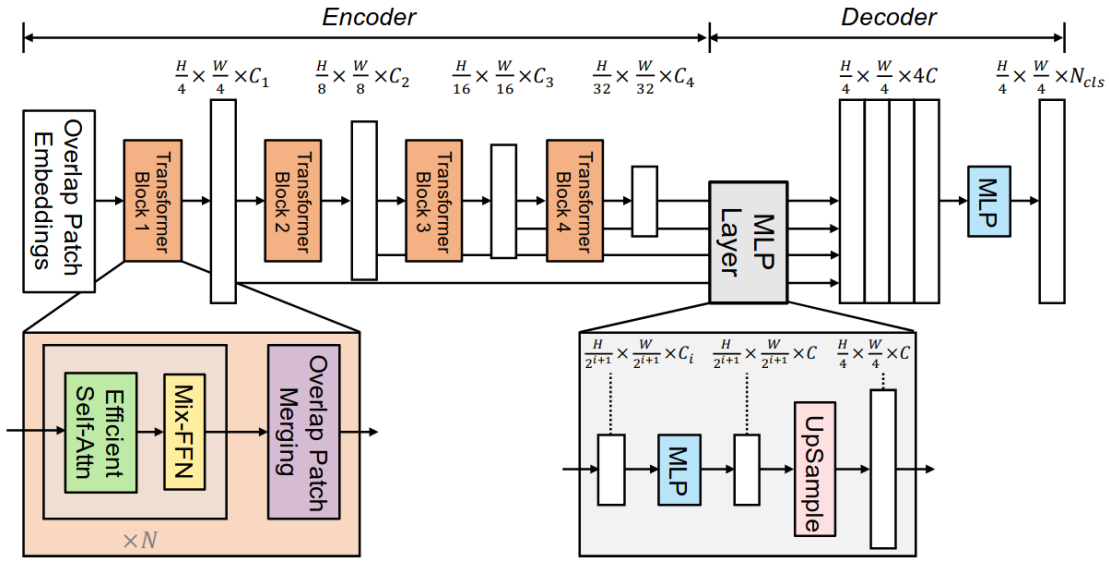


Figure 3: Segformer architecture

images on 26 different labels. For all of them, the ego pose and camera parameters metadata are provided.

There are other benchmarks [38] [39] but in orther to train the models a rich dataset is needed. Despite Cityscapes and ApolloScape being good options for this task, NuImages has been selected as it is one of the most used datasets for ADS tasks, it provides very accurate ego poses and camera parameters, it has 3D annotations and it has a very good documentation.

NuImages contains around 93.000 samples with aproximately 80% reserved for the training set and 20% for validation. Additionally, NuImages includes a private test set reserved for benchmark evaluations, whose annotations are not publicly available.

To train the models in the pipeline, a parser has been developed to convert NuImages into a sub-dataset named BEVDataset. This dataset has all front-camera images with NuImages annotations. Since the test annotations in NuImages are private, the validation set has been further splitted to ensure fair comparisons between models from different pipelines.

The conversion process is performed using a custom parser named 'OLDatasets', which transforms NuImages samples into the structured ASAM OpenLABEL [1] format, where metadata for each frame is stored. In the case of BEVDataset,

[1] https://www.asam.net/standards/detail/openlabel/

images are reprojected into the BEV domain using the Video Content Descriptor (VCD) library [40]. This library provides tools to handle OpenLABEL annotations and manage both 2D and 3D data efficiently.

The OLDatasets parser extracts the camera parameters for each sample and computes a lookup table to apply IPM reprojection. Using this data, semantic pixel masks are generated and reprojected along with the original images into the BEV space. Since this reprojection involves image warping, the interpolation method must be carefully chosen:

- Linear interpolation is applied to images.
- Nearest neighbor interpolation is used for semantic masks to preserve pixel class integrity.

The virtual BEV camera parameters are fixed: BEV reprojection is generated at the 'vehicle-iso8855' coordinate frame using a regular grid with a cell spacing of 1 meter, covering a total distance of 30 meters in front of the vehicle and 1 meter behind it. The resulting images have a resolution of 1024×1024 pixels.

Finally, BEVDataset contains a total of 16.427 images, distributed as shown in Figure 4. More details about the dataset conversion and especifications can be fount at Appendix A.

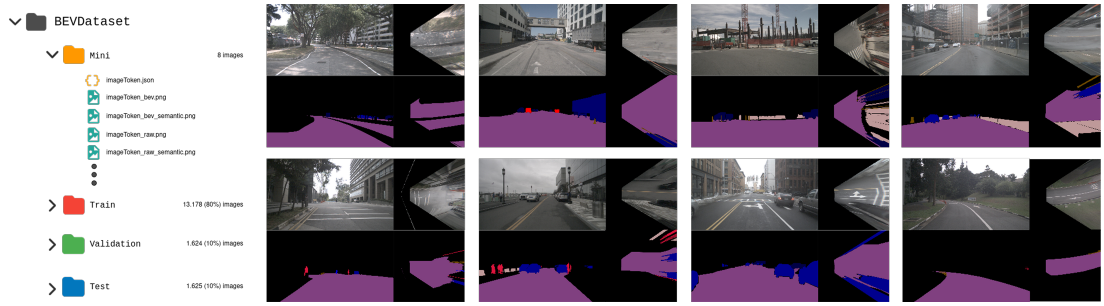
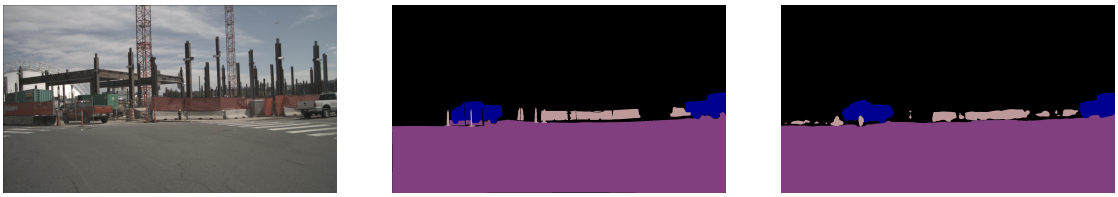


Figure 4: BEVDataset structure and mini set samples

3.1.3 Validation and comparison

When validating and comparing different segmentation models, it is essential to use a common dataset and apply appropriate metrics that allow for a quantitative and objective evaluation of model performance. The first requirement is met by using the test set of the dataset, which has not been previously exposed to the models being compared. Regarding evaluation metrics, as shown in Figure 5, where a semantic mask predicted by a segmentation model is displayed alongside the corresponding annotated ground truth mask, it is intuitive to assume that an

accurate prediction is one that maximizes the overlap between the predicted mask and the ground truth while also avoiding excessive overflow beyond the boundaries of the ground truth regions.



(a) (b) (c)

Figure 5: Semantic masks example: (a) original image, (b) colored semantic ground truth, (c) colored inferenced semantic mask.

To quantify this, two distinct but related metrics are commonly used: the Dice coefficient (also known as the F1-score) and the Jaccard index, also known as Intersection over Union (IoU), whose formal definitions are presented in Formula 1. Both metrics are bounded within the range [0, 1], where 0 indicates no overlap between prediction and ground truth, and 1 indicates a perfect match between the two masks.

$$\begin{aligned} \text{Dice}(A,B) &= \frac{2\,|A \cap B|}{|A| + |B|} = \frac{2\,TP}{2\,TP + FP + FN} \\ \text{Jaccard}(A,B) &= \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \end{aligned} \tag{1}$$

Suppose we start from an image with resolution $H \times W$ and aim to predict N different classes. The model output will be a tensor of dimensions $H \times W \times N$, representing N masks of size $H \times W$, where each element encodes a class-specific confidence value. However, in semantic segmentation, only the best prediction per pixel is retained, resulting in N binary masks that represent the final semantic classifications of the original image. In this context, the described metrics are computed independently for each of the N classes.

For a fixed ground truth, both metrics are positively correlated: if classifier A outperforms classifier B according to one metric, it will also outperform it according to the other (see Appendix B). This might suggest that the choice between the two metrics is arbitrary; however, differences become apparent when averaging scores over a set of predictions. At this stage, although both metrics may agree that model A performs better than model B, they differ in how much better one model is compared to the other.

In general, the IoU metric tends to penalize poorly classified instances more severely than the F1-score, thereby offering a performance measure that reflects worst-case behavior. In contrast, the F1-score is more representative of average-case performance. As a result, when computing averages across multiple predictions, IoU is more sensitive to significant outliers, while F1-score tends to be more forgiving in such cases.

Taking all this into account, mean Intersection over Union (mIoU) has been chosen as the primary metric for evaluating and comparing models in this work.

3.2 Driveable area automatic annotation

This section details the selected approach used to automatically annotate occupancy and occlusion masks in vehicular scenes. The method employs a 2D-to-3D approach, using estimated depth maps from monocular images to approximate object dimensions and compute the corresponding masks.

As shown in Figure 6, the method can be divided into four main stages: depth estimation, where a depth map is generated from monocular front images; frame point cloud estimation, which creates a 3D representation of the current scene; projection of the perspective semantic mask onto the 3D point cloud to infer object instances for each selected semantic class; and the final computation of occupancy, occluded, and drivable areas by combining 3D object information with BEV semantic masks.

The system's output is annotated in the OpenLABEL format and integrated with the WebLABEL [43] ecosystem, enabling further fine annotations by the user. Additionally, a visualization tool has been developed using the Open3D [42] library to facilitate the rendering process of vehicular scenes.

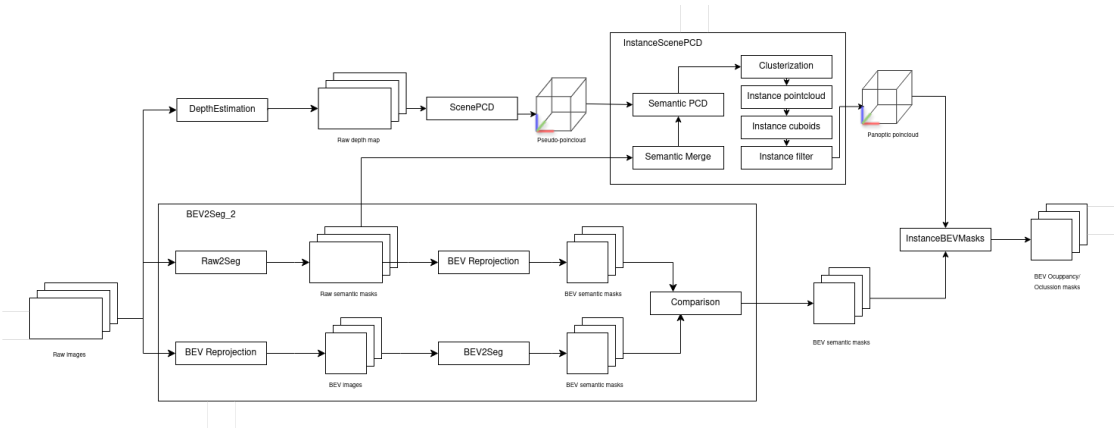
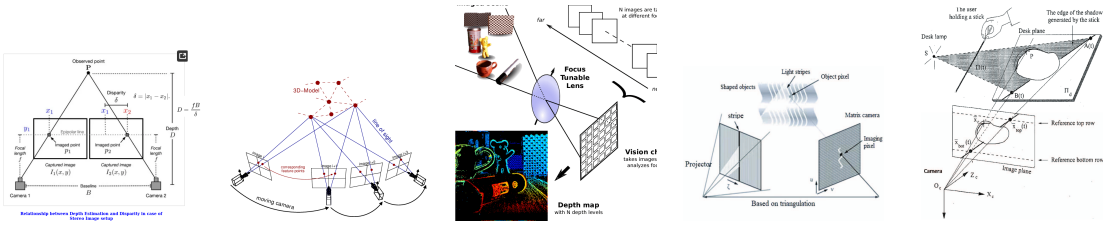


Figure 6: Annotation flow diagram.

