



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: April 11, 2025

Abstract:

Keywords:

Contents

Acronyms	5
1 Introduction	6
2 State of the art	7
2.1 Semantic segmentation	9
2.2 BEV semantic segmentation	10
3 Methodology	12
3.1 Segmentation experiment design: BEV2Seg_2	12
3.1.1 Segformer	12
3.1.2 BEVDataset	13
3.1.3 Data augmentations	15
3.1.4 Validation and comparison: Selection module	17
3.2 Driveable area automatic annotation	17
3.2.1 Depth estimation	18
3.2.2 Scene PCD	19
3.2.3 Instance scene PCD	21
3.2.4 Instance BEV mask	25
3.3 Evaluation methodology	26
4 Experiments and results	27
4.1 BEV2Seg_2	27
4.1.1 BEV images data augmentation techniques comparison . .	29
4.1.2 Pipeline comparison	29
4.2 3D detections evaluations	30
4.3 BEV masks evaluation	30
References	34

List of Figures

1	Typical ADS modular architecture.	8
2	bev2seg_2 flow diagram.	12
3	Segformer architecture	14
4	BEVDataset structure and mini set samples	15
5	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.	16
6	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.	17
7	Annotation flow diagram.	18
8	Traditional depth estimation techniques: (a) stereo-depth, (b) structure from motion, (c) depth-from-defocus, (d) photometric-depth, and (e) shape from shading.	19
9	Depth maps of BEVDataset ‘mini’ estimated with Depth-Pro model.	19
10	Ideal perspective camera model.	20
11	Reconstructed pointcloud.	22
12	Instance scene computation diagram.	22
13	Colorized semantic pointcloud.	23
14	Oriented bounding box problems.	24
15	Resulting instances (a) in 3D and (b) projected into the front camera.	25
16	BEV occupancy and occlusion masks. (a) Input 3D detections; (b) input BEV semantic masks and (c) final BEV occupied-occluded mask.	26
17	Training and evaluation loss of raw2seg_beav and raw2bev_seg MiT-b0 models without any regularization technique	28
18	Before and after regularization techniques. Light colors before, darker colors after. BEV segmentation model uses normal data augmentations.	29

List of Tables

1	Comparison of different models. Results are obtained from [30] [31] [28].	13
2	Class merge dictionary used for semantic simplification.	23
3	Hardware used for experiments	27
4	Semantic labels defined for NuImages masks	35
5	Color Palette	36

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

LUT Lookup Table

AABB Axis-Aligned Bounding Box

1 Introduction

The development of Automated Driving Systems (ADS) have 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. 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 these 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 limitation, recent methods leverage data-driven techniques for BEV representation. [22] [23] [24]. 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 with semantic BEV images outperforms a typical segmentation model for planar 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.

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]. However, 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 exist, 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.

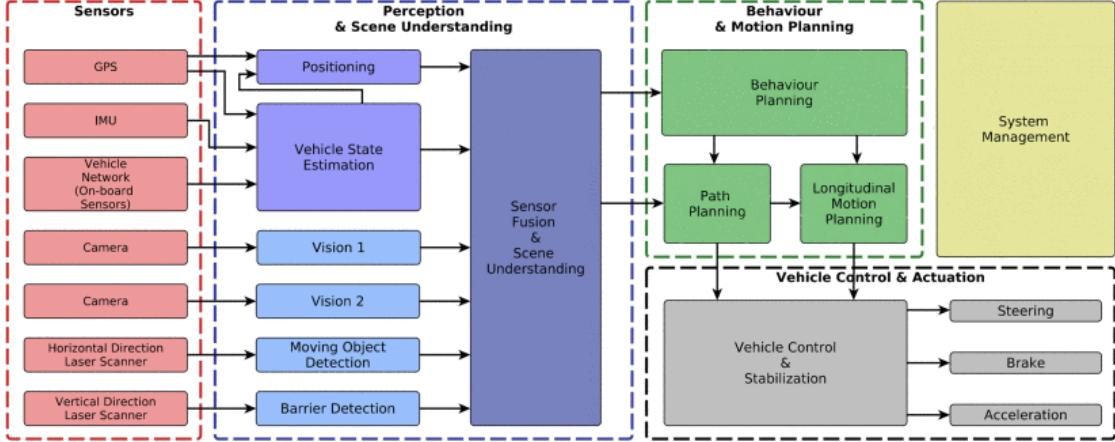


Figure 1: Typical ADS modular architecture.

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. [Add references](#).

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. [Add references](#).

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-established research topic in computer vision.

Early methods relied on hand-crafted features but the introduction of Fully Convolutional Networks (FCN) [8] boosted semantic segmentation as convolutional layers in FCN enables a data-driven feature extraction from images enabling pixel-wise predictions suitable for segmentation tasks. Additionally, U-Net [9] further improved 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 [10] and ResNet [11] into segmentation models further boosted performance. These 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 be 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 [12], 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 NLP to vision tasks, starting with Vision Transformers (ViT) [13], which model images as patch sequences using self-attention. For semantic segmentation, architectures like SETR [14] replace CNN backbones with transformer for a better global understanding, while hybrid models like Swin Transformer [15] 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 [16] 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 [17] [18]. 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 researches that addresses the local semantic map estimation with different approaches.

