MASTER THESIS

Transformer-based Lidar-RGB Fusion for Semantic Grid Prediction in Autonomous Vehicles

Author:
Gustavo A. SALAZAR-GOMEZ

Supervisors:
David SIERRA-GONZALEZ, PhD
Christian LAUGIER, HDR, Research Director

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science in Mobile, Autonomous and Robotic Systems
performed at

INRIA - CHROMA Team

August 19, 2022

Abstract

Grenoble INP
École nationale supérieure de l'énergie, l'eau et l'environnement - ENSE^3

Master of Science in Mobile, Autonomous and Robotic Systems

Transformer-based Lidar-RGB Fusion for Semantic Grid Prediction in Autonomous Vehicles

by Gustavo A. SALAZAR-GOMEZ

Semantic grids are simple and compact top-view representations of a 3D environment used in a variety of robotics tasks, including autonomous vehicles. These grids are composed by several cells that contain the occupancy status (e.g. Free or Occupied), and the object class related to it (e.g. vehicle, road). In order to obtain these semantic grids, some works rely exclusively on camera data, while others rely only in Lidar input, obtaining state-of-the-art results, nevertheless, neglecting the benefits of using both complementary modalities at the same time. In this work, we present a Transformer-based Lidar-RGB fusion network for semantic grid prediction. This architecture receives as input multi-camera views and a Lidar point cloud, and transforms this incoming data onto top-view features that are fused at multiple scales using transformers. Finally, a decoder processes the fused features to predict a semantic grid. This model is trained and tested for Vehicle, Drivable area, Lane divider and Walkway classes, and is evaluated in challenging conditions such as Night and Rain. The results show significant improvement and superior performance for the proposed architecture using Transformer feature fusion in the Vehicle, Drivable area and Walkway classes. This compared with naive fusion methods such as concatenation, and with single modality models based on camera-only and Lidar-only data.

Acknowledgements

First, I would like to express my deepest appreciation to my supervisor David Sierra, for his patience, expertise and guidance during this project. Additionally, this endeavor would not have been possible without Christian Laugier, his feedback and the opportunity he gave me to be part of CHROMA team.

I'm also grateful to my co-workers in INRIA's CHROMA team, for their advice, feedback and long discussions sessions, that helped me to complete this work.

Special thanks to the Grid5000 team, as the training and experiments presented in this work were performed using the Grid'5000 testbed. Supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

I could not have undertaken this journey without my family, who have supported me since the beginning, and have always pushed me to achieve my goals. I am also thankful to my girlfriend, Manuela, for her support, laughs and for being my travel companion during this amazing voyage. Finally, thanks should also go to all my friends, old and new, that in a way or another have supported me, and made this time abroad a stunning adventure.

Contents

Abstract ii

Acknowledgements iii

1 Introduction 1

2 State-of-the-art 2
- 2.1 Semantic Grid Prediction . . . 2
- 2.2 Camera-only methods . . . 3
  - 2.2.1 Lift-Splat-Shoot [23] . . . 3
  - 2.2.2 Cross-view Transformers for real-time Map-view Semantic Segmentation [41] . . . 4
- 2.3 Lidar-only methods . . . 4
  - 2.3.1 PillarSegNet: Pillar-based Semantic Grid Map Estimation using Sparse LiDAR Data [9] . . . 5
- 2.4 Multi-modal sensor fusion . . . 6
  - 2.4.1 Semantic Grid Estimation with a Hybrid Bayesian and Deep Neural Network Approach [7] . . . 6
  - 2.4.2 FISHING net: Future inference of semantic heatmaps in grids [12] . . . 7
- 2.5 Lidar and Camera Sensor fusion . . . 8
  - 2.5.1 Fusion relying on 2D detection . . . 8
    - 2.5.1.1 Frustum PointNets for 3D Object Detection from RGB-D Data [25] . . . 8
  - 2.5.2 Point Decoration . . . 9
    - 2.5.2.1 PointPainting: Sequential Fusion for 3D Object Detection [37] . . . 9
  - 2.5.3 Mid-level features fusion . . . 10
    - 2.5.3.1 DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection [16] . . . 11
    - 2.5.3.2 TransFuser: Multi-Modal Fusion Transformer for End-to-End Autonomous Driving [24] . . . 11

3 Transformer-based Lidar-RGB fusion network for semantic grid prediction 13
- 3.1 RGB Feature Projection . . . 14
- 3.2 Point Cloud Encoding . . . 14
- 3.3 Multi-scale fusion with Transformers . . . 15
- 3.4 Semantic Grid Generation . . . 16
- 3.5 Training . . . 17
- 3.6 Experimentation . . . 17
  - 3.6.1 Dataset . . . 18
  - 3.6.2 Metrics . . . 18
  - 3.6.3 Ablation study . . . 19

3.7 Quantitative Results 19
3.8 Qualitative Results 21

4 Conclusions **24**
4.1 Summary 24
4.2 Future work 25

A Semantic grid predictions per class **26**
A.1 Vehicle Class 26
A.2 Drivable area Class 27
A.3 Walkway Class 28
A.4 Lane Divider Class 29

B Semantic grid predictions with all classes combined **30**
B.1 Semantic grids combined 30
B.2 Semantic grids combined at Night and Rain scenarios 32

Bibliography **34**

List of Figures

2.1 Lift-Splat-Shoot architecture, image taken from [23]. 3
2.2 Lift module explanation, image taken from [23]. 3
2.3 Cross-view architecture, image taken from [41]. 4
2.4 PillarSegNet architecture, image taken from [9]. 5
2.5 Dense and Sparse Lidar scans, image taken from [9]. 5
2.6 Hybrid Bayesian filtering and Deep Neural Network pipeline, image taken from [7]. 6
2.7 FISHING net architecture, image taken from [12]. 7
2.8 Frustrum PointNets pipeline, image taken from [25]. 8
2.9 Frustrum PointNets architecture, image taken from [25]. 9
2.10 PointPainting pipeline, image taken from [37]. 10
2.11 DeepFusion architecture, image taken from [16]. 11
2.12 TransFuser architecture, image taken from [24]. 12

3.1 Proposed Transformer-based Lidar-RGB fusion architecture for semantic grid prediction. 13
3.2 RGB Feature Projection pipeline. 14
3.3 Point Cloud Encoding pipeline. 15
3.4 Multi-scale fusion with transformers. 16
3.5 Semantic Grid Generation pipeline. 17
3.6 Car set up for nuScenes dataset, image taken from [4]. 18
3.7 Intersection and union. 18
3.8 Predictions for Vehicle class. 22
3.9 Predictions for Drivable area class. 22
3.10 Predictions for Walkway class. 23
3.11 Predictions for Lane divider class. 23
3.12 Combined TFGrid predictions for all four classes. 23

List of Tables

3.1 IOU results for different number of transformer fusion modules. 20
3.2 Inference time and mIOU results (on vehicle and drivable area classes) for different number of transformer fusion modules. 20
3.3 Results on the nuScenes validation split. It is compared the IOU of the generated semantic grids for all models. Best results are presented in **bold** font, second best in Blue. 21
3.4 Results on the nuScenes validation split. It is compared the IOU of all models in only **Night** conditions and only **Rain** conditions. Best results are presented in **bold** font, second best in Blue. 21

Dedicated to my parents, who have always believed in me.

Chapter 1

Introduction

One of the main and vital components in autonomous vehicles, is the accurate and robust perception of the environment. These vehicles must be able to interpret the data and create precise representations from the surrounding environment using multiple sensors inputs. In order to segment obstacles and regions in a top-view map or Bird's-Eye-View (BEV), crucial representations used for motion planning.

Semantic Grids can be described as BEV representations of the 3D environment, where the cells of the grid outline an occupancy status associated with a semantic class. These representations compose a simpler version of the complex environment.

Furthermore, semantic grid prediction algorithms can be classified depending on the input data modality. Several methods are based entirely on camera images [28, 23, 13, 20, 21, 41], some others rely only on Lidar point clouds [9], and a few others fuse different sensor modalities [8, 12].

Between all camera-based methods, the main difference is in how they transform the information from the 2D image into the top-view semantic grid. Some works as Lift-Splat-Shoot predict a discrete set of depths per pixel, creating a pseudo point cloud [23]. Other works use cross-attention, combining the camera views with top-view embeddings to predict the BEV [41].

In contrast, methods that rely only on Lidar input have an advantage transforming the point clouds into semantic grids [9], as the laser returns already provide 3D coordinates in the scene. Nevertheless, due to their sparsity it is more difficult to detect distant objects. Furthermore, categories that rely on colors or textures, such as lane markings, are even more challenging.

In order to exploit the best of multiple data modalities, some authors have proposed different fusion architectures to combine different sensor inputs and predict semantic grids [8, 12]. At the beginning of this work, the number of fusion methods in the literature for semantic grid prediction was limited to simple models (as FISHING Net [12]). On the other hand, Transformers offer a way to combine global context data from the 3D scene, using the attention mechanism to attend to relevant features. No models had explored using transformers for fusion and semantic grid prediction tasks.

Finally, in this work we explore the prediction of semantic grids fusing camera and Lidar data. Specifically, we exploit transformers and the attention mechanism to fuse both data modalities [36, 24]. In recent works, transformer-based architectures have been used to fuse multi-modal data, in particular camera and Lidar inputs showing state-of-the-art results on 3D object detection [2]. In other work, the authors propose an architecture for motion planning that fuses camera and Lidar inputs at different scales using transformers [24]. Influenced by this, in this work we present a Transformer-based Lidar-RGB fusion network for semantic grid prediction, receiving as input multi-camera views and a Lidar point cloud, and fusing intermediate features at different scales from each data modality to generate a semantic grid.

Chapter 2

State-of-the-art

2.1 Semantic Grid Prediction

Semantic grids are 2D representations in form of a discrete grid of the vehicle's surrounding world, where each cell of the grid has an associated class (e.g. road, vehicle, sidewalk) that is of interest for the current task. They constitute a compressed and simple way of representing a complex 3D environment. The main objective behind the semantic grid maps is to be able to represent a variety of *"things"*, defined by Roddick as objects with a well defined shape and position (e.g. vehicles)[27], and *"stuff"* described as amorphous patches representing sections of the environment (e.g. grass, road or sidewalks).

A simplified version of semantic grids, where only the occupancy for each cell is predicted, has been used for a long time in the mobile robotics domain due to its simplicity.