3.2.1 Depth estimation

Traditional depth estimation techniques [44] rely on geometric and photometric marks to infer depth from images. For example, stereo vision (7a) uses two or more cameras to capture the same scene from slightly different viewpoints, allowing depth to be estimated through disparity between the images. Similarly, structure from motion (SfM) (7b) relies on multiple images taken at different times or positions to compute motion parallax and infer scene's depth. Other multi-view techniques include shape from defocus (7c), which estimates depth based on the amount of blur in the images, and photometric stereo (7d), which uses multiple images captured from the same viewpoint but under different lighting directions to recover surface orientation and depth.

In the case of single-image depth estimation, traditional methods like shape from shading (7e) attempt to infer depth based on lighting and texture patterns. However, these methods typically rely on strong assumptions about the scene's geometry and lighting conditions, and thus only work reliably in controlled or well-defined environments.



(a) (b) (c) (d) (e)

Figure 7: Traditional depth estimation techniques: (a) stereo-depth, (b) structure from motion, (c) depth-from-defocus, (d) photometric-depth, and (e) shape from shading.

With the continuous development of deep learning technologies, depth estimation and 3D reconstruction methods based on these approaches are constantly updated. They have become a powerful choice for monocular depth estimation as they offer high reconstruction accuracy and computational efficiency. However, a common limitation of such models is their dependency on training data as performance tends to decrease if the target image is not consistent with the learning database.

This is where Depth-Pro [45] comes into as it is 'a foundation model for zero-shot metric monocular depth estimation'. Trained on a large and diverse dataset, Depth-Pro generalizes well across multiple real-world scenarios and achieves state-of-the-art performance without requiring fine-tuning. For this thesis, Depth-Pro

has been chosen as the monocular depth estimation model to generate depth maps for annotation purposes.

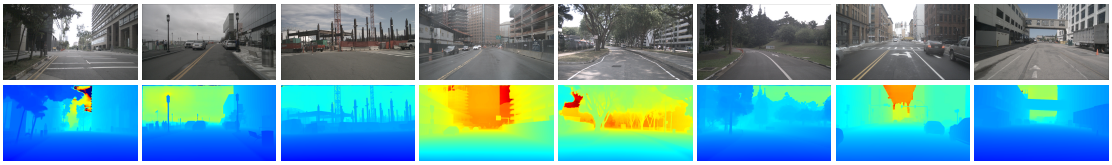


Figure 8: Depth maps of BEVDataset 'mini' estimated with Depth-Pro model.

3.2.2 Scene PCD

When an image is captured, the 3D geometry of the scene is projected onto a 2D representation, which is then rasterized into the pixel space of the image. This transformation from 3D world coordinates to 2D image coordinates is known as the *camera projection process*. In its simplest form, this is described by the *pinhole camera model*, illustrated in Figure 9. In this setup, $\mathbf{O}_w$ represents the world coordinate frame, $\mathbf{F}_c$ denotes the camera coordinate frame (typically located at the camera's optical center or pinhole), and $\mathbf{F}_i$ defines the 2D image plane.

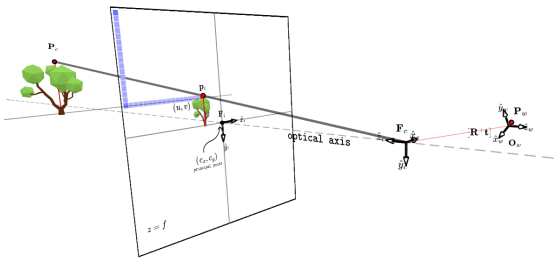


Figure 9: Ideal perspective camera model.

Starting with a 3D point expressed in world coordinates, $\mathbf{P}_w = (x_w, y_w, z_w)^\top$, the point can be transformed int the camera coordinate frame using a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. These are known as camera's extrinsic parameters and are commonly combined into a single homogeneous transformation matrix denoted as $[\mathbf{R} \,|\, \mathbf{t}]$.

$$\tilde{\mathbf{P}}_c = \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = [\mathbf{R} \,|\, \mathbf{t}]\, \tilde{\mathbf{P}}_w \tag{2}$$

Once the point is expressed in the camera coordinate frame $\mathbf{F}_c$, it is projected onto the image plane $\mathbf{F}_i$ using the intrinsic parameters of the camera, which describe how the 3D geometry of the scene is mapped to the 2D image sensor. Under the pinhole camera model, this projection involves a perspective division (to normalize the coordinates) followed by a transformation using the intrinsic matrix $\mathbf{K}$, which encodes the focal lengths and maps the continuous 2D space to the discrete pixel grid of the digital camera, while also considering the sensor's displacement of the principal point (c_x, c_y). The final pixel coordinates can be computed as:

$$\tilde{\mathbf{p}}_i = (u, v, 1)^\top = \mathbf{K} \cdot \frac{1}{z_c} \tilde{\mathbf{P}}_c \tag{3}$$

where $\mathbf{K}$ is the intrinsic calibration matrix, typically given by:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

Here, f_x and f_y are the focal lengths in pixel units along the $\hat{x}_i$ and $\hat{y}_i$ axes.

However, the images used in this thesis were captured using lens-based cameras, which introduce radial and tangential distortions. These distortions are taken into account by rectifying the normalized 3D coordinates in the camera frame before applying the camera's intrinsic matrix $\mathbf{K}$.

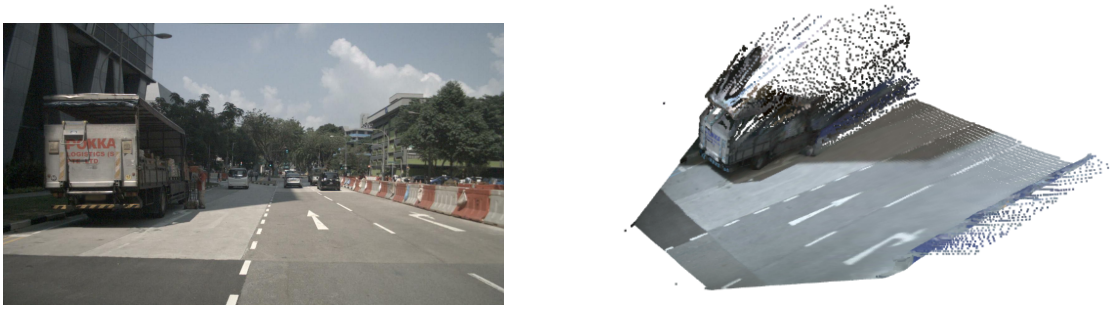
Despite this, the objective of this module is to perform the inverse process: reconstructing the 3D structure of the scene from 2D image pixels. This process is known as *reprojection*, and it has the challenging task of recovering the third spatial dimension, which is inherently lost during the image acquisition process. To address this, a depth map $D_{h \times w}$ is estimated (see Section 3.2.1), where h and w correspond to the height and width of the image, respectively. It is also assumed that the camera parameters (intrinsics and extrinsics) are known.

The approach employed for this task is detailed in Algorithm 1 where for each pixel column in the vertical range $[0, 1, \ldots, h-1]$, a 3D ray is projected from the camera center and then intersected with the depth value estimated for that pixel. The final result is a dense 3D point cloud which can be seen in Figure 10.

Review this algorithm to match the camera's projection notation.

Algorithm 1 Pointcloud calculation

Input: Depth map $D_{[h\times w]}$, camera intrinsics K
Output: Pointcloud $P^{raw}_{[3\times h\times w]}$
Initialize $P_{[3\times h\times w]} \leftarrow \mathbf{0}$
for i in $[0, 1, \ldots, h-1]$ **do**
    $pix\_coords_{3\times w} \leftarrow \begin{bmatrix}0\text{:w-1} & \mathbf{i}_w & \mathbf{1}_w\end{bmatrix}$
    $cam\_rays_{3\times w} \leftarrow \texttt{UnDistord}\left(\texttt{Normalize}\left(K^{-1}\cdot pix\_coords\right)\right)$
    $P^{raw}[:, i, :] \leftarrow cam\_rays[2, :] \cdot D[i, :]$ ▷ Update the i-th row of P. The ray's z-value is the depth
end for
return P^{raw}



(a) (b)

Figure 10: Source image (a) and reconstructed pointcloud (b).

3.2.3 Instance scene PCD

The objective of this module is to compute instance-level point clouds from the previously reconstructed scene point cloud using the corresponding semantic mask of the input perspective image (obtained as one of the BEV2Seg_2 module's output). The semantic and geometric information is fed into this module, where instance differentiation is performed for specific object classes as shown in Figure 11. This enables the isolation of individual instance point clouds and the computation of their enclosing cuboids. The resulting panoptic point cloud provides a 3D object-level representation of the scene, which can be used as input for subsequent modules. It is also interesting to point out this is achieved from a single monocular image, demonstrating the system's capability for 3D object detection without requiring multi-view or stereo data.

Firstly, the input semantic mask is preprocessed with a label merging step in which semantically similar labels are unified under a single target class. This step helps mitigate noise and inconsistencies in the model's predictions, which could negatively impact the subsequent instance estimation process. The merging

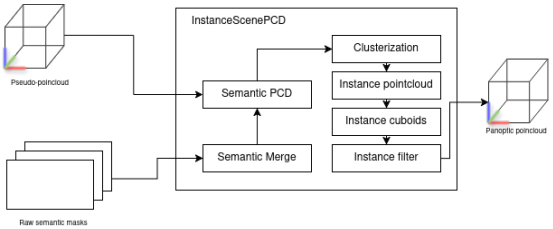


Figure 11: Instance scene computation diagram.

policy is defined by default as shown in Table 2. For example, the target class `vehicle.car` gathers all NuScenes vehicle categories that seem like cars, such as buses, trucks, trailers, and emergency vehicles.