Cam2BEV [19] applies IPM to transform multi-camera segmented input images into the BEV domain which is feeded 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 was employed to test the method in real-world data.

HDMapNet [20] 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 [21] 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 capable of learning this transformation. Different to other methods, this work uses a GAN-based framework to manage occlusions 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 [22] introduces a two-layer MLP module for multi-camera feature fusion, followed by a decoder for semantic segmentation in indoor scenes. LSS [23] 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 planning. M²BEV [24] 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 [25] tackles occlusion estimation by employing a standard encoder-decoder framework combined with adversarial training, making it suitable for predicting amodal layouts of the driving scene. BEVFormer [26] 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 from multiple sensor such as FishingNet [27] which extends VPN to use multiple sensors, or HDMapNet which is also capable 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, how data augmentations are performed 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 three main blocks: BEV semantic segmentation experimentation, design and implementation of the preannotation pipeline and a third block of how the system is evaluated.

3.1 Segmentation experiment design: BEV2Seg_2

To address the hypothesis 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.

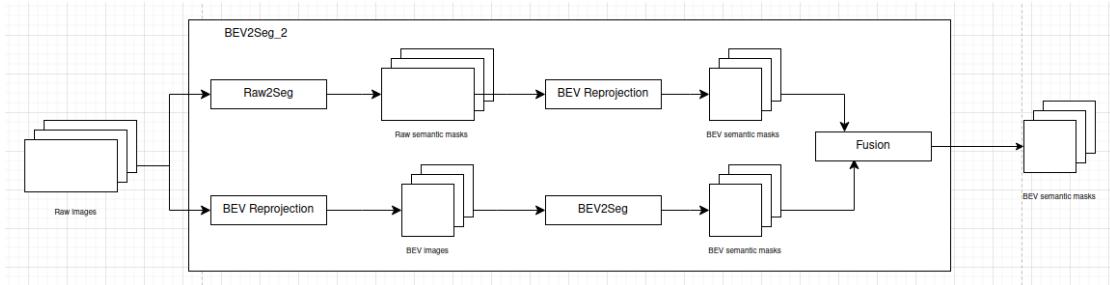


Figure 2: bev2seg_2 flow diagram.

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 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 [28] has been selected as the semantic segmentation model due to its balance between performance and efficiency. Additionally, it is integrated into the Huggingface [29] 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 [30] [31] [28].

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 [32], that defines 30 semantic classes and provides 5.000 frames with pixel-level high-quality annotations and 20.000 weakly annotated images; KITTI [33], which provices 400 annotated images with a 0.5 split for training and validation following the Cityscapes annotation format; ApolloScape [34], that provides 146.997 frames with corresponding pixel-level annotations and pose

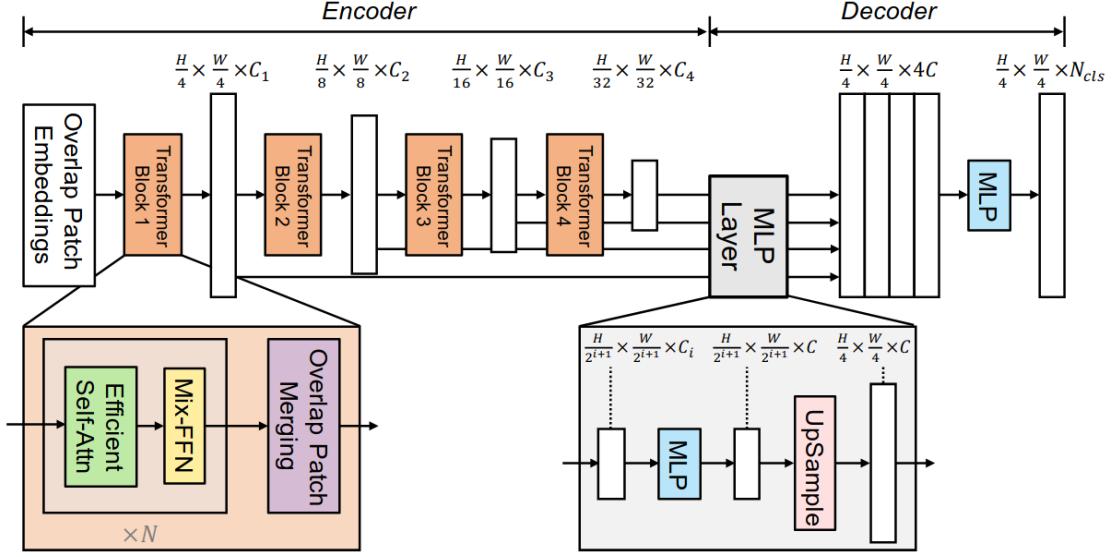


Figure 3: Segformer architecture

information for 25 labels; or NuImages, a subset of NuScenes [35], that contains annotated images on 26 different labels. For all of them, the ego pose and camera parameters metadata are provided.