Then, semantic grids can be disaggregated in a number of occupancy grids maps, introduced originally by Elfes [6]. Given a Occupancy field $O(c)$ and a set of spatial coordinates $c = (c_1, c_2 \dots c_n)$, a state variable $s(C)$ associated with a cell C of the occupancy grid is defined as a discrete variable with possible states $occupied$ and $empty$. Since the cell states are exclusive:

$$P[s(C) = occupied] + P[s(C) = empty] = 1 \tag{2.1}$$

To represent the global context, we can use a map containing multiple cells:

$$m = \{m_{x,y}\}, x = \{1, ..., w\} \; and \; y = \{1, ..., h\} \tag{2.2}$$

where x and y are discrete coordinates for a cell in the grid map, and each cell $m_{x,y}$ can be $occupied$ or $empty$ as described before, w and h represent the size of the map.

For a semantic grid map, given an observation of the world (in the case of this work containing images and points in a point cloud) and a number of classes to predict k, we obtain a map per semantic class:

$$m^k = \{m^k_{x,y}\} \tag{2.3}$$

When the occupancy grid maps are generated from the same observation but for different k class, each cell $m^k_{x,y} \in (0, 1)$ represents the confidence for this class k to be occupying the cell. Next, a threshold is applied to the confidence value in all m^k cells so they can indicate when they are empty or occupied, and the occupancy grids are merged into a single grid. Some classes are given higher priorities than others, in this sense, a car can be on top of a road and not occluded by it in the merged semantic map.

We can classify the semantic grid prediction methods based on the modality of their input data; some works rely exclusively on camera data [28, 23, 13, 20, 21, 41], others on Lidar range data [9], and finally some approaches fuse multiple sensor modalities to predict the grids [8, 12].

2.2 Camera-only methods

The methods described in this section rely entirely on the cameras information, fusing the data from multi-view images in the same observation to finally project a semantic grid prediction.

2.2.1 Lift-Splat-Shoot [23]

Philion *et al.* propose an end-to-end architecture that is able to directly extract a 2D top-view representation of a scene given image data from an arbitrary number of cameras from the autonomous vehicle. The core idea behind their approach is to "lift" each image individually into a frustum of features for each camera, then "splat" all frustums into a rasterized bird's-eye-view grid, to finally "shoot" different trajectories for planning, based on the BEV information. The overall architecture can be seen in Figure 2.1.

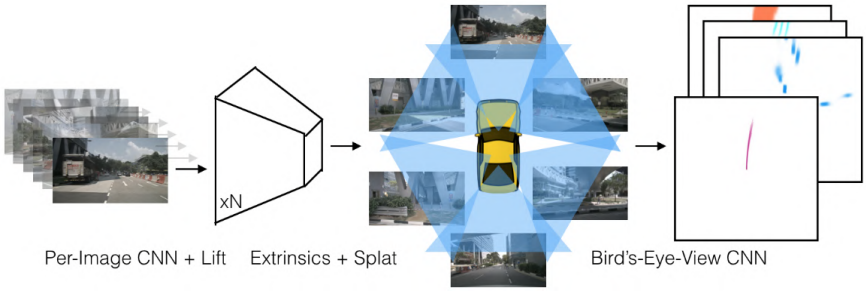


FIGURE 2.1: Lift-Splat-Shoot architecture, image taken from [23].

The main idea of the initial module is to "lift" each image from a 2D coordinate system to a 3-dimensional frame shared across all cameras. The challenge faced by Philion *et al.* is that for monocular cameras a depth is required to be able to transform the data to the reference coordinate frame, but this information is not available from these cameras. Their solution is to generate a discrete set of D depths with a distribution representations for all possible D for each pixel, as seen in Figure 2.2. After, a set of C features are obtained across all possible depths, generating a *"point cloud"* in the 3D frame.

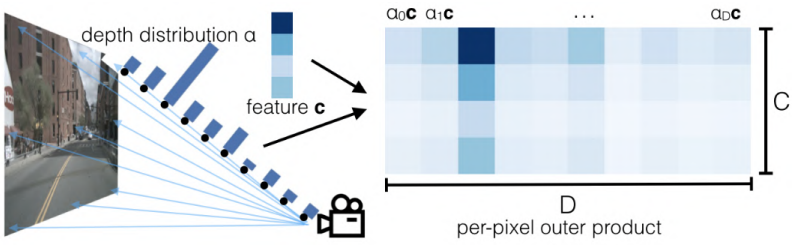


FIGURE 2.2: Lift module explanation, image taken from [23].

Based on the work of Lang *et al.* [15], the Splat section follows the main idea behind the PointPillars architecture, where pillars are created and the points are assigned to the nearest pillar. Then, a sum pooling operation creates a $C \times H \times W$ representation that after being processed by a CNN predicts the semantic grid map.

2.2.2 Cross-view Transformers for real-time Map-view Semantic Segmentation [41]

In this method, Zhou *et al.* present an encoder-decoder architecture for semantic map-view prediction. An image-encoder module produces a multi-scale feature representation of each input image. A cross-view attention mechanism then aggregates multi-scale features into a shared map-view representation. This mechanism relies on a positional embedding that is based on the geometric structure of the scene, learning to match the camera-view and map-view locations.

An important remark is that all cameras share the same image-encoder, but use a different positional embedding depending on their individual camera calibration. Finally, a convolutional decoder upsamples the refined map-view embedding and creates the final segmentation map, as can be seen in Figure 2.3.

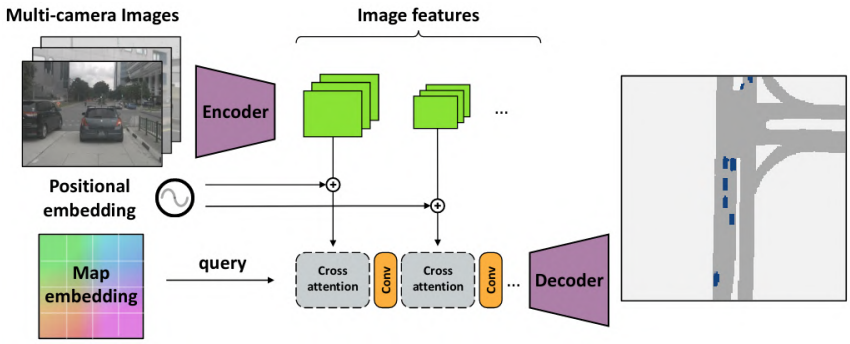


FIGURE 2.3: Cross-view architecture, image taken from [41].

The core idea behind the cross-view transformer is to combine the positional encodings from the map embedding and multi-camera images, allowing each map-view coordinate to attend one or more image locations. It is important to notice that, not every map-view location has a corresponding image patch in each view (e.g. Left cameras do not see the right side, right cameras do not see the left side).

Zhou *et al.* let the attention mechanism select both camera and location within each camera with corresponding map-view and camera-view perspectives. Consequently, they first combine all camera positional embeddings into a single key vector, and all image features into a single value vector, then these are combined and used to determine the attention keys. Finally, softmax-cross-attention is computed between the keys, values and map queries. This constitute the cross-view attention block, base of the cross-view transformer architecture.

2.3 Lidar-only methods

The method discussed in the following section relies entirely on the Lidar point cloud data to encode the information from the 3D surrounding environment into a 2D semantic grid.

2.3.1 PillarSegNet: Pillar-based Semantic Grid Map Estimation using Sparse LiDAR Data [9]

In the PillarSegNet method Fei *et al.* present a model that, given a 3D point cloud, encodes pillar features and occupancy features in two parallel streams. The pillar features are encoded following [15], while the occupancy features are encoded using a model-based ray-casting.

As a next step, an encoder-decoder U-Net is used to created the final dense semantic grid map from the aggregated features[29], as seen in Figure 2.4. It is important to notice that since the prediction is based on the observability map it excludes the occluded areas from the Lidar.

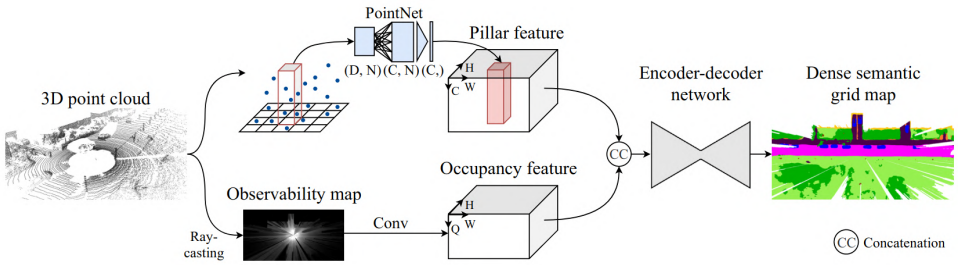


FIGURE 2.4: PillarSegNet architecture, image taken from [9].

For the pillar features stream Fei *et al.* follow PointPillars [15], where for a point with x, y, z coordinates and r reflectance, the coordinates are augmented computing the offset from the arithmetic mean of all point inside the pillar, and calculating the offset to the pillar center. The dimension of the vector with the resulting augmentation becomes $D = 10$. Then, all points from pillars are processed by a simplified PoinNet [26], and a top-view representation with all features is created.

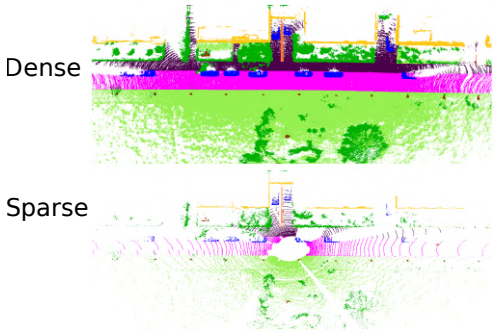


FIGURE 2.5: Dense and Sparse Lidar scans, image taken from [9].

The points in the 3D point cloud captured by the Lidar are the result of a ray-casting. Indicating that there is missing information where the ray can't reach, when occluded by other obstacles. Fei *et al.* argue that the observability information might be beneficial for the semantic grid prediction. Using a convolutional layer to extract occupancy features, the model includes to the final features a multi-channel map that represents the 3D occupancy information.

The encoder-decoder U-Net allows Fei *et al.* to combine semantic and spatial information from the features [29]. After that, a segmentation head with a set of

1×1 convolutions and softmax activation function, is used to produce probabilities for each cell in the grid.

It is important to remark that, in order to generate a dense Lidar scan for training, consecutive Lidar sweeps are superimposed, being transformed to the coordinate system of the current scan using the pose information of each scan. This allows Fei *et al.* to have a much denser point cloud with a limit of 40 sweeps maximum, as seen in Figure 2.5.

2.4 Multi-modal sensor fusion

This section explores existing methods and algorithms that fuse multiple data modalities. From a set of different sensors (e.g. Lidar, Radar, multi-view Cameras) in order to predict a semantic grid.