Merged Class	**Original Classes**
`background`	`background`
`animal`	`animal`
`human.pedestrian`	`human.pedestrian.adult`, `human.pedestrian.child`, `human.pedestrian.construction_worker`, `human.pedestrian.personal_mobility`, `human.pedestrian.police_officer`, `human.pedestrian.stroller`, `human.pedestrian.wheelchair`
`movable_object.barrier`	`movable_object.barrier`, `movable_object.debris`, `movable_object.pushable_pullable`, `movable_object.trafficcone`, `static_object.bicycle_rack`,
`vehicle.car`	`vehicle.bus.bendy`, `vehicle.bus.rigid`, `vehicle.car`, `vehicle.construction`, `vehicle.emergency.ambulance`, `vehicle.emergency.police`, `vehicle.trailer`, `vehicle.truck`
`vehicle.ego`	`vehicle.ego`
`vehicle.motorcycle`	`vehicle.bicycle`, `vehicle.motorcycle`
`flat.driveable_surface`	`flat.driveable_surface`

Table 2: Class merge dictionary used for semantic simplification.

Once the semantic mask has been preprocessed and the point cloud computed, a pre-filtering step is applied to remove points that fall outside predefined bounds (by default, $[10, 5, 30]$ meters in the $\hat{x}_c$, $\hat{y}_c$, and $\hat{z}_c$ axes). Then, the semantic mask is projected onto the point cloud to create a semantic point cloud. This process is straightforward, as each 3D point is associated with the corresponding semantic label based on the pixel indices. As a result, each 3D point is labeled with its

appropriate semantic class (see Figure 12). Additionally, each semantic point cloud is tagged with a 'dynamic' flag, which indicates whether the associated object is likely to move. Only 3D semantic points with this flag set are passed through the instance detection process. This helps avoid unnecessary processing of labels such as background, road barriers, or driveable areas.

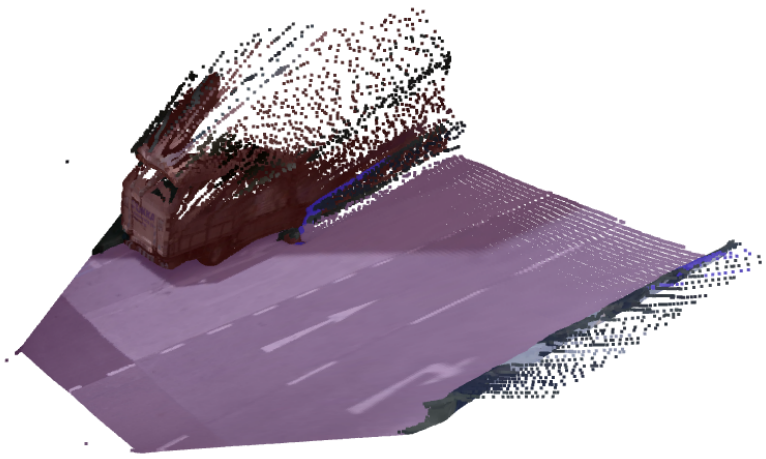


Figure 12: Colorized semantic pointcloud.

The instance detection process (Algorithm 2) is carried out using a clustering algorithm that extracts the different instances from each dynamic semantic point cloud. The chosen algorithm is DBSCAN, which creates good instance clusters and removes part of the trail left by the depth estimation on object edges. Then, for each computed instance point cloud, the minimal camera Axis-Aligned Bounding Box (AABB) is calculated by taking the minimum and maximum coordinates along each of the camera axes $\hat{x}_c$, $\hat{y}_c$, and $\hat{z}_c$. Oriented bounding boxes were not used in this work.

Estimating the correct orientation of a bounding box is a challenging task, especially when only a partial view of the object is available (see Figure 13). Depending on the viewpoint and the amount of visible surface, traditional methods are very likely to produce inaccurate orientation estimates. Modern 3D object detection models typically address this issue by learning the statistical distribution of common object shapes and orientations. Datasets like NuScenes even provide metrics such as Average Scale Error (ASE) and Average Orientation Error (AOE) to evaluate the accuracy of these predictions.

However, the main goal in this thesis is to estimate the physical space occupied by a vehicle in the scene for pre-annotation purposes, not precise orientation or object alignment. In this context, using an AABB may slightly overestimate the

occupied space when objects are not aligned with the camera's reference frame, for example during turns or when vehicles are parked at an angle. Nevertheless, this estimation does not pose a significant problem since these overestimated regions are not of special interest in the annotation process.

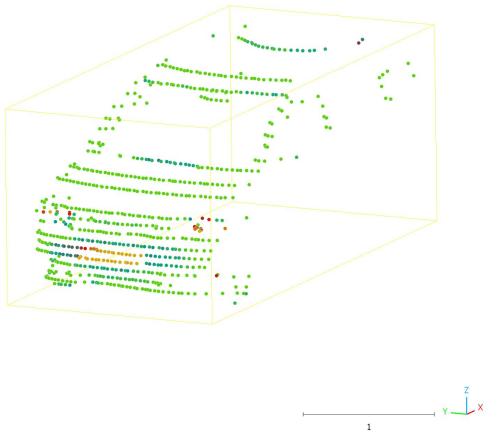


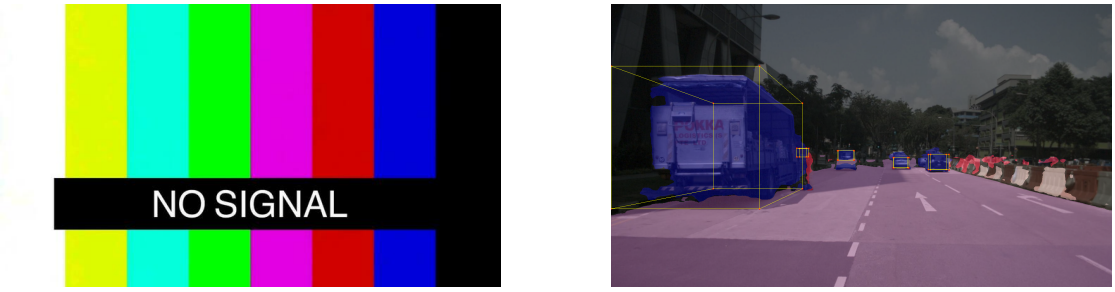
Figure 13: Oriented bounding box problems.

Change oriented bounding box problems figure. Complete this algorithm.

Algorithm 2 Instance Pointcloud Computation

```
Input: P^inst panoptic pointcloud
Output: P^inst filtered panoptic pointcloud
for label in P^inst do
    for i in P^inst [label] do
        B ←
    end for
end for
return P
```

Finally, a filtering process is performed to ensure there are no noisy, redundant, too far away or floating instances. Also, it ensures there are no overlapping bounding boxes by checking each new instance's 3D bounding box against those already accepted. If an overlap is detected, the function compares their sizes: the larger box is kept, and the smaller one is removed or ignored. This helps avoid redundant or conflicting instances that may represent the same object or spatial region, improving the clarity and consistency of the data. The final results of this module are shown in Figure 14.



(a) (b)

Figure 14: Resulting instances in 3D (a) and projected into the front camera (b).

3.2.4 Instance BEV mask

Once the 3D instance detection is performed and the panoptic point cloud is generated, the final occupancy and occlusion masks can be estimated using the semantic BEV masks.

To associate the BEV semantic masks with the detected 3D instances, the semantic masks are first processed using a connected components labeling algorithm, which extracts the individual, non-connected regions for each semantic class. Then, for each instance in the panoptic point cloud, the base of its 3D bounding box is obtained and an intersection factor is calculated between each pair of base bounding boxes and connected mask regions. Using this intersection measure, the base polygons and connected masks are associated accordingly so each mask is associated with one polygon but a polygon may have multiple masks. This is represented in Algorithm 3 and results in a final representation of occupancy, defined by the base of the 3D bounding box, and occlusion, which is captured by the distorded shape of the semantic mask from the BEV perspective.

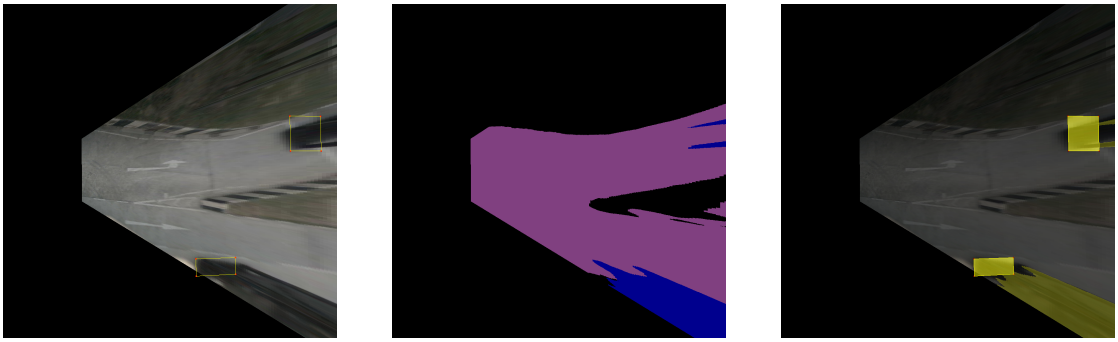
A single cuboid base can be associated with multiple BEV masks because a 3D instance may appear fragmented in the BEV semantic view. When connected components labeling is applied to these masks, each fragment is identified as a separate region. However, since all fragments originate from the same 3D object, they are associated with the same cuboid base.

Complete this algorithm.

Algorithm 3 Occupancy Occlusion masks

1: **Input:**
2: **Output:**
3: **return** x

This method allows for estimating the vehicle's extent while handling certain ambiguities. For example, when a vehicle is perfectly aligned with the camera view, only the rear width and height are clearly visible (see Figure 13), making the depth estimation ambiguous. However, when the vehicle is rotated relative to the camera, its dimensions become more discernible. In such cases, the 3D detection phase, despite the ovestimation of AABB, enables a more reliable approximation of the vehicle's actual footprint.



(a) (b) (c)

Figure 15: BEV occupancy and occlusion masks. (a) Input 3D detections; (b) input BEV semantic masks and (c) final BEV occupied-occluded mask.

The final BEV masks have to be generated using the labels of this table. This is not done yet.

Name	ID	Color (RGB)
background	0	(0, 0, 0)
occuped	1	(0, 74, 179)
occluded	2	(101, 164, 255)
driveable	3	(255, 192, 101)

Table 3: Semantic labels defined for resulting masks

3.2.5 Evaluation strategy

As early mentioned, the final objective of the annotation pipeline is to provide occupancy, occlusion and driveable area masks. To achieve this, the process starts from semantic masks of the environment, both in the camera and BEV domains. A crucial intermediate step in this pipeline is the 3D object detection from monocular images in the camera view to estimate the dimensions and positions of obstacles.