There are other benchmarks [36] [37] but in order 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 approximately 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 OpenLABEL¹

¹<https://www.asam.net/standards/detail/openlabel/>

format, where metadata for each frame is stored. In the case of BEVDataset, images are reprojected into the BEV domain using the Video Content Descriptor (VCD) library [38]. 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 (LUT) 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 using a regular grid with a cell spacing of 1 meter, covering a total distance of 30 meters in front of the camera 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.

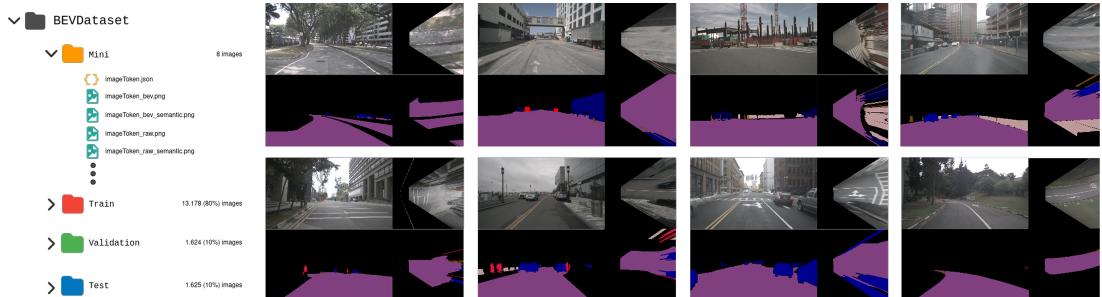


Figure 4: BEVDataset structure and mini set samples

Include the semantic class distribution of all splits.

3.1.3 Data augmentations

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 distortions.

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 [28]. 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 [39]. 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 5).

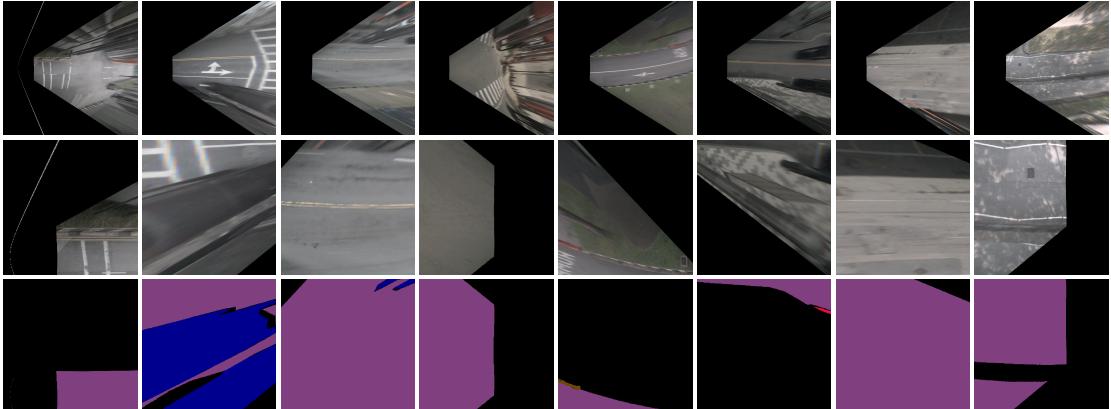


Figure 5: Random cropping and horizontal flipping on BEV images. Original BEV images on first row; random flipped and cropped images on second row and corresponding semantic masks on last row.

In this context, a different approach was also considered: applying geometric transformations 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 6).

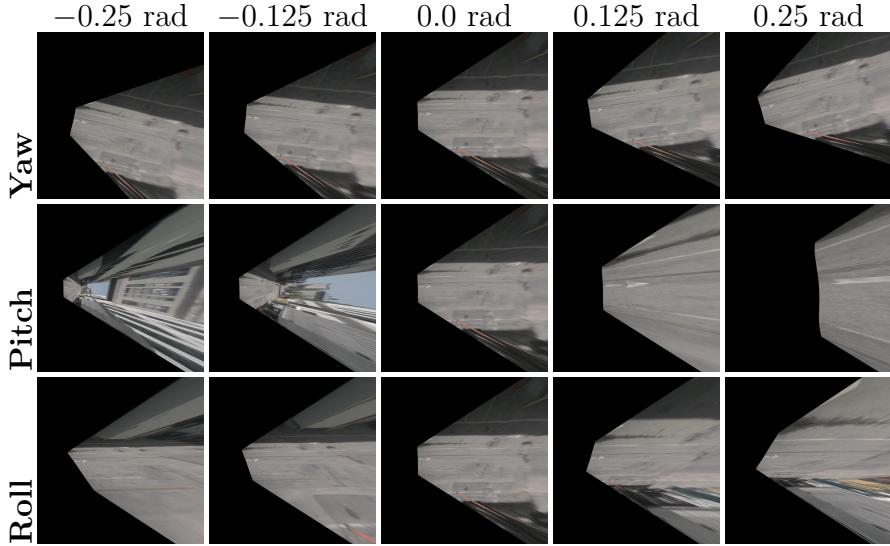


Figure 6: Effect of camera transformations on BEV projection. The first row shows variations in the yaw axes, the second in pitch, and the third in roll.