2.4.1 Semantic Grid Estimation with a Hybrid Bayesian and Deep Neural Network Approach [7]

Erkent *et al.* express their interest in 2D egocentric representation, and propose a model that estimates an occupancy semantic grid. Their model leverage and fuse data from sensor modalities such as Lidar point clouds, camera images and odometry. Their approach is a hybrid algorithm that benefits from deep neural networks and Bayesian filtering, as seen in Figure 2.6. The main task of the model is to estimate the semantic grid for classes { Building, Sky, Road, Vegetation, Sidewalk, Car, Pedestrian, Cyclist, Signage, Fence, Free, Static, Dynamic}.

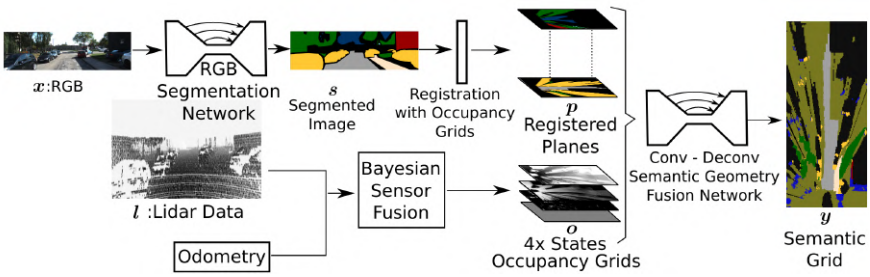


FIGURE 2.6: Hybrid Bayesian filtering and Deep Neural Network pipeline, image taken from [7].

In this architecture, Bayesian particle filtering processes the Lidar and odometry data, in order to predict an egocentric bird's-eye-view containing the occupancy information from the surroundings of the vehicle. Deep learning is used to perform semantic segmentation from the camera images, and finally to fuse the occupancy grid with the semantics to predict the semantic grid.

For the RGB input, Erkent *et al.* use a pre-trained SegNet version for the semantic segmentation [1], although its results are not the highest, the authors find the trade-off between the runtime and accuracy very favorable for this encoder-decoder network. They use the parameters from a pre-trained version with VGG16 architecture for object recognition [34].

For Lidar and odometry data, the model uses Bayesian filtering methods to predict the occupancy and dynamic state of the surroundings and represent it in a occupancy grid map. The algorithm used by the authors is the Conditional Monte Carlo

Dense Occupancy Tracker (CMCDOT) [30], which indicates if a cell is empty, statically occupied, dynamically occupied of unknown. It is important to notice that an empty cell does not mean that it can be drivable, e.g. a sidewalk could be free but the vehicle is not allowed to drive on it.

In the final stage, Erkent *et al.* express that the fusion between these data modalities is a significant problem, as the output from the Lidar stream offers an occupancy grid, and the output from the camera data is projective semantic information. To solve this, the authors propose to transform the projective input into a bird's-eye-view intermediate representation p, that would be compatible with the coordinate system of the occupancy grid. Finally, both representations are fed into a deep neural network that predicts the semantic grid.

2.4.2 FISHING net: Future inference of semantic heatmaps in grids [12]

In FISHING net, Hendy *et al.* indicate that top-down representations aggregate geometric and semantic understanding, and propose to leverage these representations to serve as a common output between the data modalities. Some of the benefits expressed rely on the argument that top-down representations are independent of the sensor characteristics, and are thus extensible to new sensor modalities. Furthermore, the late fusion task is simplified by sharing spatial representations in a concise way.

In addition to finding a representation of the current scene, FISHING net predicts the future states of the objects of interest, in order to make effective decisions considering the behaviour prediction. The main task of this model is to predict vehicle, background and pedestrian classes. Hendy *et al.* propose a method where the pipeline for the camera input relies on the ResNet architecture and use a modified View Parsing Network in the middle of the model [11, 21]. In contrast, for the Lidar and radar inputs a U-Net architecture is used [29], as seen in Figure 2.7.

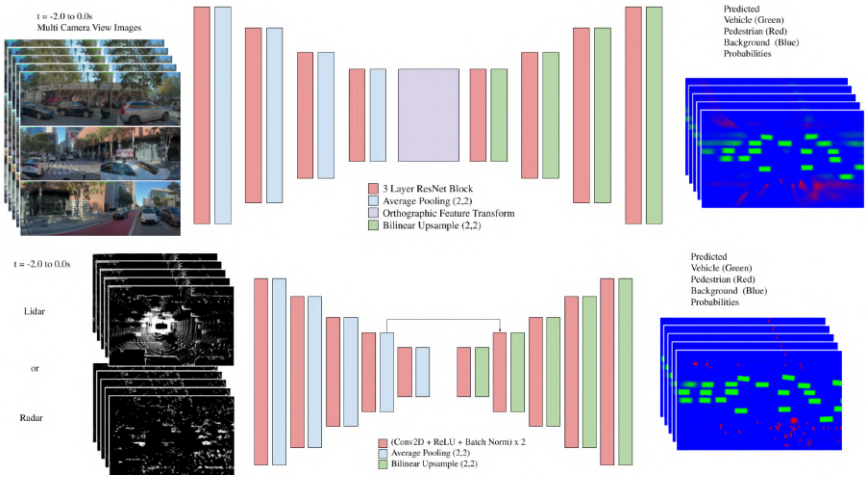


FIGURE 2.7: FISHING net architecture, image taken from [12].

For the image input, multi-view cameras are used providing a 360 degrees field of view. The data from the cameras is fed to the vision network architecture composed by encoder-decoder based on a fully convolutional ResNet model [11]. It uses three layer ResNet blocks for encoder and decoder, and in the middle a modified View Parsing Network that the author call orthographic feature transform layer [11, 21].

Lidar and radar inputs are processed by an architecture based on U-Net [29]. Similarly, with an encoder-decoder shape this architecture uses 5 blocks with convolutional, batch normalization and average pooling layers. This network includes a skip connection from the fourth block of encoder to the second of the decoder.

In order to fuse the representation outputs, as these are in the same top-view frame, Hendy *et al.* aggregate them by applying an aggregation function to the softmax values across modalities outputs. The authors evaluate two functions. The first is an arithmetic average, allowing to reduce the variance of the output. The second is a priority pool where classes are set a priority value as pedestrian: 3, vehicle: 2, background: 1. Then, if the modalities disagree on a pixel class, the model picks the highest priority predicted, breaking ties by the larger value of the softmax prediction.

2.5 Lidar and Camera Sensor fusion

This section presents different methods and algorithms used to fuse Lidar and Camera sensor data for autonomous vehicles, regardless of their task.

These algorithms can be classified by the stage at which the data is fused (e.g. early stage, mid stage, late stage) and the way the data from the sensors is fused (e.g. concatenation, decoration, summation, attention mechanism). This section will explore the following: Fusion relying on 2D detection, Point decoration and Fusion of mid-level features.

2.5.1 Fusion relying on 2D detection

This method depends on state of the art 2D detectors for finding and detecting the objects of interest in the image, while the points from the 3D point cloud are taken from this region and processed.

2.5.1.1 Frustum PointNets for 3D Object Detection from RGB-D Data [25]

For Frustum PointNets Qi *et al.* address a key challenge on how to locate 3D objects in a 3D environment, using the capabilities of PointNet [26], being able to predict in a point cloud a semantic class for each point.

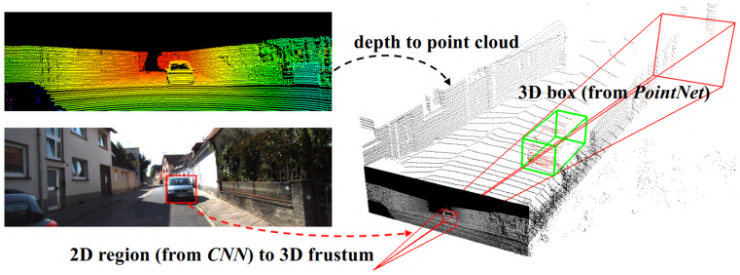


FIGURE 2.8: Frustrum PointNets pipeline, image taken from [25].

In order to reduce the search space in the 3D representation, the authors decided to take advantage of a mature 2D object detector(trained CNN), where it detects an object in the image and after extruding the region a 3D frustum is extracted, as seen in Figure 2.8. With the 3D space limited by the bounds of the frustum, the model performs an instance segmentation and 3D bounding box regression. The

segmentation network is able to predict a 3D mask of the object, and the regression head estimates the coordinates and heading of the 3D bounding box.

As seen in Figure 2.9, the model proposed by Qi *et al.* is composed by 3 main modules; 2D detection and frustum proposal, 3D instance segmentation and 3D bounding box regression.

For the 2D detections, Qi *et al.* leverage mature models to propose the regions in the RGB images. From the 2D bounding box the frustum is lifted defining the 3D limits for the search space. Then, all points between these limits are considered. This model is pre-trained using ImageNet classification and COCO object detection datasets [31, 18]. Finally, it is fine tuned using KITTI 2D object detection dataset to classify and predict the 2D boxes [10].

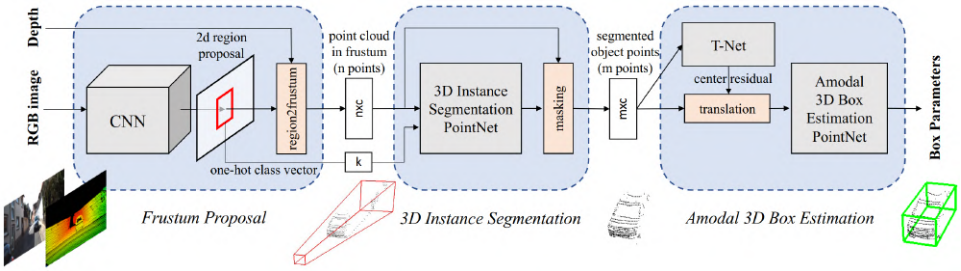


FIGURE 2.9: Frustrum PointNets architecture, image taken from [25].

The instance segmentation module processes the point cloud inside the frustum region, predicting the probability for each point indicating if it belongs to the object of interest. Other points could be part of areas such as vegetation or background, that are considered as non-relevant. Then, the segmentation PointNet filters those points that are not part of the object detected [26].

For the 3D bounding box, Qi *et al.* explained that the architecture is similar to the one for object classification in PointNet [26], but the output is modified to predict the parameters center (c_x, c_y, c_z), size (h, w, l) and heading angle θ for a 3D bounding box.

An important remark to consider, is that in this fusion method we can appreciate that both data modalities are used separately and sequentially one after the other. A drawback for this kind of algorithms is that if the first stage fails to recognize or has any kind of interference, the whole pipeline is at risk of failing in the prediction. Thus, environmental conditions such as night or rain might have a strong disruption effect in the RGB data and in the regression of the parameters.

2.5.2 Point Decoration

Point decoration methods are part of early stage fusion, as they initially add features from camera data modality to the 3D input in order to add class scores or image features that can help in the point cloud prediction, as seen in PointPainting and PointAugmenting [37, 38].