As this detection is performed using pointcloud clustering algorithms it is particularly important to assess its performance using appropriate evaluation metrics.

Another key aspect of the evaluation involves assessing the quality of the generated BEV masks. Therefore, the evaluation methodology must address two distinct components: the accuracy of the monocular 3D object detection, and the fidelity of the resulting BEV masks when compared to a defined ground truth. This evaluation uses the pipeline's output OpenLabel file for a vehicular scene, alongside the corresponding ground truth OpenLabel file which contains annotated elements. A 'vehicular scene' is defined here as a sequence of frames with annotations that include not only the environment, but also vehicle position and odometry.

3.2.5.1 3D detections evaluation

The nuScenes metrics [37] were considered as a starting point as this dataset was specifically developed to facilitate the training and evaluation of 3D perception strategies. A central component of its evaluation policy is the matching between predictions and ground truth, which is performed using the 2D distance between the centers of the predicted and ground truth bounding boxes projected onto the ground plane. A prediction is considered a true positive if this distance falls below a predefined threshold. Then, Average Precision (AP) is computed across several distance thresholds ($\{0.5, 1, 2, 4\}$ meters), and the final mean Average Precision (mAP) score is obtained by averaging AP values over all object classes.

Beyond the mAP, nuScenes includes additional metrics to evaluate key attributes of correctly detected objects, using a fixed association threshold of 2 meters. These include translation, scale, orientation, velocity, and attribute classification errors. Each of these metrics targets a specific aspect of detection performance, such as localization accuracy or motion estimation. To provide a unified performance score, the nuScenes Detection Score (NDS) aggregates all these metrics into a single weighted average, offering a comprehensive view of system performance.

Nevertheless, the use case addressed in this work diverges significantly from the standard nuScenes benchmark. Since 3D detections are performed from monocular images via clustering, the resulting cuboids inherently represent only the visible portion of objects from the camera's perspective. This makes centroid-based 3D box evaluation less meaningful due to an expected displacement between prediction and ground truth. Additionally, the camera-aligned nature of these cuboids makes the orientation error metric impractical. Consequently, the evaluation strategy of this 3D detections is focused on the precision of the estimated cuboid dimensions and positions, which directly assesses the pipeline's ability to achieve reasonable detection accuracy.

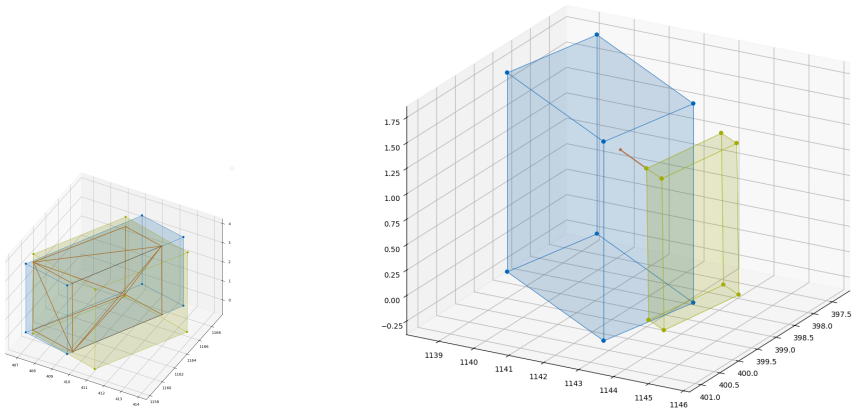
In this scenario, employing centroid distance as the criterion for evaluating position predictions with ground truth poses a significant challenge. Therefore, a more suitable metric, the volume-to-volume distance (v2v), has been adopted. This metric is defined as the minimum distance between the convex hulls of the considered volumes (Figure 16b).

However, the v2v metric has a notable limitation: it yields a value of zero whenever there is any intersection between the volumes, regardless of the degree of overlap. To overcome this, we utilize the Bounding Box Disparity (BBD) metric [47], which combines IoU and v2v into a single, continuous, and non-negative measure. This metric allows for the quantification of dissimilarity between two cuboids, whether they overlap or not. As stated in its original definition, while IoU can only compare intersecting cuboids, BBD provides a disparity measure even in the absence of overlap. Its interpretation is as follows:

$$\text{BBD} = 1 - \text{IoU} + v2v \tag{5}$$

- If $BBD = 0$, the cuboids are identical (total intersection).
- If $0 < BBD < 1$, there is overlap but also disparity.
- If $BBD \geq 1$, the cuboids do not overlap.

These distinct cases are illustrated in Figure 16.



(a) (b)

Figure 16: Bounding Box Disparity scene calculation. (a) shows an example of intersection, while (b) shows an example of volumetric distance

On the other hand, to evaluate the error in the estimation of the detected object's dimensions, the absolute difference between the dimensions of the predicted and

ground truth cuboids along each axis (length, width, and height) has been considered. This allows for the analysis of which dimension exhibits the greatest error and under what conditions.

The evaluation process begins by defining the spatial region in which the detections produced by the pipeline will be considered. Since the system is capable of generating predictions at distances greater than those actually used for producing the BEV occupancy and occlusion masks, the evaluation range is restricted to the same area used for generating those masks. Specifically, this involves projecting the camera's field of view onto the ground plane and limiting the maximum detection distance to 15 meters. As a result, only the predictions and ground truth elements that fall within this projected polygon at a given time frame are included in the evaluation.

Explain how the camera FOV is obtained from camera's intrinsics and how its converted to a polygon.

Then, the matching between predictions and ground truth is performed, formalized as a Linear Sum Assignment Problem (LSAP) and solved using the Hungarian algorithm. Because of this, the cost matrix is initially calculated as the v2v distance between all ground truth and prediction instances. However, unlike a standard LSAP, this implementation incorporates a maximum distance threshold for valid assignments. Detections with no sufficiently close ground truth counterparts are removed before solving the LSAP, and the cost matrix is padded to a square shape. Finally, assignments exceeding the distance threshold are discarded after the LSAP solution, ensuring only meaningful matches are retained. Maybe it is interesting to add an Appendix about this

Finally, the following metrics are calculated:

- **TP/FP/FN**: Computed from prediction-ground truth associations.
- **DD**: Volume difference between associated cuboids.
- **DED**: Absolute difference per axis (length, width, height).
- **v2v**: Minimum distance between convex hulls.
- **IoU**: 3D intersection-over-union.
- **BBD**: Dissimilarity metric for overlapping/non-overlapping boxes.

3.2.5.2 BEV masks evaluation

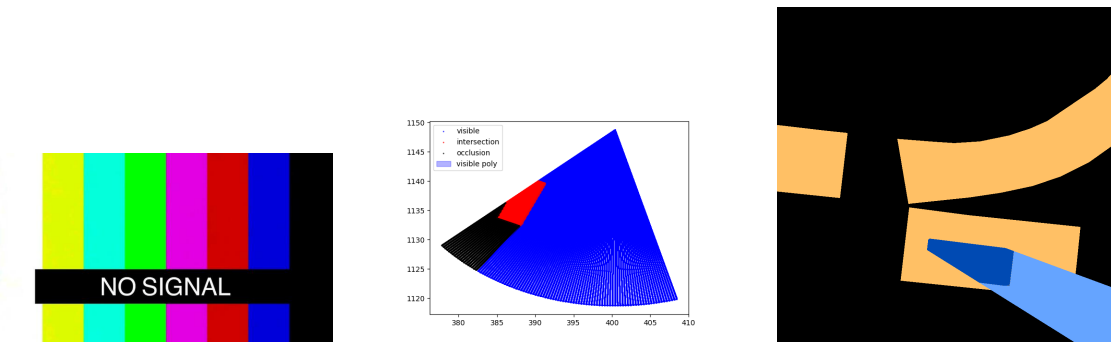
To evaluate the occupancy, occlusion, and drivable area masks, it is necessary to generate corresponding ground truth masks. These masks are computed from the

OpenLABEL annotated vehicular scene data as follows. For each frame, only the elements visible to the camera are considered (as detailed in Section 3.2.5.1) and a custom algorithm is applied to process this visible information.

This algorithm calculates camera viewing angles along the $\hat{x}_c$ axis to define N rays. A step length h is also defined, which determines the intervals at which each ray will sample points to classify them as visible, occluded, or belonging to an occupied area. All rays originate from the camera coordinate frame and are subsequently transformed to the vehicle coordinate frame (*vehicle-iso8855*) as it is the coordinate system in which the BEV masks are generated (see Section3.1.2).

Once these rays and their corresponding check points are established in the BEV mask coordinate frame, the point classification process begins. An iteration process goes through camera-visible objects (also transformed to the vehicle coordinate frame) and check if the base of the object's bounding box contains each ray's point. Here, three outcomes are possible: no intersection, indicating a visible point; intersection, classifying the point as occupied; or no current intersection but a previous intersection along the same ray, resulting in occlusion classification.

Following ray point evaluation, closed polygon sets are constructed to represent all points within each of the three classes. A clustering algorithm groups points of the same class to distinguish connected areas and generate individual sub-polygons. These sub-polygons are formed using only the edge points of each cluster and finally, the ground truth drivable area, which is directly annotated as polygons, undergoes a simpler process of identifying camera-visible polygons. All polygon sets are then rasterized into semantic masks, adhering to the used BEV parameters and considering the semantic classes described in Table 3. This whole process is represented in Figure 17.



(a) (b) (c)

Figure 17: BEV ground truth masks: (a) WebLABEL visualization, (b) resulting polygons, (c) rasterized and colored BEV mask.

4 Experiments and results

This section includes all experiments carried out for evaluating the difference between the two approaches considered in the BEV2Seg_2 pipeline, experiments to study what is the influence of extrinsic parameters modification as data augmentation technique for semantic segmentation of BEV images and the final evaluation of the proposed annotation pipeline of occupancy, occlusion and driveable areas.

4.1 BEV2Seg_2

Both approaches, segmenting-then-IPM and IPM-then-segmenting, were trained under the same conditions. During fine-tuning for the segmentation task, encoder layers were left unfrozen, allowing the entire model to adapt to the training data. Input images were preprocessed using the *SegformerImageProcessor*, which includes resizing to 512×512 pixels, rescaling pixel values by a factor of $1/255$, and normalizing with ImageNet mean and standard deviation values [48]. This preprocessing step ensures the input format is consistent with what the pretrained encoders expect.