This raises an important question: *Which approach is more effective: traditional geometric data augmentation techniques or the modification of extrinsic parameters?*

3.1.4 Validation and comparison: Selection module

Mean intersection over union (mIoU) or Jaccard index is used as the metric for evaluation and comparison between the two approaches.

Fusion module in the pipeline. Semantic masks are selected from each model depending on its mIoU in the test set?

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 7, 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 [41] ecosystem, enabling further fine annotations by the user. Additionally, a visualization tool has been developed using the Open3D [40] library to facilitate the rendering process of vehicular scenes.

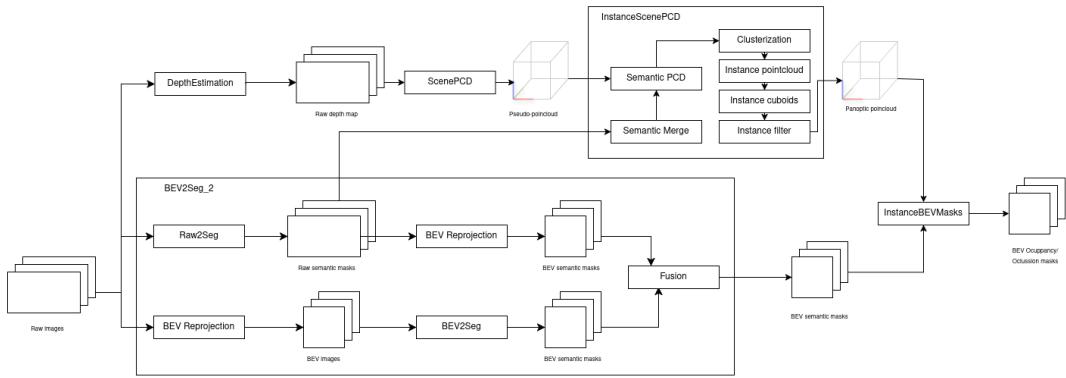


Figure 7: Annotation flow diagram.

3.2.1 Depth estimation

Traditional depth estimation techniques [42] rely on geometric and photometric marks to infer depth from images. For example, stereo vision (8a) 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) (8b) 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 (8c), which estimates depth based on the amount of blur in the images, and photometric stereo (8d), 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 (8e) 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.

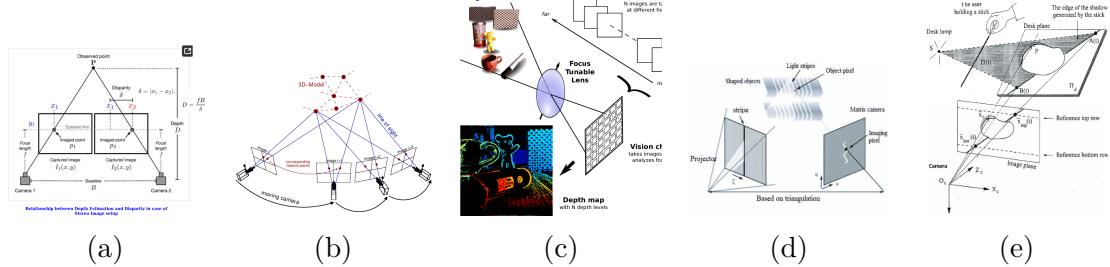


Figure 8: Traditional depth estimation techniques: (a) stereo-depth, (b) structure from motion, (c) depth-from-defocus, (d) photometric-depth, and (e) shape from shading.

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 [43] comes into play 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.

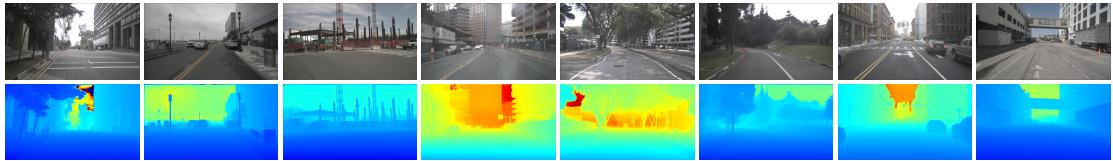


Figure 9: Depth maps of BEVDataset 'mini' estimated with Depth-Pro model.

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 10. 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.