2.5.2.1 PointPainting: Sequential Fusion for 3D Object Detection [37]

In PointPainting, Vora *et al.* address the problem of fusing Lidar and Camera data modalities. The authors explain that while point clouds provide really accurate range view information, it fails in resolution and texture. Secondly, cameras offer good quality data in terms of texture and color, but lacks information in depth.

As seen in Figure 2.10, this architecture is composed by three main stages: first, Semantic Segmentation performed in the data provided by the camera to a pixel wise level. Second, is where the fusion happens and the points in the point cloud are painted with the semantic segmentation scores. Finally, a 3D detection network that predicts the bounding boxes.

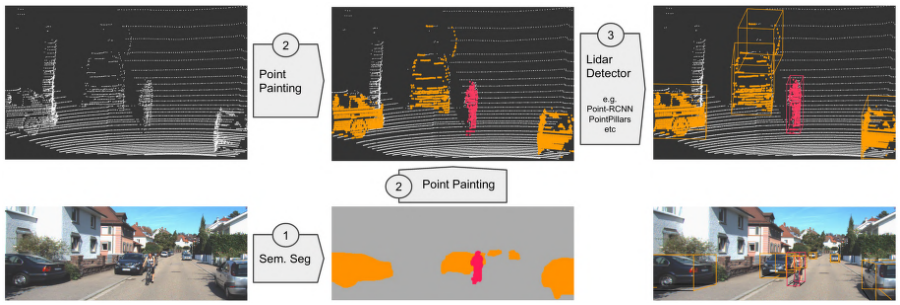


FIGURE 2.10: PointPainting pipeline, image taken from [37].

The Semantic segmentation network takes as input the image from the camera, and delivers per pixel class scores that work as summarized features of the image. The authors argue that some advantages of using semantic segmentation in the pipeline consist on the fact that per pixel classification is an easier task than 3D object detection. Furthermore, semantic segmentation networks are easier to train and are fast in inference time. Lastly, Vora *et al.* assure that PointPainting architecture is invariant to the design of the image segmentation network.

In the point painting section, the algorithm receives as inputs the Lidar point cloud L, the segmentation scores S, a homogeneous transformation matrix T and camera matrix M. With this information, first the points l in the 3D point cloud L are transformed and projected into the image M. Then for the corresponding pixel in the image, the segmentation scores S are extracted as s and concatenated along with the current point l being processed $[l, s]$.

Finally, the decorated point cloud is consumed by a Lidar network to predict the 3D bounding box. The authors demonstrate in the original work that their architecture can work with models such as PointPillars [15], VoxelNet [42], or PointRCNN [33].

It is possible to appreciate in this fusion method, a better cohesion between the features from one data modality to the other. Nevertheless, this method still depends on a sequential pipeline, where the risk of failing one of the stages compromises the final prediction.

2.5.3 Mid-level features fusion

The methods described in this section show different approaches to the ones mentioned before. Here, for each sensor and data modality features are extracted in parallel, and then processed together with attention blocks, allowing the models to attend to specific features and be robust in front of changing environment conditions and other sources of interference.

2.5.3.1 DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection [16]

In DeepFusion, Li *et al.* propose two techniques to address the alignment between features in the fusion stage, the first one is called *InverseAug* that inverse geometric related augmentations performed on the point cloud, enabling geometric alignment between Lidar and image data. The second technique is called *LearnableAlign* that leverages cross-attention mechanism to align the feature maps between Lidar and Camera data, as seen in Figure 2.11. Camera features are processed by a existing 2D feature extractor such as ResNet [11], and the point cloud is fed to a existing Lidar features extractor such as PointPillars [15]. The final task of this method is to predict 3D bounding boxes.

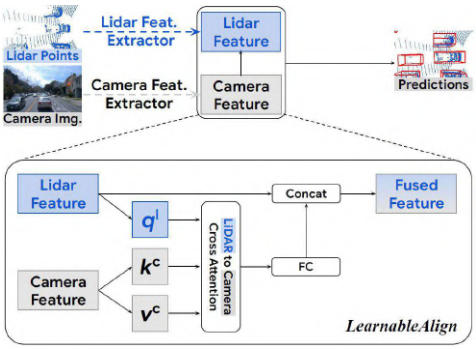


FIGURE 2.11: DeepFusion architecture, image taken from [16].

In *InverseAug*, Li *et al.* explain that in order to obtain higher performance on existing benchmarks, most of the time in training phase, models require important data augmentation to avoid overfitting. These augmentations lead to challenging alignments between the two data modalities. Then, the authors propose that after the data augmentation is applied to a 3D point cloud, for a single point the method saves the geometry-related augmentation parameters. Next, at the fusion stage it reverses all operations saved to get the original coordinate for the point and then finds its coordinates in the camera frame.

For a better alignment between Lidar and camera features, Li *et al.* present *LearnableAlign* that leverages cross-attention to capture the correlations between the two data modalities. This method uses three fully-connected layers to transform the point cloud features to the queries q^l, and camera features to the keys k^c and values v^c. The queries, keys and values are processed by the cross-attention block, then through a fully-connected layer, to be finally concatenated to the original Lidar features and sent to a 3D detection framework.

Even though this architecture explores deeply alignment methods for the different data modalities feature maps, their fusion strategy relies on using cross-attention blocks to fuse the features from each data stream. This, allows the model to dynamically capture the associations between the images and point cloud, attending to relevant information found in the representations.

2.5.3.2 TransFuser: Multi-Modal Fusion Transformer for End-to-End Autonomous Driving [24]

In this work, Prakash *et al.* find that geometric data alone is not sufficient to represent the global context of the scene. The main task is set to predict the future waypoints

for the local planning, based on the presence of traffic from various directions and global context of the scene.

TransFuser integrates image and Lidar representations, first by processing each data stream with a different ResNet architecture [11], then using attention blocks at different scales, where the intermediate features are fused, as seen in Figure 2.12.

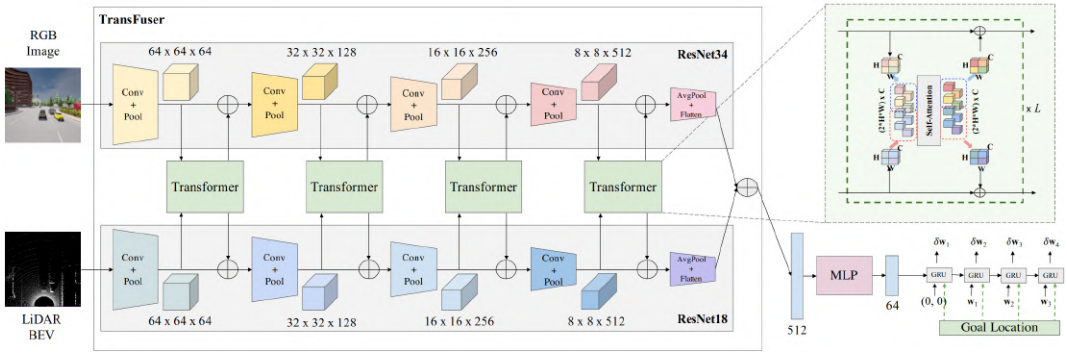


FIGURE 2.12: TransFuser architecture, image taken from [24].

Prakash *et al.* emphasize that camera-only methods perform unsatisfactory due to the lack the 3D information of the scene, while Lidar scans are normally very sparse, and require from other sensors to capture the missing information.

For the RGB image input, only the front-view camera is considered. The image is cropped to 256×256 pixels in order to remove radial distortion at the edges. For the Lidar input, the point cloud is converted in a 2D bird's-eye-view grid, for a fixed resolution of $32m \times 32m$, considering points only $32m$ in front and $16m$ for each side. The view is discretized in a histogram of 2 bins, representing points below and above the ground plane. The final Lidar input map is 256×256 pixels wide and two channels deep.

The previous input maps are fed to the model to two separated streams, each input backbone is a different ResNet version [11]; for the Lidar input it is a ResNet18 and for the image input it is a ResNet34, as seen in Figure 2.12. Both models process the inputs, bringing the information to different scales after a set of convolutions and pooling operations. At each scale, the intermediate feature maps are concatenated and sent to a self-attention block.

The authors emphasize that their main idea is to exploit the self-attention mechanism in order to combine image and Lidar modalities into the global context. The feature vectors are supplemented by a positional encoding. Next, with linear projections the feature vectors are transformed in a set of queries Q, keys K and values V. Then, the scaled dot product is used to compute the output [36]. The output fused features are fed back to the existing feature maps using element-wise summation.

After four scales, the last set of features are flattened to a 512 dimensional feature vector and then combined using element-wise summation. This compact tensor represent features combined from both data modalities. As a final step, the features are passed to a set of GRU cells that predict the final waypoints [5].

It is important to notice, that this method only considers what is in front of the vehicle and neglects the rest of its surroundings, information important for a whole representation of the global context. Nevertheless, as its task is focused on predicting the waypoints for the planner, its performance could improve at intersections, but in other scenarios (e.g. highways, roundabouts) it would drop as it lacks of vital information.

Chapter 3

Transformer-based Lidar-RGB fusion network for semantic grid prediction

In this chapter we present a sensor fusion approach for the semantic grid prediction approach for autonomous vehicles, based on the different methods and algorithms explored in chapter 2.

We first formalize the problem. Given a:

- Set of n images $\{I_k\}^{k=1...n}$ from the camera views.
- Set of Lidar returns $\{\mathbf{x}_1, \ldots, \mathbf{x}_P\}$, where P is the variable number of points and $\mathbf{x}_i \in \mathbb{R}^4$ represents the 3D location of each point and its intensity.
- The corresponding transformation matrices between the sensors.

the main objective is to predict a top-view segmentation of the scene.

This approach uses as input data from Lidar and multi-camera sensors in order to predict a semantic grid. The data from both modalities contains complementary information, nevertheless, is different in terms of structure, perspective and density. In consequence, fusing these modalities represents a challenging task.

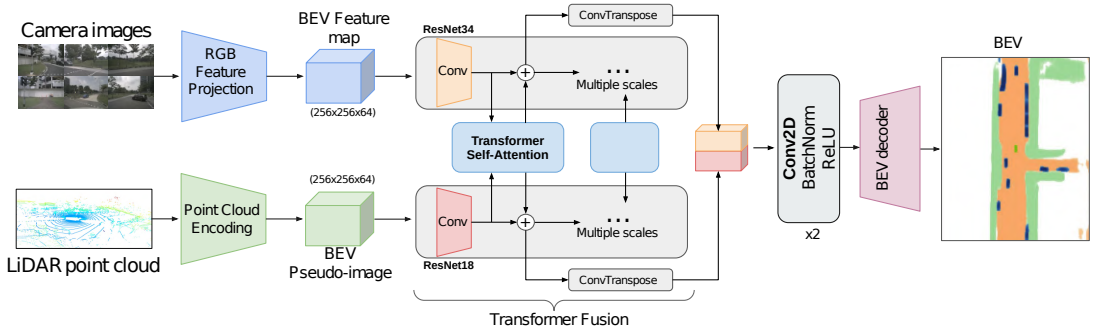


FIGURE 3.1: Proposed Transformer-based Lidar-RGB fusion architecture for semantic grid prediction.