Semantic masks were provided during training with the `reduce_labels` parameter set to `False`, as the dataset includes a "background" class. This configuration ensures that all pixel classes, including background, contribute to gradient computation during optimization. All experiments were executed on the hardware setup described in Table 4. From the six available Segformer encoder variants, three were selected for evaluation: MiT-b0, MiT-b2, and MiT-b4. Due to its larger size, the MiT-b4 model required two gradient accumulation steps to fit within the GPU's memory constraints. Checkpoints were saved based on evaluation loss, a design decision made despite ongoing discussions about whether loss or mean intersection over union is the better metric for model selection. A basic linear learning rate scheduler was applied throughout training and the loss function employed was `BCEWithLogitsLoss`, which is well-suited for multi-label semantic segmentation tasks, as it combines sigmoid activation and binary cross-entropy in a numerically stable way following the Segformer implementation.

Regarding to the used notation, the models that follow the segment-then-IPM pipeline to obtain BEV semantic masks are referred to as `raw2seg_bev`, while those that first apply IPM and then perform segmentation are named `raw2bev_seg`.

The two approaches were firstly trained using the smallest Segformer model variant, MiT-b0, for $20.7K$ steps without applying any regularization technique. This initial experiment was performed to observe whether the models were able to learn and predict on the dataset and see if them suffered from overfitting. Also, for this

Component	Specifications	Num workers
CPU	Intel Xeon Gold 6230 (80) @ 3.900GHz	8
GPU	NVIDIA Tesla V100-SXM2-32GB	2
Memmory	772643MiB	-
OS	Ubuntu 22.04.3 LTS x86_64	-

Table 4: Hardware used for experiments

purpose, the choice of MiT-b0 was intentional as it trains faster and, due to its limited capacity, is less prone to extreme overfitting compared to larger models. This made it a suitable candidate for testing different hyperparameter configurations in a lightweight environment.

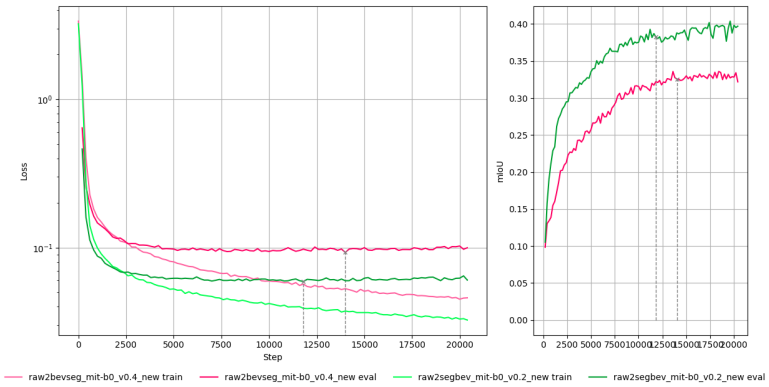


Figure 18: Training and evaluation loss of raw2seg_bev and raw2bev_seg MiT-b0 models without any regularization technique

As shown in Figure 18, both models showed clear signs of overfitting. While the training loss continued to decrease continuously, the validation loss began to increase, indicating a lack of generalization and failure to converge. These results highlight the importance of introducing regularization techniques even for small model sizes. Additionally, no signs of exploding gradients were observed during the training of these models.

Two main approaches were selected to tackle the overfitting problem: weight decay (also known as L2 regularization) and data augmentation. Weight decay penalizes large weights during training, and makes the model more robust and less prone to memorizing irrelevant details; while data augmentation techniques introduces variability in the training dataset enabling the model to adapt better to unseen data. However, the introduction of data augmentation techniques into BEV images domain is not trivial and raises another research question.

It is also important to highlight the difference in metrics between the two models. Even though the backbone and training hyperparameters remain the same, there is a noticeable difference in both the loss values and the mIoU between the models. This difference is primarily due to the fact that the evaluation datasets differ. Specifically, the approach that segments the image first and then reprojects it is evaluated using semantic masks in the camera view, while the approach that first reprojects images into BEV and then performs segmentation is evaluated using masks directly in the BEV domain.

This distinction motivates the analysis shown in Figure 19, where the model `raw2segbev_mit-b0_v0.3` is evaluated on the test set twice: once using semantic masks in the camera domain and once using BEV masks. The figure displays the per-class mIoU for all the semantic categories the model was trained on, and two main observations emerge.

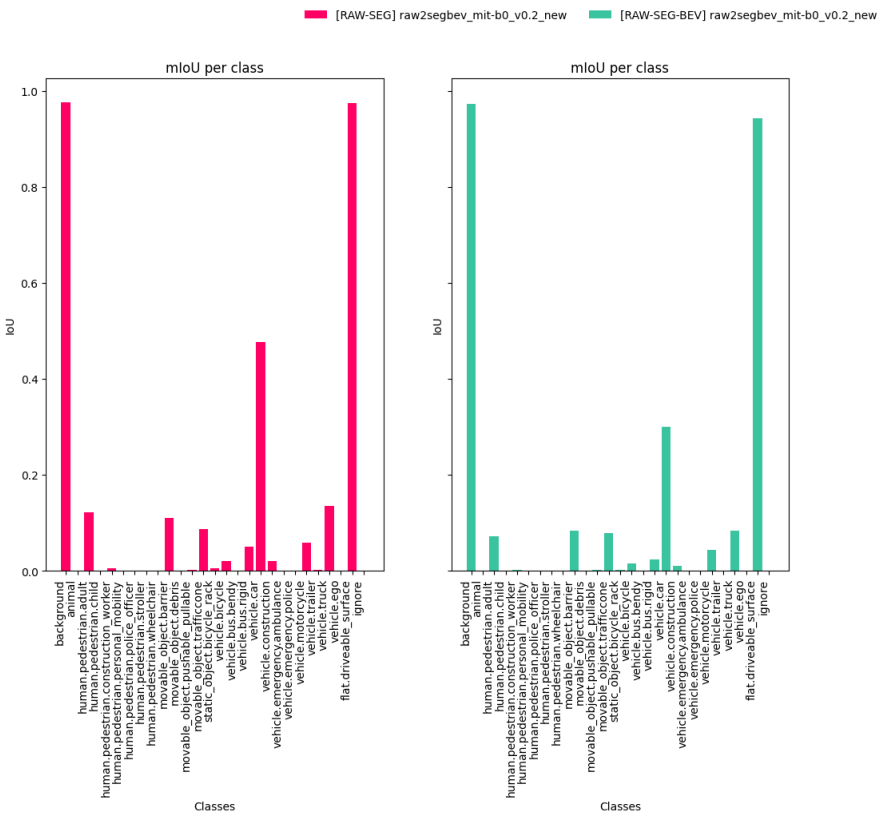


Figure 19: Model evaluated with normal and BEV images

First, there's a clear imbalance in performance between dominant classes such as `background` and `flat.driveable_area`, and the remaining, less frequent classes. Many of the minority classes show extremely low or even zero IoU scores. This is

partly due to the inherent class imbalance in the dataset: some semantic classes have no annotated pixels at all (see Appendix A for more details).

Second, all metrics are significantly higher when the model is evaluated using standard camera-domain masks. Reprojecting the masks to BEV introduces several effects: (1) a fixed background area appears in the BEV space that lies outside the camera's field of view; (2) the maximum scene distance is reduced to 15 meters (see reprojection parameters in Section 3.1), which limits the presence of distant objects: minority classes like pedestrians are often excluded unless they are very close to the vehicle, while driveable areas and vehicles dominate the scene; and (3) the resolution is reduced to 1024×1024 pixels.

All these factors lead to a lower total number of pixels in the BEV dataset. As a result, each misclassified pixel has a greater impact on the overall evaluation, making errors more costly and consequently lowering the average performance metrics.

To address this, many training methods use class weighting in the loss function to penalize misclassifications of minority classes more heavily than those of majority classes. However, in this particular project, it is not essential to maintain all class distinctions provided by the dataset. Therefore, we investigate whether merging semantic classes can help mitigate the effects of class imbalance.

With all of this in place, the training strategy is designed to support experimentation in order to answer three main research questions:

- *Does the label's merging strategy helps increasing the model's performance on the low presence classes?*
- *Which data augmentation technique is more effective for trainig a model directly on BEV images on a semantic segmentation task?*
- *Which of the two approaches performs better for BEV driveable area segmentation?*

Mention the used hyperparameters for training the models.

4.1.1 Merging labels

As previously discussed, to improve model performance on the dataset's minority classes, a semantic label merging strategy was applied. This approach combines less represented semantic classes into more general, representative categories. Notably, this same merging strategy was already used in the annotation pipeline, so it was incorporated during training using the same look-up table of merging rules

described in Table 3. The training and evaluation scripts were adapted accordingly to apply the merging rules consistently.

Table 5 shows the results of two models trained with camera-perspective images and evaluated on the same test set using BEV masks. Model A was trained using the full set of original semantic classes, while Model B was trained using the merged classes, thereby reducing the total number of labels.

In general, the results indicate that both models achieve comparable performance on the majority classes, specifically `background` and `flat.driveable_surface`. However, a significant difference is observed in the minority classes, where Model B outperforms Model A across all evaluation metrics. This suggests that simplifying the semantic space does not decrease the model's ability to identify important instances and, instead, it reduces the penalty from underrepresented classes. Additionally, by reducing the number of classes, the overall average of the metrics improves as it avoids dilution from irrelevant or sparsely populated classes. This makes label merging especially useful in cases where fine-grained segmentation isn't necessary and consistency across the whole model is more important.

Merged Class	**Original Label**	**mIoU A**	**mIoU B**	**mF1 A**	**mF1 B**
background	background	0.97	0.97	0.99	0.99
animal	animal	0.00	0.00	0.00	0.00
human.pedestrian.adult	human.pedestrian.adult	0.07	0.08	0.09	0.11
	human.pedestrian.child	0.00	-	0.00	-
	human.pedestrian.construction_worker	0.00	-	0.00	-
	human.pedestrian.personal_mobility	0.00	-	0.00	-
	human.pedestrian.police_officer	0.00	-	0.00	-
	human.pedestrian.stroller	0.00	-	0.00	-
	human.pedestrian.wheelchair	0.00	-	0.00	-
movable_object.barrier	movable_object.barrier	0.08	0.14	0.09	0.17
	movable_object.debris	0.00	-	0.00	-
	movable_object.pushable_pullable	0.00	-	0.00	-
	movable_object.trafficcone	0.08	-	0.10	-
	static_object.bicycle_rack	0.00	-	0.00	-
vehicle.car	vehicle.bus.bendy	0.00	-	0.00	-
	vehicle.bus.rigid	0.02	-	0.03	-
	vehicle.car	0.30	0.38	0.33	0.42
	vehicle.construction	0.01	-	0.01	-
	vehicle.emergency.ambulance	0.00	-	0.00	-
	vehicle.emergency.police	0.00	-	0.00	-
	vehicle.trailer	0.00	-	0.00	-
	vehicle.truck	0.08	-	0.09	-
vehicle.ego	vehicle.ego	0.00	0.00	0.00	0.00
vehicle.motorcycle	vehicle.bicycle	0.02	-	0.02	-
	vehicle.motorcycle	0.04	0.06	0.05	0.07
flat.driveable_surface	flat.driveable_surface	0.94	0.94	0.97	0.97
		0.10	0.32	0.11	0.34

Table 5: Per-class metric comparison between Model A and merged Model B evaluated with BEV images

4.1.2 Data augmentation

Data augmentations are commonly used in deep learning models to mitigate overfitting during training and improve model generalization. There exists multiple types of data augmentation on the image domain: from pixel-based transformations, such as color space modifications, histogram equalization or filtering operations; to geometric transformations, including translations, rotations, shearings and homographies. These techniques have been widely applied in computer vision tasks and have shown to enhance model performance. However, performing data augmentation in BEV is not an easy task, as IPM images are already homographies of camera images, resulting in inherent distorsions.