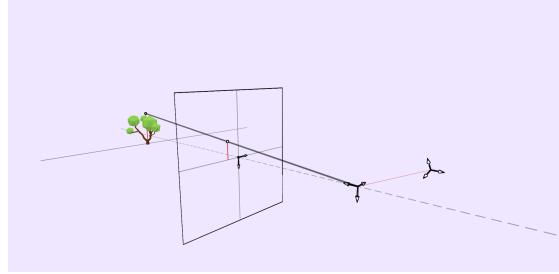


Figure 10: Ideal perspective camera model.

Starting with a 3D point expressed in world coordinates, $\mathbf{P}_w = (x_w, y_w, z_w)^\top$, the point can be transformed into 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 $[\mathbf{R} \mid \mathbf{t}]$.

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

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:

$$\tilde{\mathbf{p}}_i = (u, v, 1)^\top = \mathbf{K} \cdot \frac{1}{z_c} \tilde{\mathbf{P}}_c \quad (2)$$

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

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

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 $[0, 1, \dots, h-1]$, 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 11.

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

Algorithm 1 Pointcloud calculation

```

1: Input: Depth map  $D_{[h \times w]}$ , camera intrinsics  $K$ 
2: Output: Pointcloud  $P^{raw}_{[3 \times h \times w]}$ 
3: Initialize  $P_{[3 \times h \times w]} \leftarrow \mathbf{0}$ 
4: for  $i$  in  $[0, 1, \dots, h - 1]$  do
5:    $pix\_coords_{3 \times w} \leftarrow [0:w-1 \ i_w \ \mathbf{1}_w]$ 
6:    $cam\_rays_{3 \times w} \leftarrow \text{Undistord}(\text{Normalize}(K^{-1} \cdot pix\_coords))$ 
7:    $P^{raw}[:, i, :] \leftarrow cam\_rays[2, :] \cdot D[i, :] \triangleright$  Update the  $i$ -th row of  $P$ . The ray's z-value is the depth
8: end for
9: return  $P^{raw}$ 
```

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

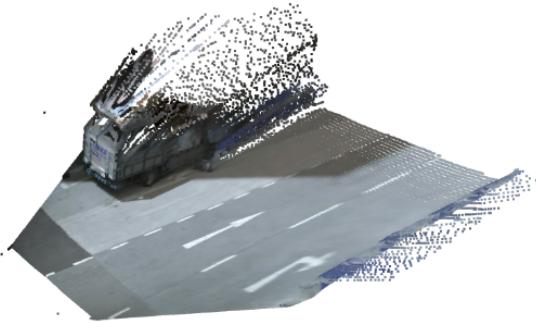


Figure 11: Reconstructed pointcloud.

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 12. 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.



Figure 12: Instance scene computation diagram.

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 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.

Once the semantic mask has been preprocessed and the point cloud computed, a

Merged Class	Original Classes
vehicle.car	vehicle.bus.bendy, vehicle.bus.rigid, vehicle.car, vehicle.construction, vehicle.emergency.ambulance, vehicle.emergency.police, vehicle.trailer, vehicle.truck
vehicle.motorcycle	vehicle.bicycle, vehicle.motorcycle

Table 2: Class merge dictionary used for semantic simplification.

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 13). 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.

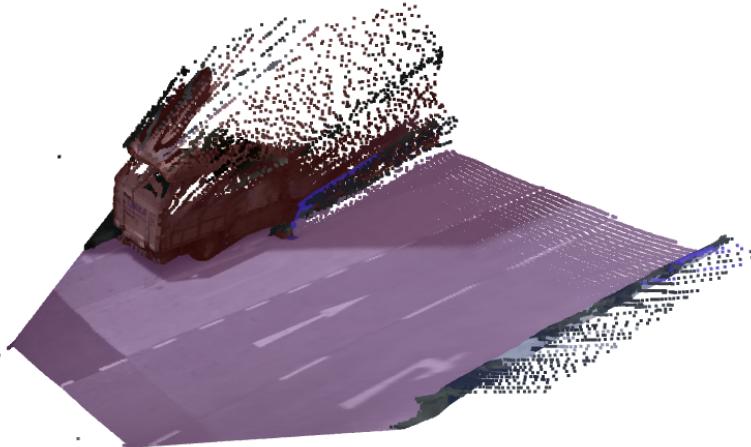


Figure 13: Colorized semantic pointcloud.

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 DBSCAN², which creates good instance clusters and removes part of the trail left by the depth estimation on object edges. Then, for

²<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>

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 14). 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.

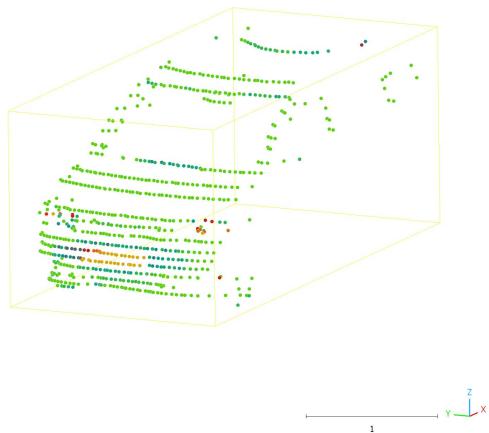


Figure 14: Oriented bounding box problems.