To overcome this, as shown in Figure 3.1, we propose a network design with three main stages. In the first stage, both data modalities are processed and converted into a common representation of the scene. The RGB Feature Projector is used to process camera images and project features onto a top-view plane. In parallel, the 3D point cloud is fed to the Point Cloud encoding network that creates a

BEV pseudo image. In the second stage, both feature representations are fused in a multi-scale transformer-based network, technique seen in [24]. In the final stage, the intermediate features are up-scaled and then fed to a BEV Decoder module to predict the semantic grids.

3.1 RGB Feature Projection

From an arbitrary number of camera views, in order to transform the images onto a BEV representation, the proposed approach uses the recently presented model Lift-Splat-Shoot [23].

Similarly as in, the original work, in the proposed architecture the images from the multiple cameras are resized and cropped to a shape of 128×352 $(H \times W)$. Then, they are fed to a pre-trained EfficientNet-B0 [35], that is in charge of lifting the images from 2D coordinate system to a 3D frame. As explained in Section 2.2.1, in order to transform the images into the reference frame, it is necessary to find the depth associated to each pixel.

The solution presented in [23], shows a way to estimate all possible discrete depths D defined by $\{d_0 + \Delta, ..., d_0 + |D|\Delta\}$ for $d \in D$, where a large point cloud of size $D \times H \times W$ is created for a given image. Given a Δ that is the distance or step between the discrete depths, a minimum d_{min} and maximum d_{max} distances for which the depths are estimated, the total number of discrete depths D is calculated as:

$$D = \frac{d_{max} - d_{min}}{\Delta} \tag{3.1}$$

Then, for a pixel i the model predicts a context $c \in \mathbb{R}^C$ and a distribution over a depth $\alpha \in \Delta^{D-1}$. Consequently, the feature c_d that belongs to pixel i is defined as the context c for the same pixel scaled by α_d, as $c_d = \alpha_d \times c$.

The next stage transforms the point cloud created in the lift stage. Following PointPillars [15], the algorithm assigns each point to the corresponding nearest pillar. Then, is created a tensor of size $H \times W \times C$ using sum pooling, this tensor contains the feature representation from the multi-view cameras. The final size of the BEV Feature map is $256 \times 256 \times 64$ $(H \times W \times C)$, where C is the number of feature channels, as shown in Figure 3.2.

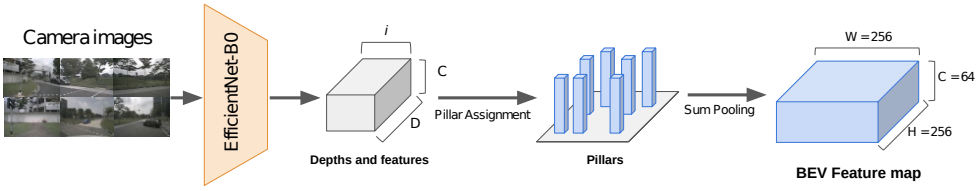


FIGURE 3.2: RGB Feature Projection pipeline.

3.2 Point Cloud Encoding

In order to encode the 3D point cloud input into a BEV Pseudo-image, the proposed architecture uses the Pillar Feature Net part of [15]. In a first stage, the model discretizes the 3D point cloud into a spaced grid in the BEV plane, where a set of bins or pillars are created containing the points.

Each point p in the pillars, with coordinates x, y, z and reflectance r, is augmented with x_m, y_m, z_m that denotes the distance to the arithmetic mean of all points, and with x_c, y_c that denotes the offset from the pillar x, y center. In total, the Lidar point is augmented to 9 dimensions $(D = 9)$.

Normally, due to the sparsity of the point cloud the pillars would be mainly empty, and the ones with points would have only a few of them. Then, in order to exploit the sparsity of the point cloud, the number of non-empty pillars (P) and the number of points per pillar (N) are limited to create a tensor of size $(D \times P \times N)$. Subsequently, if a pillar holds too many points to fit in the tensor the points are randomly sampled and removed. On the contrary, if a pillar does not have enough points to fill the tensor a zero padding is used.

In the next stage, a simplified version of PointNet is used [26], where each point is processed by a linear layer followed by Batch-Norm and ReLU in order to obtain a tensor $(C \times P \times N)$. Followed by a max operation over the channels a tensor $C \times P$ with the encoded features is created. Finally, the encoded features are scattered back to the original pillars and the BEV Pseudo-image is created, with a size of $(H \times W \times C)$.

The model proposed in this work is set with a maximum number of pillars $P = 10000$, and maximum number of points per pillar $N = 100$. The pillars are set in x and y from $-64m$ to $64m$, for a full range of $128m \times 128m$. With a resolution of $0.5m$, the size of the resulting BEV Pseudo-image canvas is $256 \times 256 \times C$, with features $C = 64$, as shown in Figure 3.3.

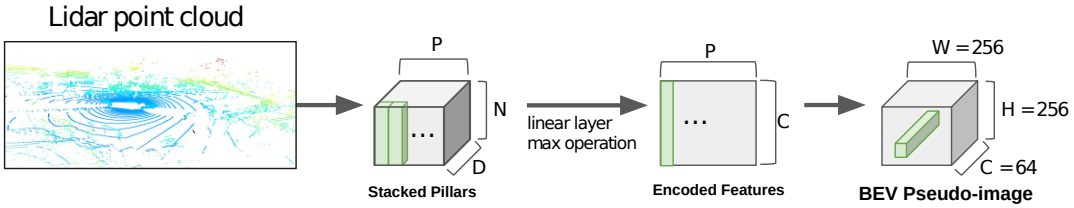


FIGURE 3.3: Point Cloud Encoding pipeline.

3.3 Multi-scale fusion with Transformers

Once the RGB Feature projector and point cloud encoding modules have created the feature maps, from the images and point cloud respectively, the next step in the proposed architecture is to fuse them at multiple scales with self-attention transformers, fusion technique seen in [24].

The main idea is to exploit the self-attention mechanism [36] to learn optimal features and global context for the point cloud and image modalities. Convolutional feature extractors are used to scale the representations streams from the cameras and Lidar streams, and then fed the resulting maps to the transformer modules.

The proposed architecture uses ResNet-34 and ResNet-18 as convolutional feature extractors for the RGB Feature projector and point cloud encoding representations, respectively [11]. Following a convolutional block, the intermediate features are complemented by a positional encoding before being fed to the self-attention block. This is a learnable parameter that allows the network to infer spatial dependencies between the feature vectors.

The transformer receives as input an intermediate feature sequence F_i, that with a linear projection computes a set of queries Q, keys K and values V, using M^Q, M^K and M^V weight matrices. Using the scaled dot-product [36], the attention is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.2}$$

where d_k is the dimension of K. Finally, the model uses a linear transformation and an element-wise summation with the input features F_i to compute the features $F_o = f(Att) + F_i$, representing the output of the transformer module and the returning of the features to the convolutional feature extractor, as seen in Figure 3.4.

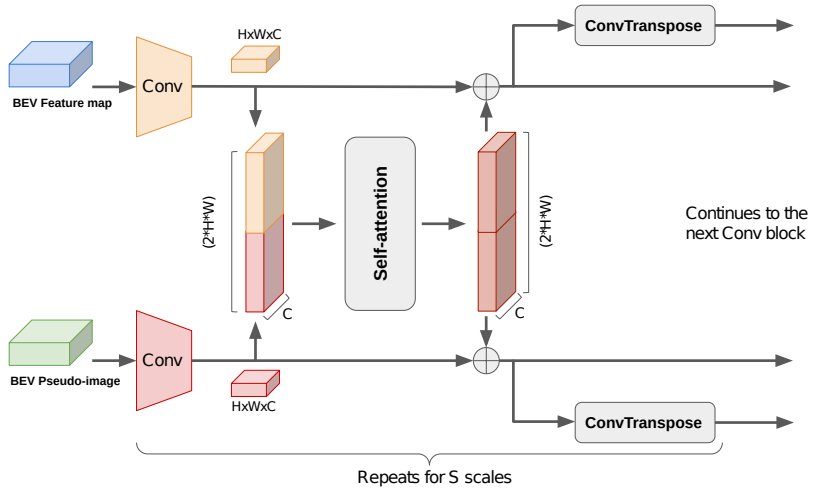


FIGURE 3.4: Multi-scale fusion with transformers.

With the ResNet architecture [11], the possible scales S where the intermediate features can be fused are four. At each scale, the feature depth C and the size of the feature map $(H \times W)$ change as:

$$(H \times W \times C) = \{(64 \times 64 \times 64), (32 \times 32 \times 128), (16 \times 16 \times 256), (8 \times 8 \times 512)\}$$

Following the element-wise summation at each different scale, the feature representations are up-scaled with a transpose convolution layer, which results on a feature map of size $256 \times 256 \times C$.

3.4 Semantic Grid Generation

The intermediate features obtained from the transformer are up-scaled with transpose convolutions layers for each scale, converting all feature maps to the same size $256 \times 256 \times C$. These feature maps are concatenated along C, for a total of $C = 384$.

Next, two convolutional blocks composed in sequence by Conv2D, BatchNorm and ReLU process the feature maps and outputs the final tensor for the BEV Decoder.

The BEV Decoder is composed by a series of ResNet-18 blocks [11], Convolutional and up-sampling layers. Initially a Conv2D, BatchNorm and ReLU process the input features, then feed them in sequence one after the other to Block-1, Block-2

and Block-3 of the ResNet-18 model. Finally, two more up-samplings and one last single convolution outputs the final semantic grid, as seen in Figure 3.5.

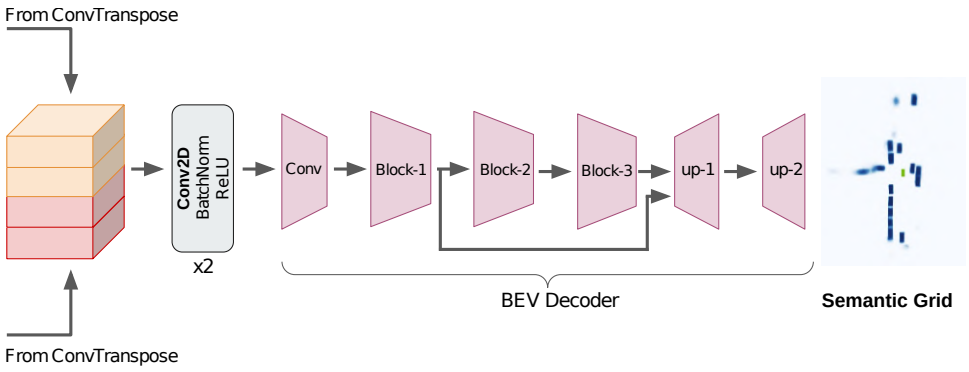


FIGURE 3.5: Semantic Grid Generation pipeline.