Filtering operations can be applied to both standard and BEV images but geometric transformations were selected as the primary data augmentation method for camera domain images following the strategies employed in training the SegFormer model [30]. Accordingly, random resizing, random cropping, and horizontal flipping were chosen as augmentation operations for perspective images.

Regarding BEV data augmentations, some multi-view methods implement strategies such as random flipping and random scaling, while others operate in the frequency domain [41]. However, these approaches apply augmentations to perspective images before the BEV transformation. Performing random cropping on a BEV image may lead to significant information loss, as large portions of the image may consist of unlabeled background data, potentially resulting in crops with insufficient information for effective training (Figure 20).

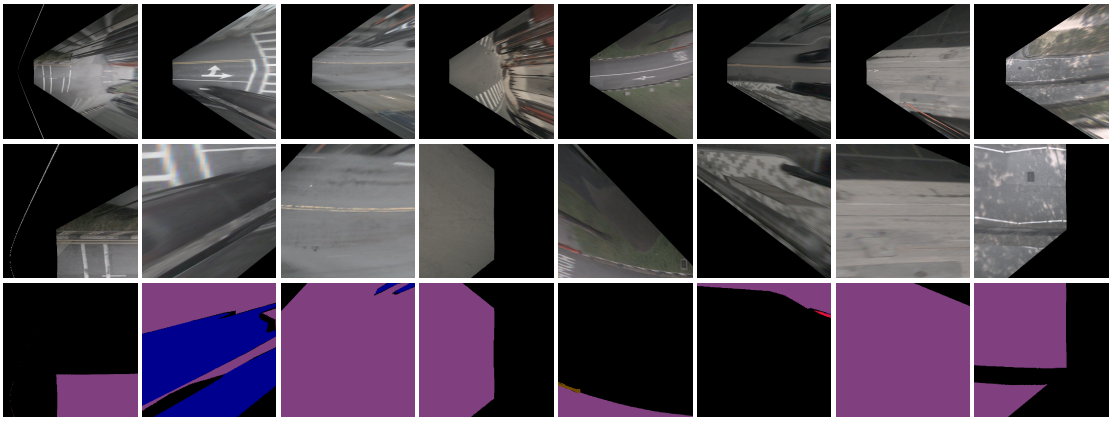


Figure 20: Random cropping and horizontal flipping on BEV images. Original BEV images on first row; random flipped and cropped images on second row and corresponding semantic masks on last row.

In this context, a different approach was also considered: applying geometric trans-

formations by modifying the camera's extrinsic parameters before reprojecting to BEV space. The objective is to introduce random transformations along one of the camera's rotation axes, generating diverse BEV reprojections with varying degrees of distortion. This technique may enable the model to adapt to different extrinsic camera configurations, improving its robustness to variations in camera placement and orientation (Figure 21).

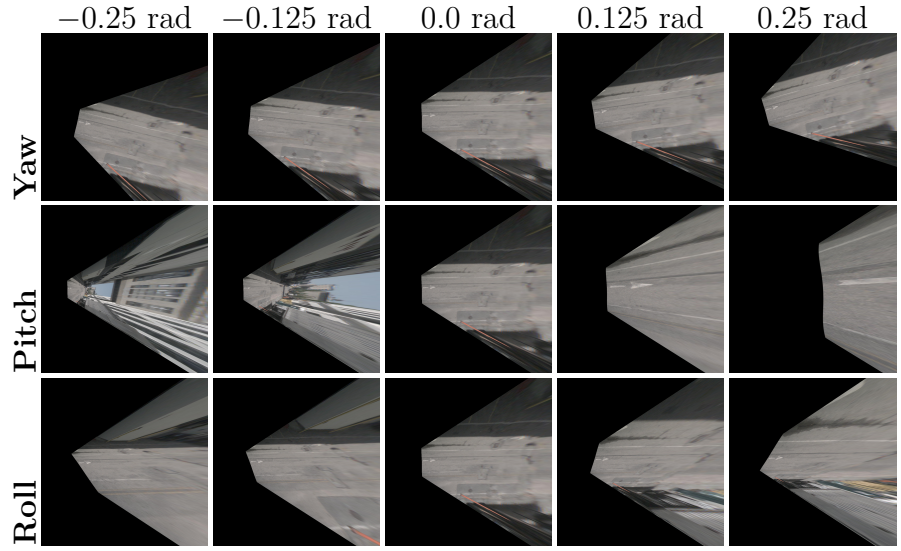


Figure 21: Effect of camera transformations on BEV projection. The first row shows variations in the yaw axes, the second in pitch, and the third in roll.

This section has a dual purpose. First, the effect of applying geometric transformations to camera-perspective images as data augmetation strategy for training a traditional segmentation model in order to mitigate overfitting is studied. Second, the effectiveness of two described techniques for performing data augmentation on BEV images is explored: applying the same geometric augmentations used for regular images, or modifying the camera's extrinsic parameters before performing the projections.

Therefore, the introduction of data augmentation strategies such as random cropping, horizontal flipping, and rescaling on regular images significantly reduces overfitting, as shown in Figure 22. This reduction in overfitting allows for longer training periods and improves model performance on the evaluation set, leading to convergence at higher mIoU values. This effect is also reflected in the evaluation of the models on the test set, where the model with data augmentation presents higher mIoU than the model affected by overfitting.

Secondly, the Figure 23 shows the trainig process graph for the three models

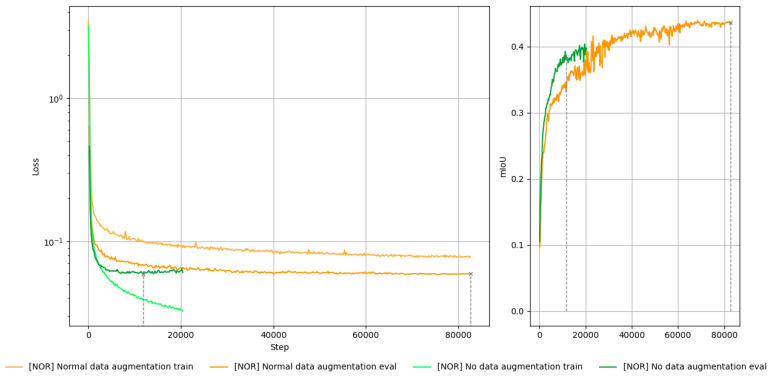


Figure 22: Before and after regularization techniques on models trained with camera-domain images

trained with BEV images and reveals some significant differences. The first model, without any data augmentation, exhibits notable overfitting. In contrast, both data augmentation strategies, the traditional geometric augmentations (random flipping, random cropping, and rescaling) and the custom strategy of modifying the camera's extrinsic parameters, seems to partially reduce this overfitting.

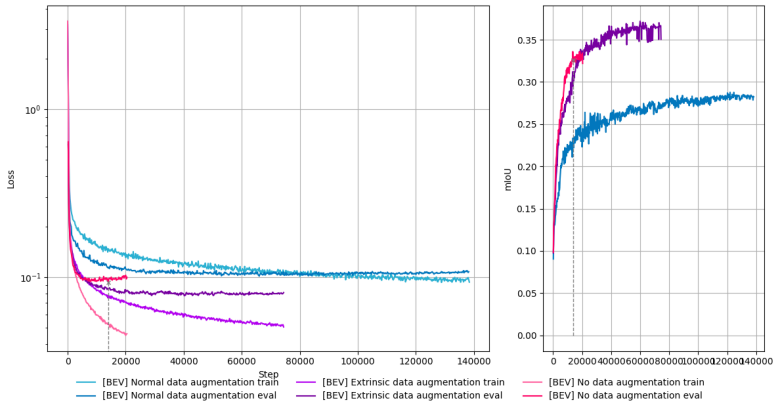


Figure 23: Training logs of models with BEV data augmentation techniques

Focusing on the model employing traditional data augmentation techniques on the BEV images, the training error is generally higher, and the training and validation error curves remain closely aligned with a considerably lower mIoU on the validation set compared to the other two models. This pattern suggests potential underfitting. This observation appears to support the initial hypothesis that directly applying traditional image augmentations to BEV representations might not provide the model with sufficiently informative variations for effective training in this specific domain.

On the other hand, the model trained with the extrinsic's data augmentation strategy shows less overfitting than the model without any data augmentation technique. This allows for more extended training and achieves superior results on the validation set, outperforming the overfitted model.

Overall, the findings from the validation set suggest that the custom data augmentation techniques for BEV images offer better performance than directly applying traditional augmentations. However, it is crucial to examine the models' performance on the unseen test set. Figure 24 illustrates this evaluation, revealing that the performance difference between the three models is not as substantial in reality. While the model trained with extrinsic parameter modifications shows slightly better results, the performance gap between the other two models is less clear, with the model using traditional data augmentation on BEV images occasionally surpassing the overfitted model in vehicle-related classes.

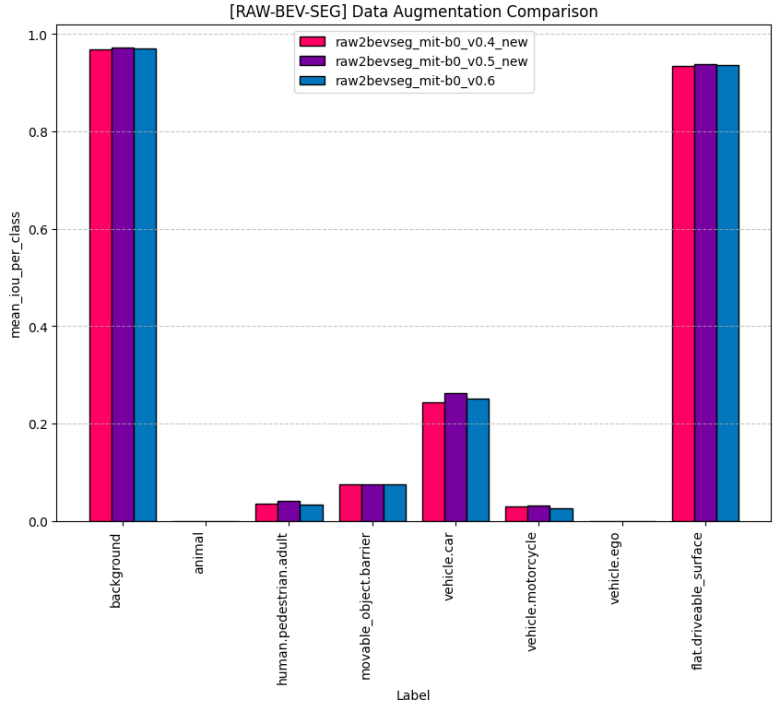


Figure 24: Different data augmentation stategies evaluated with BEV images.