Change oriented bounding box problems figure. Complete this algorithm.

Finally, a filtering process is performed to ensure there are no noisy, redundant,

Algorithm 2 Instance Pointcloud Computation

```

1: Input:  $P^{inst}$  panoptic pointcloud
2: Output:  $P^{inst}$  filtered panoptic pointcloud
3: for  $label$  in  $P^{inst}$  do
4:   for  $i$  in  $P^{inst}[label]$  do
5:      $B \leftarrow$ 
6:   end for
7: end for
8: return  $P$ 

```

too far away or floating instances. Explain the filtering process: AABB intersection factor.

The results of this module are shown in Figure 15.

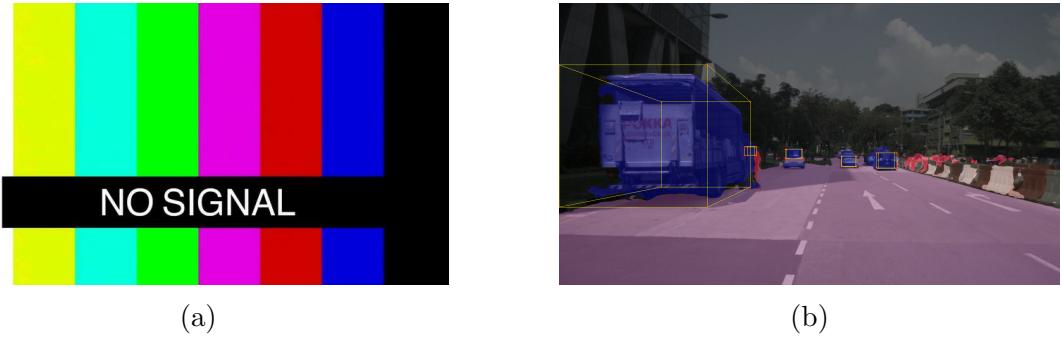


Figure 15: Resulting instances (a) in 3D and (b) projected into the front camera.

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 distorted shape of the semantic mask from the BEV perspective.

Explain why one cuboid base is associated with multiple BEV masks. 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 14), 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 overestimation of AABB, enables a more reliable approximation of the vehicle's actual footprint improving the accuracy of the resulting BEV masks.

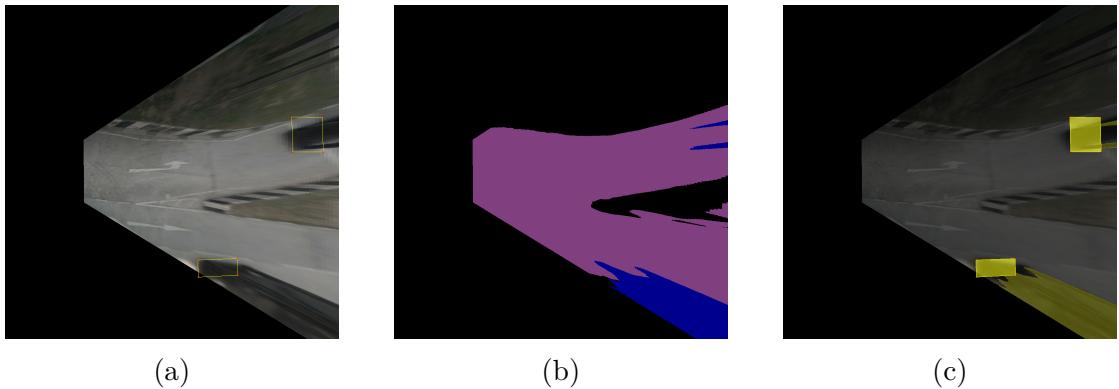


Figure 16: BEV occupancy and occlusion masks. (a) Input 3D detections; (b) input BEV semantic masks and (c) final BEV occupied-occluded mask.

3.3 Evaluation methodology

Methodology introduction has to be reviewed as this section is being moved to 4.

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: the encoder layers were not frozen during fine-tuning for the segmentation task, allowing the entire model to adapt to the data. Input image preprocessing was performed using the *SegformerImageProcessor*, which includes resizing the images to 512×512 pixels, rescaling pixel values by a factor of $1/255$, and normalizing using the mean and standard deviation values from ImageNet [45]. This ensures the input format is consistent with what the pretrained encoders expect.

Additionally, during training, semantic masks were provided with the `reduce_labels` parameter set to `False`, since our dataset includes the "background" class. This means gradients are computed for all image regions during training. All experiments were carried out using the same hardware configuration described in Table 3. Among the six available Segformer encoder variants, three were selected for evaluation: MiT-b0, MiT-b2, and MiT-b4.

Complete this table.