3.5 Training

For each semantic class $\in$ {`vehicle`, `drivable_area`, `walkway`, `lane_divider`} a model is trained. The training is performed using Binary Cross-Entropy (BCE) loss, as the problem presented is a binary classification (if a cell from the grid is part of the class or not). The BCE function is presented in Equation 3.3.

$$BCE(Y_N, X_N) = -\frac{1}{N}\sum_{n=1}^{N} y_n \cdot \log(x_n) + (1 - y_n) \cdot \log(1 - x_n) \tag{3.3}$$

Where x_n is the prediction, y_n is the ground truth and N is the batch size. In Equation 3.3 it is possible to appreciate that when the ground truth y_n is 1, the function adds $\log(x_n)$ the log probability of being the class to the loss, and when the label y_n is 0, the function adds the log probability of not being the class $\log(1 - x_n)$ to the loss.

Adam(adaptive moment estimation) is used as optimizer [14], an extension of stochastic gradient descent that instead of having a fixed learning rate, computes different adaptive learning rates from estimates of first and second moments of the gradients. This optimizer was set with an initial learning rate value $1e^{-3}$ and a weight decay $1e^{-7}$. The training batch size was set to 2 due to the memory requirements from the dataloader and model, and all models were trained for a minimum of 60 epochs. The PyTorch framework was used for coding the model [22], working mainly in the GPU nodes in Lyon cluster of Grid5000 [3]. The source code will be open-sourced at github repo `https://github.com/gsg213/TFGrid`.

3.6 Experimentation

In this section, we describe the dataset used for training and validation, the metrics used to evaluate the performance of the model, and the ablation studies performed in order to understand the effect of the different modules in the whole architecture.

3.6.1 Dataset

The dataset used for training and validation is nuScenes [4], a real-world dataset composed by 1000 scenes from Boston and Singapore that are well known for their high traffic density and different driving situations. Each scene is 20 seconds long and selected for the diversity and challenge in traffic and environment conditions.

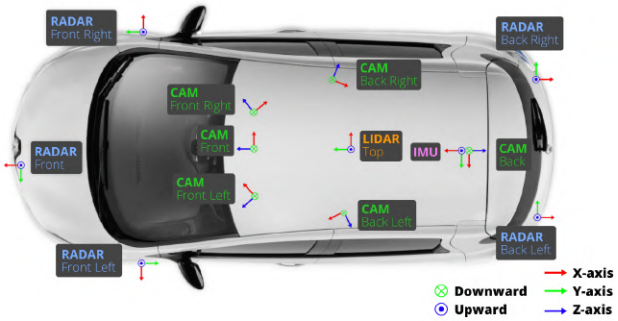


FIGURE 3.6: Car set up for nuScenes dataset, image taken from [4].

As seen in Figure 3.6, the car used for collecting the data is a Renault Zoe with 6 Cameras, 1 Lidar, 5 Radar, GPS and IMU. As each sensor has a different sampling time, in all scenes sensors are synchronized and the final sampling frequency is 2Hz. That is, 20 seconds scene has a total of 40 samples. The calibration of the sensors allows to obtain the transformation matrices between the different coordinate systems. In this work only Camera and Lidar data for `drivable_area`, `walkway`, `lane_divider` and `vehicle` classes are used.

Finally, with nuScenes it is possible to filter different conditions and situations within the scene, and in this work, Rain and Night scenarios are used to compare how the different models perform under these challenging conditions.

3.6.2 Metrics

The metric Intersection Over Union (IOU) is used to evaluate the model. With this metric it is possible to quantify the degree of overlap between the predicted semantic grid and the ground truth map from nuScenes. The intersection $\cap$ is defined as the area shared between two regions and union $\cup$ is the total area of the shape created by the two regions, as seen in Figure 3.7.

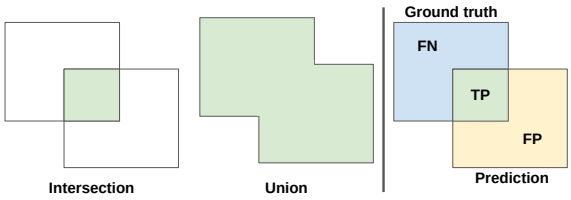


FIGURE 3.7: Intersection and union.

The IOU is formally defined as intersection divided by the union, as seen in Equation 3.4. As seen in Figure 3.7 intersection $\cap$ can be defined as the True Positives (TP) that are the cells that match the object in the prediction and ground truth, while union $\cup$ is defined as the sum of True Positives (TP) + False Positives (FP:

Positives from the prediction that are outside the ground truth) + False Negatives (FN: Negatives from the prediction that are inside the ground truth) cells from the prediction, as seen in Equation 3.4.

$$IOU = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \tag{3.4}$$

Finally, in order to evaluate how the model performs along all classes K, the mean IOU is computed based on the sum of all IOU calculated for each class k, and divided by the total classes K, as seen in Equation 3.5.

$$mIOU = \frac{1}{K} \sum_{k=1}^{K} IOU_k \tag{3.5}$$

3.6.3 Ablation study

In order to understand how each model affects the final result in the proposed architecture, some ablation studies are preformed to demonstrate how the sensor fusion with transformers helps to improve the semantic grid prediction. Two variants are trained; these are single-modality meaning that one relies on Camera data and the other relies on the Lidar information.

The first variant denominated LSS is Lift-Splat-Shoot [23], which uses only camera images. From this architecture two tests are performed, one is the evaluation on the pre-trained model given by Philion *et al.*, that is only trained for `vehicle`. The other is trained in this work for the classes `drivable_area`, `walkway`, `lane_divider` and `vehicle`, using the original code released by Philion *et al.*

The second single-modality variant is Pillar feature Net [15], that relies on the Lidar information. The Transformer Fusion module is removed and the features obtained from the Point Cloud Encoding are sent directly to the BEV Decoder.

Finally, in order to analyse the improvement by adding the transformer fusion, a series of models using the proposed architecture are trained using a number of transformers $T \in \{1, 2, 3, 4\}$. Other model is trained where no transformer fusion is done denominated as *concat*, due to the fact that the feature maps are only concatenated as fusion technique. In this model as seen in Figure 3.1, the Transformer Fusion section is removed and the features are fed directly to the BEV Decoder.

3.7 Quantitative Results

In the results presented in this section, the proposed architecture is called **TFGrid** as short name for **T**ransfomer-based Lidar-RGB **F**usion Network for Semantic **Grid** Prediction. Additionally, results presented for TFGrid and all models are from semantic grids cropped to a resolution of $100m \times 100m$, as the original model for LSS has this resolution, and in order to be compared all grids must have the same size.

Initially the ablation study is performed where the number of transformers changes, resulting in fusion at different scales as seen in Table 3.1. In this table, it is possible to appreciate the performance for classes $\in$ { `drivable_area`, `vehicle` } at night, in rain and in all conditions, as these categories are the most representative in the autonomous driving task.

This table shows that in all conditions the model with 2 Transformer modules outperforms the other models in both categories, being followed by the model with 4 Transformers really close in vehicle class. In the results for this class at night conditions the model with 1 Transformer performs better and for rain conditions the

model with 4 Transformers outperforms the others. In Drivable area category the model with 2 Transformers outperforms the other models at night, in rain and in all conditions.

TABLE 3.1: IOU results for different number of transformer fusion modules.

Model	Transformers	Vehicle	night	rain	Drivable area	night	rain
TFGrid	1T	34.48	**40.34**	35.62	76.68	68.41	69.72
TFGrid	2T	**35.88**	39.33	37.14	**78.77**	**70.45**	**69.74**
TFGrid	3T	34.38	37.56	34.39	72.55	64.67	63.79
TFGrid	4T	35.86	40.11	**38.71**	76.13	68.16	67.67

Table 3.2, shows the mIOU for the classes presented in Table 3.1. This measurement is essential to understand which one of the four models performs better, where the model with 2 Transformers outperforms the other models with $mIOU = 57.33$. In terms of inference time the fastest model is with 1 Transformer with a time of $47.90ms$, while the lowest model is with 4 Transformers and a time of $66.21ms$. The model with 2 Transformers has an inference time of $54.36ms$ that is translated to 18 Hz or 18 Frames per Second (FPS). These times were obtained on a single AMD Radeon Instinct MI50 GPU on the Grid5000 testbed [3].

TABLE 3.2: Inference time and mIOU results (on vehicle and drivable area classes) for different number of transformer fusion modules.

Model	Transformers	mIOU	Time (milliseconds)
TFGrid	1T	55.58	47.90
TFGrid	**2T**	**57.33**	54.36
TFGrid	3T	53.47	61.44
TFGrid	4T	56.00	66.21

Based on the previous results, the TFGrid model with 2 Transformers is compared in Table 3.3 with single-modality methods and with the same architecture but without transformer fusion. In Table 3.3, the best results are presented in **bold** and the second best results in blue.

For the Vehicle class, it is possible to appreciate how the proposed architecture outperforms the other models with IOU 35.88, being just followed by TFGrid using concatenation.

The same situation can be observed for the Drivable area class where TFGrid obtains IOU 78.87, being followed by TFGrid using concatenation with IOU 74.18.

A different result can be appreciated in the predictions for Lane divider, as the model with best results is LSS followed by TFGrid. This shows that the only-camera method has the ability to identify markings, and as TFGrid is using the same architecture for RGB Feature Projection, having lower results means that the Lidar features could be introducing noise in the prediction for this class.

Lastly, for the Walkway class TFGrid shows an improvement over the other methods, and is followed closely by LSS model.

Finally, the $mIOU$ is computed for all methods showing that TFGrid obtains the higher results $mIOU = 50.36$, followed by the TFGrid version using concatenation instead of transformers. This demonstrates that the fusion of different sensor modalities is able to improve in each class and in total the prediction of the semantic grid. Furthermore, it is possible to appreciate how the transformer sensor fusion helps to

increase even more the improvement in IOU, placing TFGrid above the concatenation model.

TABLE 3.3: Results on the nuScenes validation split. It is compared the IOU of the generated semantic grids for all models. Best results are presented in **bold** font, second best in Blue.