4.1.3 raw2seg_bev vs raw2bev_seg

raw2seg_bev results compared with raw2bev_seg results.

4.2 Annotation evaluation

Explain the selected scene, the two experiments carried out and their results.

Acknowledgements

Lore ipsum...

References

[1] Ekim Yurtsever et al. "A Survey of Autonomous Driving: Common Practices and Emerging Technologies". In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149.

[2] Enrique Marti et al. "A Review of Sensor Technologies for Perception in Automated Driving". In: *IEEE Intelligent Transportation Systems Magazine* 11.4 (2019), pp. 94–108. DOI: 10.1109/MITS.2019.2907630.

[3] Stephanie Grubmüller, Jiri Plihal, and Pavel Nedoma. "Automated Driving from the View of Technical Standards". In: *Automated Driving: Safer and More Efficient Future Driving*. Ed. by Daniel Watzenig and Martin Horn. Cham: Springer International Publishing, 2017, pp. 29–40. ISBN: 978-3-319-31895-0. DOI: 10.1007/978-3-319-31895-0_3. URL: https://doi.org/10.1007/978-3-319-31895-0_3.

[4] Jing Yu and Feng Luo. "Fallback Strategy for Level 4+ Automated Driving System". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 156–162. DOI: 10.1109/ITSC.2019.8917404.

[5] Ardi Tampuu et al. "A Survey of End-to-End Driving: Architectures and Training Methods". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.4 (2022), pp. 1364–1384. DOI: 10.1109/TNNLS.2020.3043505.

[6] Scott Drew Pendleton et al. "Perception, Planning, Control, and Coordination for Autonomous Vehicles". In: *Machines* 5.1 (2017). ISSN: 2075-1702. DOI: 10.3390/machines5010006. URL: https://www.mdpi.com/2075-1702/5/1/6.

[7] Ömer Şahin Taş et al. "Functional system architectures towards fully automated driving". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016, pp. 304–309. DOI: 10.1109/IVS.2016.7535402.

[8] Jiageng Mao et al. *3D Object Detection for Autonomous Driving: A Comprehensive Survey*. 2023. arXiv: 2206.09474 [cs.CV]. URL: https://arxiv.org/abs/2206.09474.

[9] Yuexin Ma et al. "Vision-Centric BEV Perception: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.12 (2024), pp. 10978–10997. DOI: 10.1109/TPAMI.2024.3449912.

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

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by

Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[12] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[13] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[14] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2016. arXiv: 1511.07122 [cs.CV]. URL: https://arxiv.org/abs/1511.07122.

[15] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[16] Sixiao Zheng et al. "Rethinking Semantic Segmentation From a Sequence-to-Sequence Perspective With Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 6881–6890.

[17] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV]. URL: https://arxiv.org/abs/2103.14030.

[18] Andreas Geiger et al. "3D Traffic Scene Understanding From Movable Platforms". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.5 (2014), pp. 1012–1025. DOI: 10.1109/TPAMI.2013.185.

[19] Chien-Chuan Lin and Ming-Shi Wang. "A Vision Based Top-View Transformation Model for a Vehicle Parking Assistant". In: *Sensors* 12.4 (2012), pp. 4431–4446. ISSN: 1424-8220. DOI: 10.3390/s120404431. URL: https://www.mdpi.com/1424-8220/12/4/4431.

[20] Ammar Abbas and Andrew Zisserman. *A Geometric Approach to Obtain a Bird's Eye View from an Image*. 2020. arXiv: 1905.02231 [cs.CV]. URL: https://arxiv.org/abs/1905.02231.

[21] L. Reiher, B. Lampe, and L. Eckstein. "A Sim2Real Deep Learning Approach for the Transformation of Images from Multiple Vehicle-Mounted Cameras to a Semantically Segmented Image in Bird's Eye View". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020. DOI: 10.1109/ITSC45102.2020.9294462.

[22] Qi Li et al. *HDMapNet: An Online HD Map Construction and Evaluation Framework*. 2022. arXiv: 2107.06307 [cs.CV]. URL: https://arxiv.org/abs/2107.06307.

[23] Weixiang Yang et al. "Projecting Your View Attentively: Monocular Road Scene Layout Estimation via Cross-View Transformation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15536–15545.

[24] Bowen Pan et al. "Cross-View Semantic Segmentation for Sensing Surroundings". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4867–4873. DOI: 10.1109/LRA.2020.3004325.

[25] Jonah Philion and Sanja Fidler. "Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 194–210. ISBN: 978-3-030-58568-6.

[26] Enze Xie et al. "Mˆ 2BEV: Multi-Camera Joint 3D Detection and Segmentation with Unified Birds-Eye View Representation". In: *arXiv preprint arXiv:2204.05088* (2022).

[27] Kaustubh Mani et al. "MonoLayout: Amodal scene layout from a single image". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020.

[28] Zhiqi Li et al. *BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers*. 2022. arXiv: 2203.17270 [cs.CV]. URL: https://arxiv.org/abs/2203.17270.

[29] Noureldin Hendy et al. *FISHING Net: Future Inference of Semantic Heatmaps In Grids*. 2020. arXiv: 2006.09917 [cs.CV]. URL: https://arxiv.org/abs/2006.09917.

[30] Enze Xie et al. "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers". In: *CoRR* abs/2105.15203 (2021). arXiv: 2105.15203. URL: https://arxiv.org/abs/2105.15203.

[31] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[32] Yuanduo Hong et al. *Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes*. 2021. arXiv: 2101.06085 [cs.CV]. URL: https://arxiv.org/abs/2101.06085.

[33] Jiacong Xu, Zixiang Xiong, and Shankar P. Bhattacharyya. *PIDNet: A Real-time Semantic Segmentation Network Inspired by PID Controllers*. 2023. arXiv: 2206.02066 [cs.CV]. URL: https://arxiv.org/abs/2206.02066.

[34] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. arXiv: 1604.01685 [cs.CV]. URL: https://arxiv.org/abs/1604.01685.

[35] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[36] Xinyu Huang et al. "The ApolloScape Open Dataset for Autonomous Driving and Its Application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (Oct. 2020), pp. 2702–2719. ISSN: 1939-3539. DOI: 10.1109/tpami.2019.2926463. URL: http://dx.doi.org/10.1109/TPAMI.2019.2926463.

[37] Holger Caesar et al. "nuScenes: A Multimodal Dataset for Autonomous Driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[38] Oliver Zendel et al. "Unifying Panoptic Segmentation for Autonomous Driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 21351–21360.

[39] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database". In: *Pattern Recognition Letters* 30.2 (2009). Video-based Object and Event Analysis, pp. 88–97. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2008.04.005. URL: https://www.sciencedirect.com/science/article/pii/S0167865508001220.

[40] Marcos Nieto, Orti Senderos, and Oihana Otaegui. "Boosting AI applications: Labeling format for complex datasets". In: *SoftwareX* 13 (2021), p. 100653. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2020.100653. URL: https://www.sciencedirect.com/science/article/pii/S2352711020303666.

[41] Calvin Glisson and Qiuxiao Chen. "HSDA: High-frequency Shuffle Data Augmentation for Bird's-Eye-View Map Segmentation". In: *arXiv preprint arXiv:2412.06127* (2024).

[42] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[43] Itziar Urbieta et al. "Correction to: WebLabel: OpenLABEL-compliant multi-sensor labelling". In: *Multimedia Tools and Applications* 83.9 (2024), pp. 26525–26525. ISSN: 1573-7721. DOI: 10.1007/s11042-023-17602-0. URL: https://doi.org/10.1007/s11042-023-17602-0.

[44] Yang Liu et al. "A Survey of Depth Estimation Based on Computer Vision". In: *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*. 2020, pp. 135–141. DOI: 10.1109/DSC50466.2020.00028.

[45] Aleksei Bochkovskii et al. *Depth Pro: Sharp Monocular Metric Depth in Less Than a Second*. 2024. URL: https://arxiv.org/abs/2410.02073.

[46] Gregory G Slabaugh. "Computing Euler angles from a rotation matrix". In: *Retrieved on August* 6.2000 (1999), pp. 39–63.

[47] Michael G. Adam et al. "Bounding Box Disparity: 3D Metrics for Object Detection with Full Degree of Freedom". In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 1491–1495. DOI: 10.1109/icip46576.2022.9897588. URL: http://dx.doi.org/10.1109/ICIP46576.2022.9897588.

[48] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Appendices

A OLDatasets

Standing for OpenLABEL Datasets, OLDatasets is a custom parser from NuImages format to OpenLABEL format developed in order to integrate it with the VCD library ecosystem developed and maintained by Vicomtech [2].