Component	Specifications	Num workers
CPU	-	8
GPU	-	2
Memmory	-	-
OS	-	-

Table 3: Hardware used for experiments

Regarding to the used notation, the models that follow the segment-then-IPM pipeline to obtain BEV semantic masks are referred to as `raw2seg_beav`, 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 techniques. This initial experiment was carried out to observe whether the models were able to learn and predict on the dataset and see if they suffered from overfitting. Also, for this 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.

I didn't find what is the segformer's loss function.

As shown in Figure 17, 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.

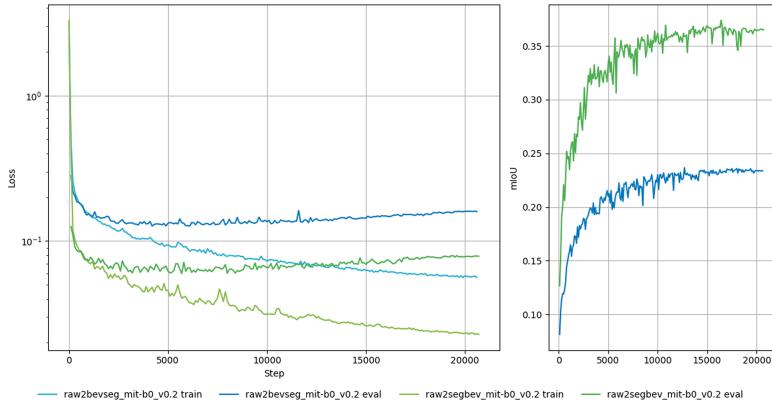


Figure 17: Training and evaluation loss of raw2seg_beav and raw2bev_seg MiT-b0 models without any regularization technique

It is important to note that the evaluation datasets differ between the two pipelines as one evaluates performance on BEV projections while the other does so on regular images motivating the gap in their validation results.

To address the overfitting, data augmentation strategies described in Section 3.1.3 were applied to introduce variability in the training data. Additionally, other regularization methods such as weight decay (also known as L2 regularization) were employed. Weight decay penalizes large weights during training, and makes the model more robust and less prone to memorizing irrelevant details. With these regularization techniques in place, the overfitting behavior was significantly

reduced as shown in Figure 18.

Temporal graph!!

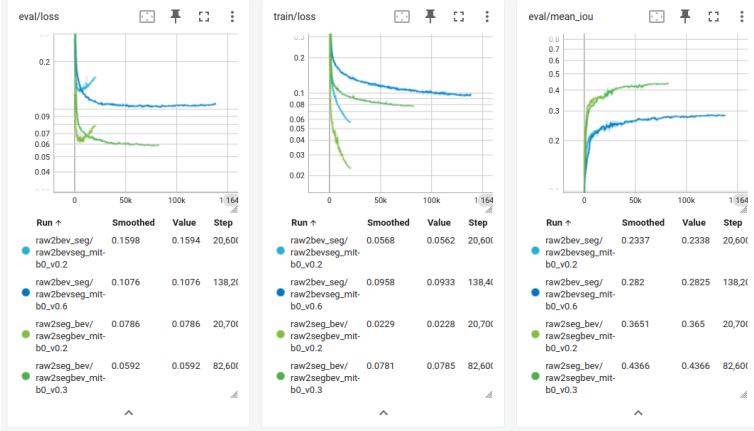


Figure 18: Before and after regularization techniques. Light colors before, darker colors after. BEV segmentation model uses normal data augmentations.

Comment that with model size MiT-b4 gradient accumulation techniques were used to fit the GPU memory specs.

Comment the max epochs set to 200 and the checkpoint strategy used: it is controversial whether to use the loss or mIoU. We have decided to use the loss.

With all of this in place, the training strategy is designed to support experimentation in order to answer two main research questions: In the case of segmentation in BEV images, *which data augmentation technique is more effective: using the same strategy applied in the other approach, or modifying the extrinsic camera parameters?*; and *which of the two approaches performs better for BEV image segmentation?*

4.1.1 BEV images data augmentation techniques comparison

Comparison between normal geometric data augmentation techniques and camera's extrinsic parameters modifications.

4.1.2 Pipeline comparison

raw2seg_bev results compared with raw2bev_seg results.

4.2 3D detections evaluations

Explain the selected NuScenes selected vehicle scene, and expose the metrics used (mIoU and v2v distance).

4.3 BEV masks evaluation

Groundtruth BEV masks could be generated from annotations and compute mIoU between the annotated BEV masks and ground truth ones.

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](https://doi.org/10.1109/ACCESS.2020.2983149).
- [2] Enrique Martí 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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1109/IVS.2016.7535402).
- [8] 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.
- [9] 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.
- [10] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: [1409.1556 \[cs.CV\]](https://arxiv.org/abs/1409.1556). URL: <https://arxiv.org/abs/1409.1556>.