Models	Vehicle	Drivable area	Lane divider	Walkway	mIOU
LSS (pre-trained) [23]	32.80	-	-	-	-
LSS [23]	28.94	61.98	**37.41**	50.07	44.60
Pillar feature Net [15]	28.67	69.19	26.05	30.57	37.31
TFGrid (concat)	32.88	74.18	30.41	43.78	45.31
TFGrid	**35.88**	**78.87**	35.70	**50.98**	**50.36**

In Table 3.4, we show how the different models respond to adverse environmental conditions such as Rain and Night, being these challenging scenarios for the autonomous driving tasks. For the Vehicle and Drivable area classes, TFGrid demonstrates superior performance, followed by TFGrid using concatenation.

As expected from the results in the previous Table 3.3, LSS prediction values are higher for Lane divider class in both Rain and Night scenarios, followed closely by TFGrid. Remarkably, it can be seen that even if the Pillar feature Net has a low IOU result, the outcome of TFGrid is close to LSS demonstrating how both sensors modalities are complementary to each other. This shows the importance of using both instead of a single modality and fusing the features using transformer modules.

Lastly, for the Walkway class TFGrid is able to perform better at Night, nevertheless, LSS model results are higher in Rain conditions closely followed by TFGrid. This could be due to the low performance of the Point Cloud Encoder, that can be really affected under rain conditions, as the laser beams can intersect the raindrops and retrieve wrong information.

TABLE 3.4: Results on the nuScenes validation split. It is compared the IOU of all models in only **Night** conditions and only **Rain** conditions. Best results are presented in **bold** font, second best in Blue.

Class	Vehicle		Drivable area		Lane divider		Walkway	
Model	Night	Rain	Night	Rain	Night	Rain	Night	Rain
LSS (pre-trained) [23]	31.06	34.06	-	-	-	-	-	-
LSS [23]	28.41	30.81	49.13	56.86	**15.87**	**34.72**	24.56	**46.59**
Pillar feature Net [15]	28.54	19.56	63.89	64.01	3.60	23.72	23.52	27.22
TFGrid (concat)	37.89	33.52	68.98	67.25	7.27	27.90	23.60	40.98
TFGrid	**39.33**	**37.14**	**70.45**	**69.74**	15.17	30.25	**24.65**	44.26

3.8 Qualitative Results

The following Figures show the qualitative results, where it is possible to appreciate the accuracy in the semantic grids created by each model. In green circles, we highlight details not seen by the other models; in red, we show the regions where the prediction is missing information. In all predictions the ego-vehicle is shown in the center of the grid as a small green square. Predictions are oriented in a way that the top of the image is the front of the ego-vehicle.

In Figure 3.8 the class vehicle is predicted, and almost instantly in the first row is possible to observe how TFGrid is superior and able to find occluded vehicles that LSS is missing. Furthermore, is able to draw divisions between cars much better than the other models, showing the improvement of fusing camera and Lidar

modalities with transformers. The second row also shows TFGrid's predictions outperforming the other models. Nevertheless, there is a car that is missing across all models as is totally occluded by other vehicle. For more vehicle predictions refer to Appendix A.1.

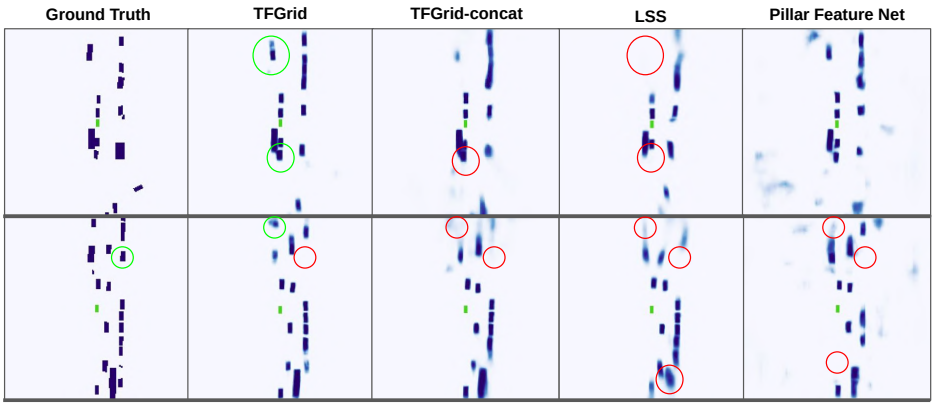


FIGURE 3.8: Predictions for Vehicle class.

In Figure 3.9 Drivable area predictions are shown. In the first row, TFGrid shows outstanding performance compared with the other models, as LSS and Pillar Feature Net predictions show large regions of noise and road discontinuity. In the second road, it is possible to observe that LSS prediction has trouble drawing the road after a turn creating a large region, while TFGrid shows a good estimation of the road even if part of the road is occluded by the building corner. This is possible thanks to the geometric information from the Lidar, indicating a potential direction for the continuing road. For more Drivable area predictions refer to Appendix A.2.

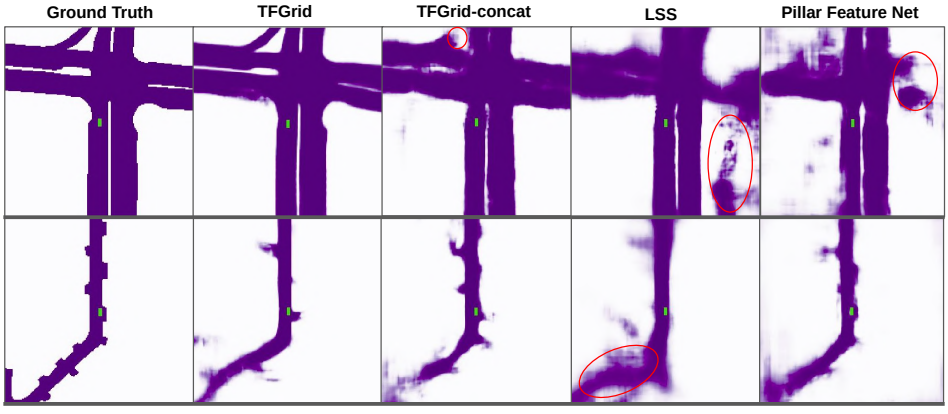


FIGURE 3.9: Predictions for Drivable area class.

Walkway predictions are shown in Figure 3.10. In the first row is possible to appreciate how poorly Pillar Feature Net performs compared with the other models. LSS alone creates a better defined representation of the scene, although the best results are presented by TFGrid demonstrating the improvement of using transformers for the camera and Lidar fusion. In the second row, it is possible to notice the detailed predictions that TFGrid can generate, with less noise than the other models. For more Walkway predictions refer to Appendix A.3.

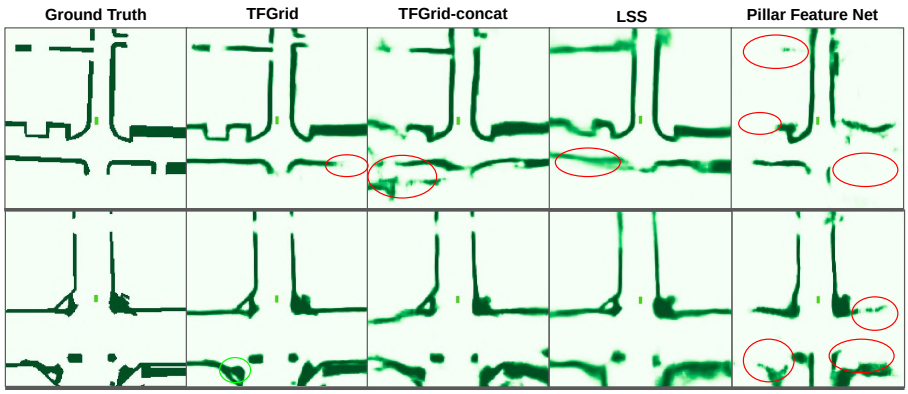


FIGURE 3.10: Predictions for Walkway class.

In Figure 3.11 are presented the predictions for Lane divider. In the first row, TFGrid shows better definition on the top lane lines than the other models. Nevertheless, LSS shows accuracy predicting the length of the lanes in the bottom, while for Pillar Feature Net is most difficult to identify the lanes on the crossing road.

The second row shows better results for LSS in definition and lane longitude, nevertheless, TFGrid is able to find further lanes and manages to better define them. For more Lane Divider predictions refer to Appendix A.4.

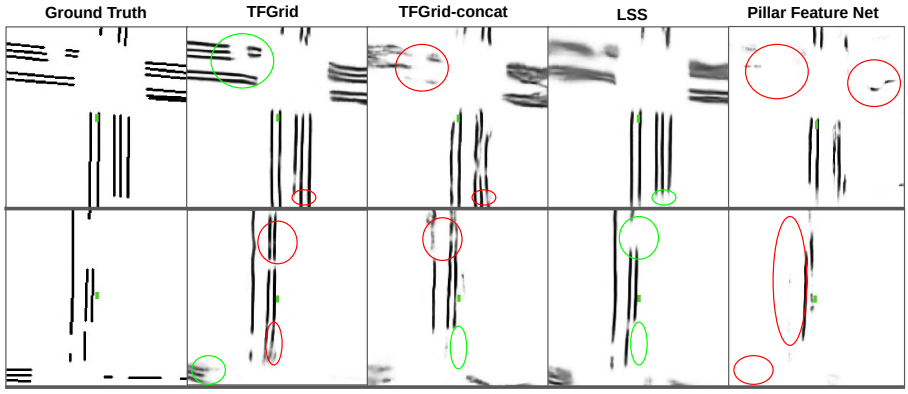


FIGURE 3.11: Predictions for Lane divider class.

Finally, an example from TFGrid with all classes combined in just one semantic grid map is presented in Figure 3.12. More predictions can be found in Appendix B.

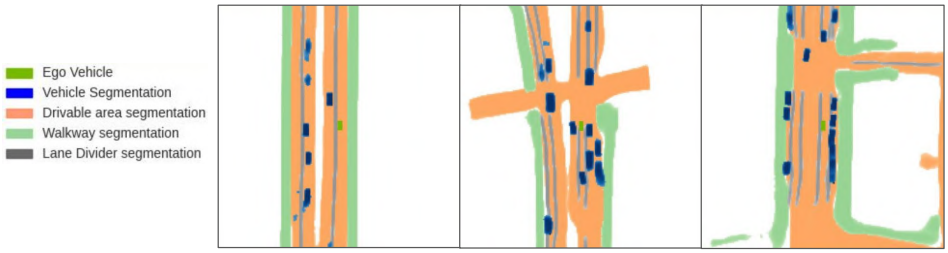


FIGURE 3.12: Combined TFGrid predictions for all four classes.

Chapter 4

Conclusions

4.1 Summary

In this work, we presented a Transformer-based Lidar-RGB fusion network for semantic grid prediction. The proposed architecture processes Camera and Lidar inputs with RGB Feature Projection and Point Cloud Encoding modules, feeding the intermediate features to the Transformer block. Finally, fusion takes place at multiple scales with self-attention, where the model learns the optimal features, which the BEV Decoder uses to predicts the semantic grid. This architecture demonstrates superior results compared with camera-only and Lidar-only models for Vehicle, Drivable area and Walkway classes.