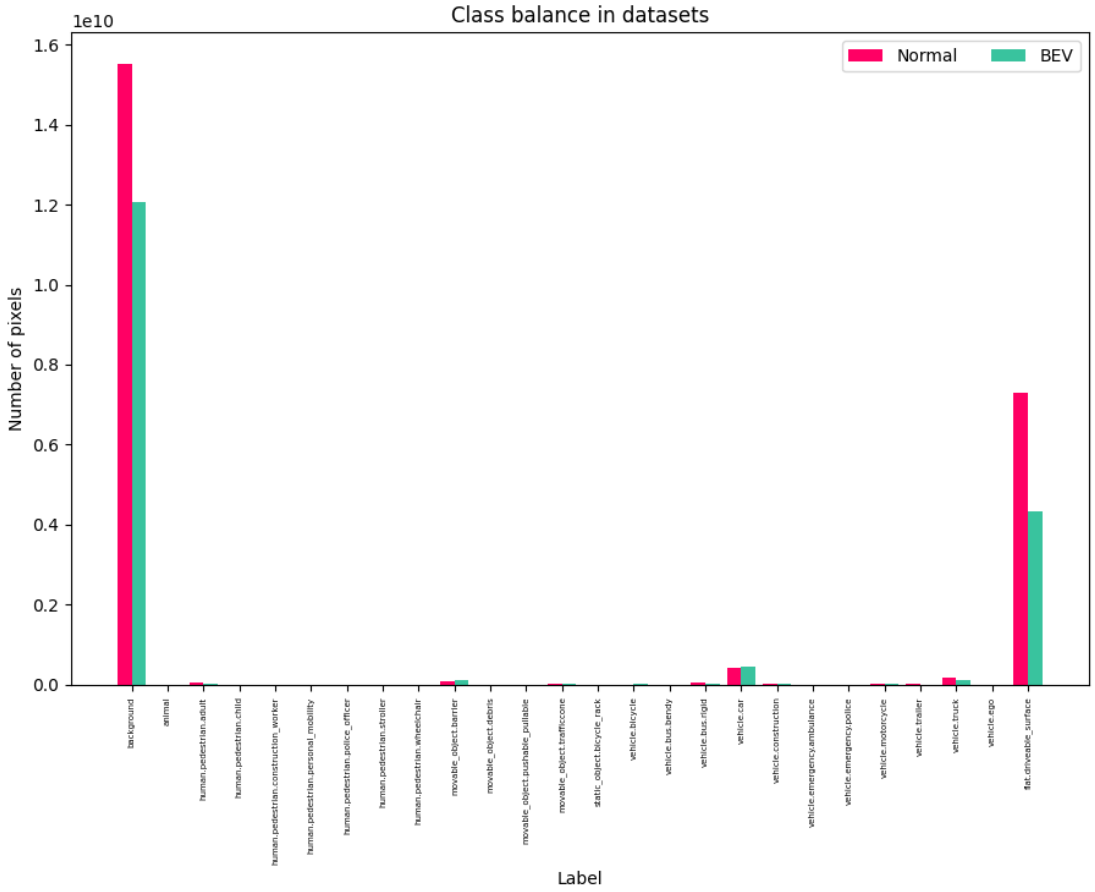


Figure 25: Number of pixels per class in datasets

[2] https://www.vicomtech.org/en/

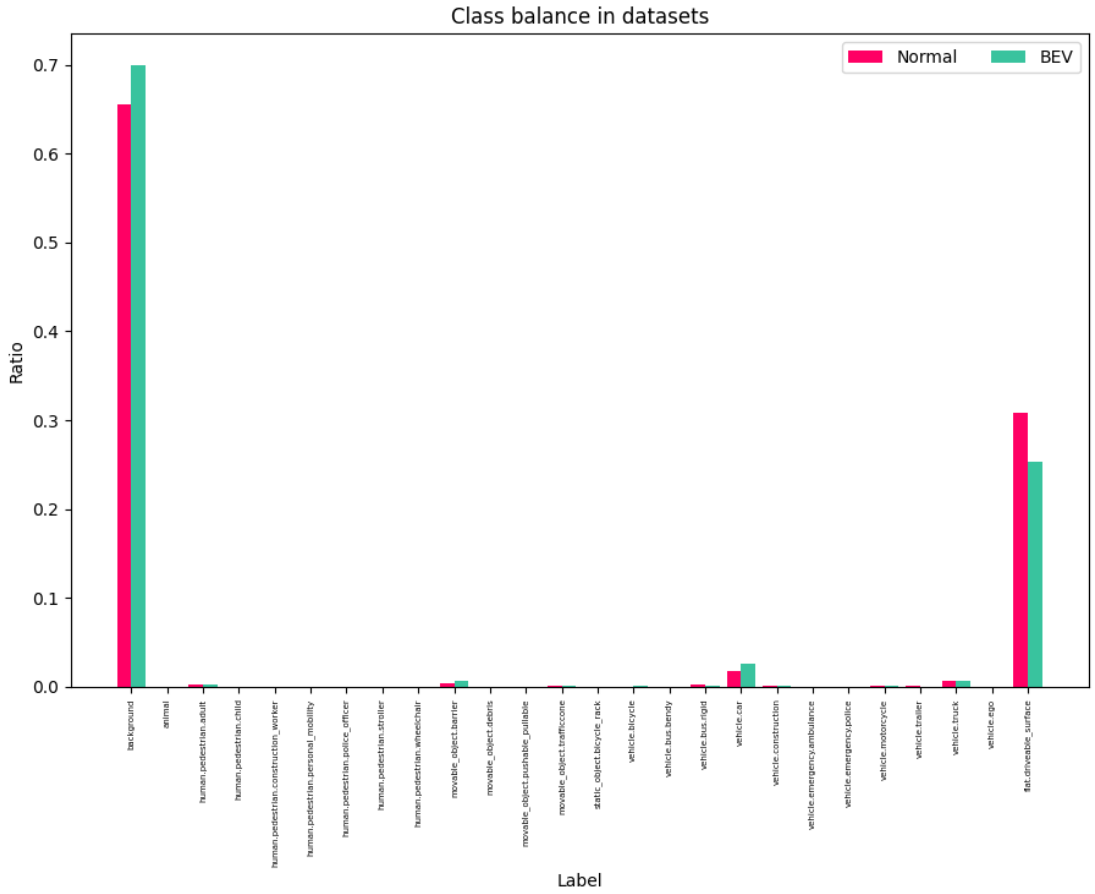


Figure 26: Relative distribution of classes in dataset

B F1-Score in terms of Intersection Over Union

$$\begin{aligned} \text{F1} &= \frac{2TP}{2TP + FP + FN} \\ \text{IoU} &= \frac{TP}{TP + FP + FN} \end{aligned} \tag{6}$$

We know that:

$$\text{IoU} = \frac{TP}{TP + FP + FN} \Rightarrow TP + FP + FN = \frac{TP}{\text{IoU}}$$

$$\begin{aligned} \mathrm{F1} &= \frac{2TP}{2TP+FP+FN} = \frac{2TP}{TP+(TP+FP+FN)} = \frac{2TP}{TP+\frac{TP}{\mathrm{IoU}}} \\ &= \frac{2TP}{TP\left(1+\frac{1}{\mathrm{IoU}}\right)} = \frac{2}{1+\frac{1}{\mathrm{IoU}}} = \frac{2\mathrm{IoU}}{1+\mathrm{IoU}} \end{aligned} \tag{7}$$

Now solve for IoU in terms of F1:

$$\mathrm{F1} = \frac{2\mathrm{IoU}}{1+\mathrm{IoU}}$$

$$\mathrm{F1}(1+\mathrm{IoU}) = 2\mathrm{IoU} \Rightarrow \mathrm{F1} + \mathrm{F1}\cdot\mathrm{IoU} = 2\cdot\mathrm{IoU}$$

$$\mathrm{F1} = 2\mathrm{IoU} - \mathrm{F1}\cdot\mathrm{IoU} \Rightarrow \mathrm{F1} = \mathrm{IoU}(2-\mathrm{F1})$$

$$\mathrm{IoU} = \frac{\mathrm{F1}}{2-\mathrm{F1}}$$

$$\boxed{\mathrm{F1} = \frac{2\mathrm{IoU}}{1+\mathrm{IoU}} \qquad \mathrm{IoU} = \frac{\mathrm{F1}}{2-\mathrm{F1}}} \tag{8}$$

C Merging comparison

D Color palette

Name	ID	trainId	Dynamic	Color (RGB)
background	0	0	False	(0, 0, 0)
animal	1	1	True	(255, 0, 0)
human.pedestrian.adult	2	2	True	(220, 20, 60)
human.pedestrian.child	3	3	True	(220, 20, 60)
human.pedestrian.construction_worker	4	4	True	(220, 20, 60)
human.pedestrian.personal_mobility	5	5	True	(220, 20, 60)
human.pedestrian.police_officer	6	6	True	(220, 20, 60)
human.pedestrian.stroller	7	7	True	(220, 20, 60)
human.pedestrian.wheelchair	8	8	True	(220, 20, 60)
movable_object.barrier	9	9	False	(190, 153, 153)
movable_object.debris	10	10	False	(152, 251, 152)
movable_object.pushable_pullable	11	11	False	(255, 0, 0)
movable_object.trafficcone	12	12	True	(111, 74, 0)
static_object.bicycle_rack	13	13	False	(255, 0, 0)
vehicle.bicycle	14	14	True	(119, 11, 32)
vehicle.bus.bendy	15	15	True	(0, 60, 100)
vehicle.bus.rigid	16	16	True	(0, 60, 100)
vehicle.car	17	17	True	(0, 0, 142)
vehicle.construction	18	18	True	(255, 0, 0)
vehicle.emergency.ambulance	19	19	True	(255, 0, 0)
vehicle.emergency.police	20	20	True	(255, 0, 0)
vehicle.motorcycle	21	21	True	(0, 0, 230)
vehicle.trailer	22	22	True	(0, 0, 110)
vehicle.truck	23	23	True	(0, 0, 70)
vehicle.ego	24	24	True	(255, 255, 255)
flat.driveable_surface	25	25	False	(128, 64, 128)
ignore	255	255		

Table 6: Semantic labels defined for NuImages masks

Merged Class	Original Label	mIoU A	mIoU B	mF1 A	mF1 B
background	background	0.98	0.98	0.99	0.99
animal	animal	0.00	0.00	0.00	0.00
human.pedestrian.adult	human.pedestrian.adult	0.12	0.14	0.17	0.19
	human.pedestrian.child	0.00	-	0.00	-
	human.pedestrian.construction_worker	0.01	-	0.01	-
	human.pedestrian.personal_mobility	0.00	-	0.00	-
	human.pedestrian.police_officer	0.00	-	0.00	-
	human.pedestrian.stroller	0.00	-	0.00	-
	human.pedestrian.wheelchair	0.00	-	0.00	-
movable_object.barrier	movable_object.barrier	0.11	0.17	0.13	0.21
	movable_object.debris	0.00	-	0.00	-
	movable_object.pushable_pullable	0.00	-	0.00	-
	movable_object.trafficcone	0.09	-	0.12	-
	static_object.bicycle_rack	0.01	-	0.01	-
vehicle.car	vehicle.bus.bendy	0.00	-	0.00	-
	vehicle.bus.rigid	0.05	-	0.06	-
	vehicle.car	0.48	0.58	0.56	0.66
	vehicle.construction	0.02	-	0.03	-
	vehicle.emergency.ambulance	0.00	-	0.00	-
	vehicle.emergency.police	0.00	-	0.00	-
	vehicle.trailer	0.00	-	0.00	-
	vehicle.truck	0.14	-	0.16	-
vehicle.ego	vehicle.ego	0.00	0.00	0.00	0.00
vehicle.motorcycle	vehicle.bicycle	0.02	-	0.03	-
	vehicle.motorcycle	0.06	0.07	0.07	0.09
flat.driveable_surface	flat.driveable_surface	0.97	0.97	0.99	0.99
		0.12	0.36	0.13	0.39

Table 7: Per-class metric comparison between Model A and merged Model B evaluated with normal images

Hex Code	Color Sample
#FF0064	
#FF62A0	
#0277BD	
#2AB0D2	
#7700A0	
#B200EF	
#00A032	
#03FF52	
#FF9800	
#FFB03B	
#C00071	
#FF37AD	
#00A697	
#44FFEE	

Table 8: Color Palette