- [11] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [12] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2016. arXiv: [1511.07122 \[cs.CV\]](https://arxiv.org/abs/1511.07122). URL: <https://arxiv.org/abs/1511.07122>.
- [13] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: [2010.11929 \[cs.CV\]](https://arxiv.org/abs/2010.11929). URL: <https://arxiv.org/abs/2010.11929>.
- [14] 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.
- [15] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: [2103.14030 \[cs.CV\]](https://arxiv.org/abs/2103.14030). URL: <https://arxiv.org/abs/2103.14030>.
- [16] 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](https://doi.org/10.1109/TPAMI.2013.185).
- [17] 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](https://doi.org/10.3390/s120404431). URL: <https://www.mdpi.com/1424-8220/12/4/4431>.
- [18] Ammar Abbas and Andrew Zisserman. *A Geometric Approach to Obtain a Bird’s Eye View from an Image*. 2020. arXiv: [1905.02231 \[cs.CV\]](https://arxiv.org/abs/1905.02231). URL: <https://arxiv.org/abs/1905.02231>.
- [19] 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](https://doi.org/10.1109/ITSC45102.2020.9294462).
- [20] Qi Li et al. *HDMapNet: An Online HD Map Construction and Evaluation Framework*. 2022. arXiv: [2107.06307 \[cs.CV\]](https://arxiv.org/abs/2107.06307). URL: <https://arxiv.org/abs/2107.06307>.
- [21] 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.

- [22] 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](https://doi.org/10.1109/LRA.2020.3004325).
- [23] 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.
- [24] Enze Xie et al. "M[^] 2BEV: Multi-Camera Joint 3D Detection and Segmentation with Unified Birds-Eye View Representation". In: *arXiv preprint arXiv:2204.05088* (2022).
- [25] 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.
- [26] Zhiqi Li et al. *BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers*. 2022. arXiv: [2203.17270 \[cs.CV\]](https://arxiv.org/abs/2203.17270). URL: <https://arxiv.org/abs/2203.17270>.
- [27] Noureldin Hendy et al. *FISHING Net: Future Inference of Semantic Heatmaps In Grids*. 2020. arXiv: [2006 . 09917 \[cs.CV\]](https://arxiv.org/abs/2006.09917). URL: <https://arxiv.org/abs/2006.09917>.
- [28] Enze Xie et al. "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers". In: *CoRR* abs/2105.15203 (2021). arXiv: [2105.15203](https://arxiv.org/abs/2105.15203). URL: <https://arxiv.org/abs/2105.15203>.
- [29] 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>.
- [30] Yuanduo Hong et al. *Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes*. 2021. arXiv: [2101.06085 \[cs.CV\]](https://arxiv.org/abs/2101.06085). URL: <https://arxiv.org/abs/2101.06085>.
- [31] Jiacong Xu, Zixiang Xiong, and Shankar P. Bhattacharyya. *PIDNet: A Real-time Semantic Segmentation Network Inspired by PID Controllers*. 2023. arXiv: [2206.02066 \[cs.CV\]](https://arxiv.org/abs/2206.02066). URL: <https://arxiv.org/abs/2206.02066>.
- [32] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. arXiv: [1604.01685 \[cs.CV\]](https://arxiv.org/abs/1604.01685). URL: <https://arxiv.org/abs/1604.01685>.
- [33] 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.

- [34] 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](https://doi.org/10.1109/tpami.2019.2926463). URL: <http://dx.doi.org/10.1109/TPAMI.2019.2926463>.
- [35] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *CVPR*. 2020.
- [36] Oliver Zendel et al. “Unifying Panoptic Segmentation for Autonomous Driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 21351–21360.
- [37] 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>.
- [38] Marcos Nieto, Ortí 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>.
- [39] 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).
- [40] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).
- [41] Itziar Urbieta et al. “Correction to: WebLabel: OpenLABEL-compliant multi-sensor labelling”. In: *Multimedia Tools and Applications* 83.9 (Mar. 2024), pp. 26525–26525. ISSN: 1573-7721. DOI: [10.1007/s11042-023-17602-0](https://doi.org/10.1007/s11042-023-17602-0). URL: <https://doi.org/10.1007/s11042-023-17602-0>.
- [42] 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](https://doi.org/10.1109/DSC50466.2020.00028).
- [43] Aleksei Bochkovskii et al. *Depth Pro: Sharp Monocular Metric Depth in Less Than a Second*. 2024. URL: <https://arxiv.org/abs/2410.02073>.
- [44] Gregory G Slabaugh. “Computing Euler angles from a rotation matrix”. In: *Retrieved on August 6.2000* (1999), pp. 39–63.
- [45] 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](https://doi.org/10.1109/CVPR.2009.5206848).

Annexes

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 Vicomtech³.

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 4: Semantic labels defined for NuImages masks

³<https://www.vicomtech.org/en/>

Rotation matrix to euler angles

Here [44] is the pseudocode.

Color palette

Hex Code	Color Sample
#F44336	
#EF5350	
#F1464E	
#FF9800	
#FF0064	
#4CAF50	
#86BB48	
#39C39E	
#0277BD	
#0097A7	
#2AB0D2	
#01A1FF	
#0073B7	

Table 5: Color Palette