Furthermore, the presented model is able to outperform a naive fusion method that simply concatenates intermediate feature maps. This highlights the improvement introduced by the multi-scale transformer-based fusion architecture.

We performed ablation studies in order to understand how each module of the architecture contributes to the final prediction. Additionally, we perform these studies to analyse the number of transformer modules and scales at which the model performs the best.

The proposed model does not reach the same performance as the camera-only method for the Lane divider class. After analysing the quantitative and qualitative results, we hypotesize that Lidar features could be introducing noise from different objects in the scene occluding the Lane markings. Also, the fact that due to the sparse nature of the point cloud could not be returning enough points.

Based on the results, we found that due to the complementary type of data between camera and Lidar, models using both modalities are more robust for semantic grid predictions. This is demonstrated for example in the Drivable area predictions, where even when the road takes a turn behind a building, the model estimates the road direction thanks to the geometric data from the Lidar. Also, in the Lane divider class, the model presented is able to find the Lanes, even when the Lidar returns do not contribute as much information as the camera images. Thus, the attention mechanism combined with multi-scale features have shown significant results, attending to the important features from both modalities improving the performance for the proposed model.

Finally, concurrently with this work, new approaches for semantic grid prediction with camera-lidar sensor fusion have been presented. Such as BEVFusion presented by Liu *et al.*[19], that uses convolution modules to process features from both modalities, obtaining state-of-the-art results. Additionally, new works aim to improve camera-only methods to predict semantic grids, as seen in [32, 40, 17, 39]. This demonstrates the relevance of semantic grids in the autonomous vehicles, and of the work presented in this thesis.

4.2 Future work

Initially, as continuation to the work presented, we plan on training and testing the proposed architecture with more classes so as to increase the difficulty level of detection. Classes such as pedestrians or movable objects in the road, could lead to a better understanding of its response in different situations.

As seen in the qualitative results, in a common scenario vehicles in movement can occlude other vehicles or objects of interest in the road removing them from the semantic grid. In order to continue with the work presented, the predictions can be improved by processing past semantic grids or feature maps and fusing them with the current data from the environment. This could lead objects of interest occluded can be predicted by their previous position, even if the sensors cannot reach them.

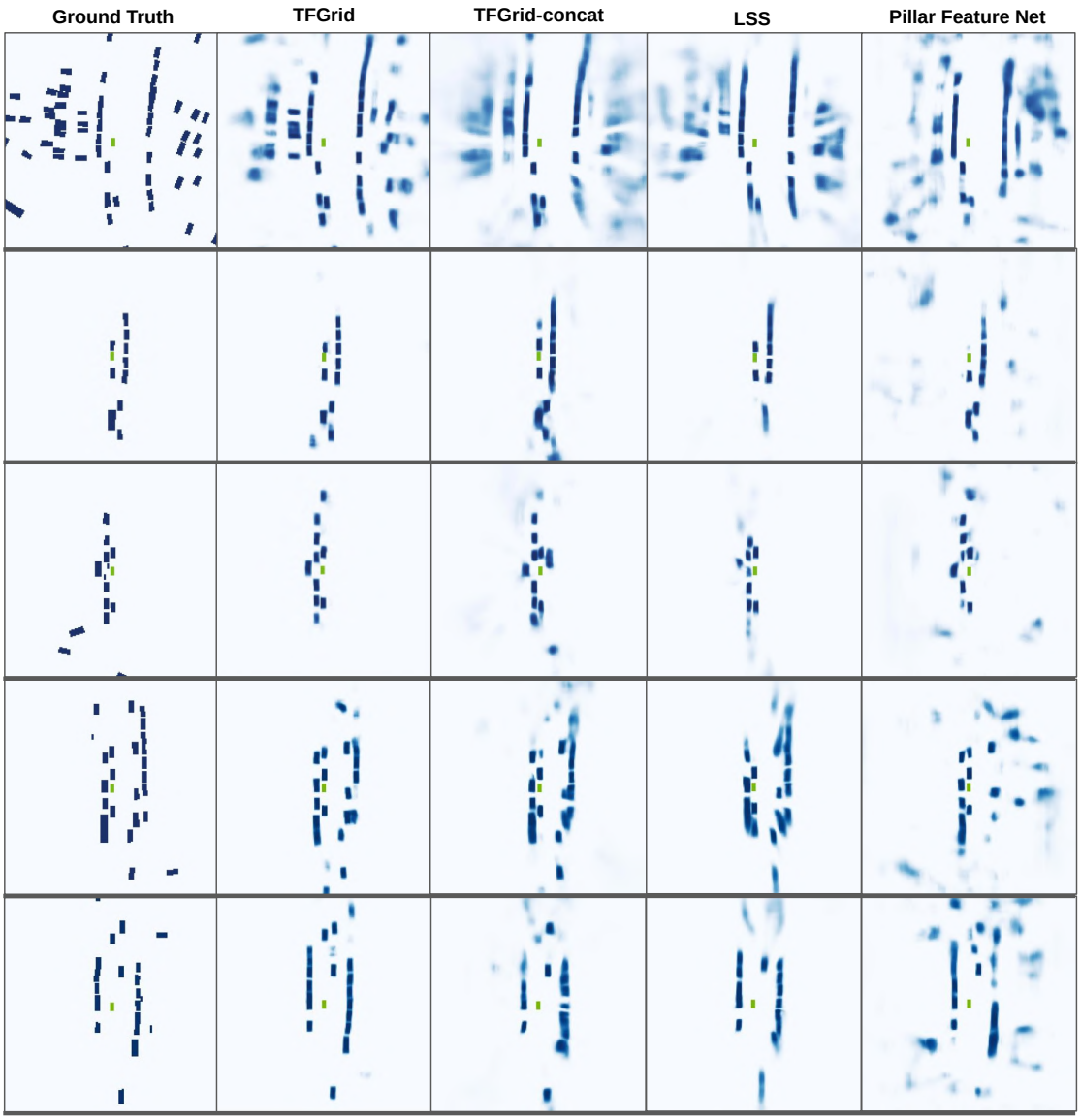
Given the sparse nature of point clouds compared with images, to improve the detections and the features from the Point Cloud Encoder, the input can be augmented adding past point clouds transformed to the reference frame of the current time instant. This would allow denser point clouds with more returns from the surrounding objects.

Finally, explore different transformer-based architectures for the multi-modal fusion, and comparing them with the one proposed in this work. However, transformers have a high computational and memory footprint. Part of the future work will be on how to overcome those limitations to handle large amounts of input data (point cloud and multi-camera views, at potentially multiple time steps). Additionally, add more tasks to the model as obstacle detection and tracking, providing the necessary information for motion forecasting models.

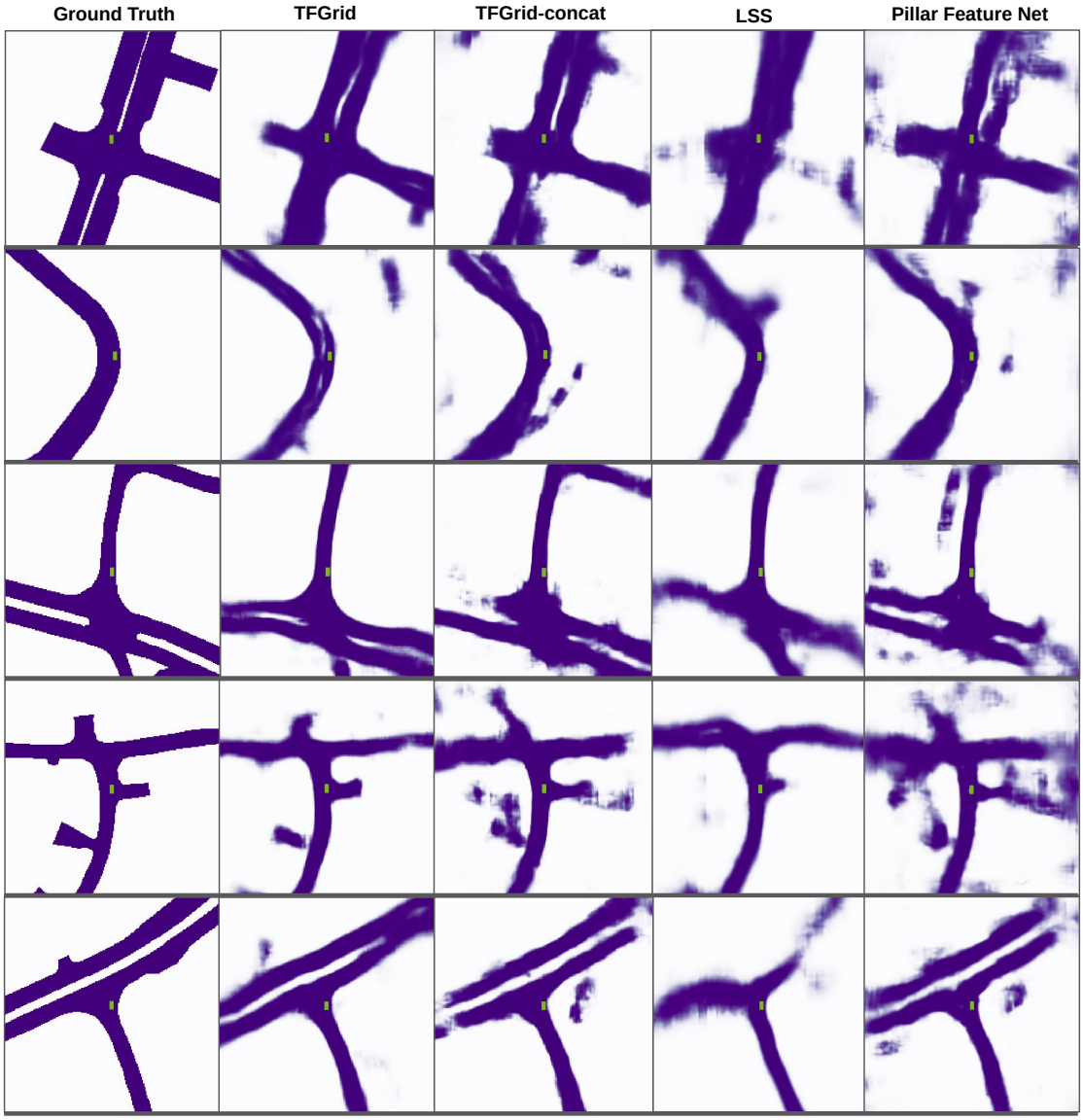
Appendix A

Semantic grid predictions per class

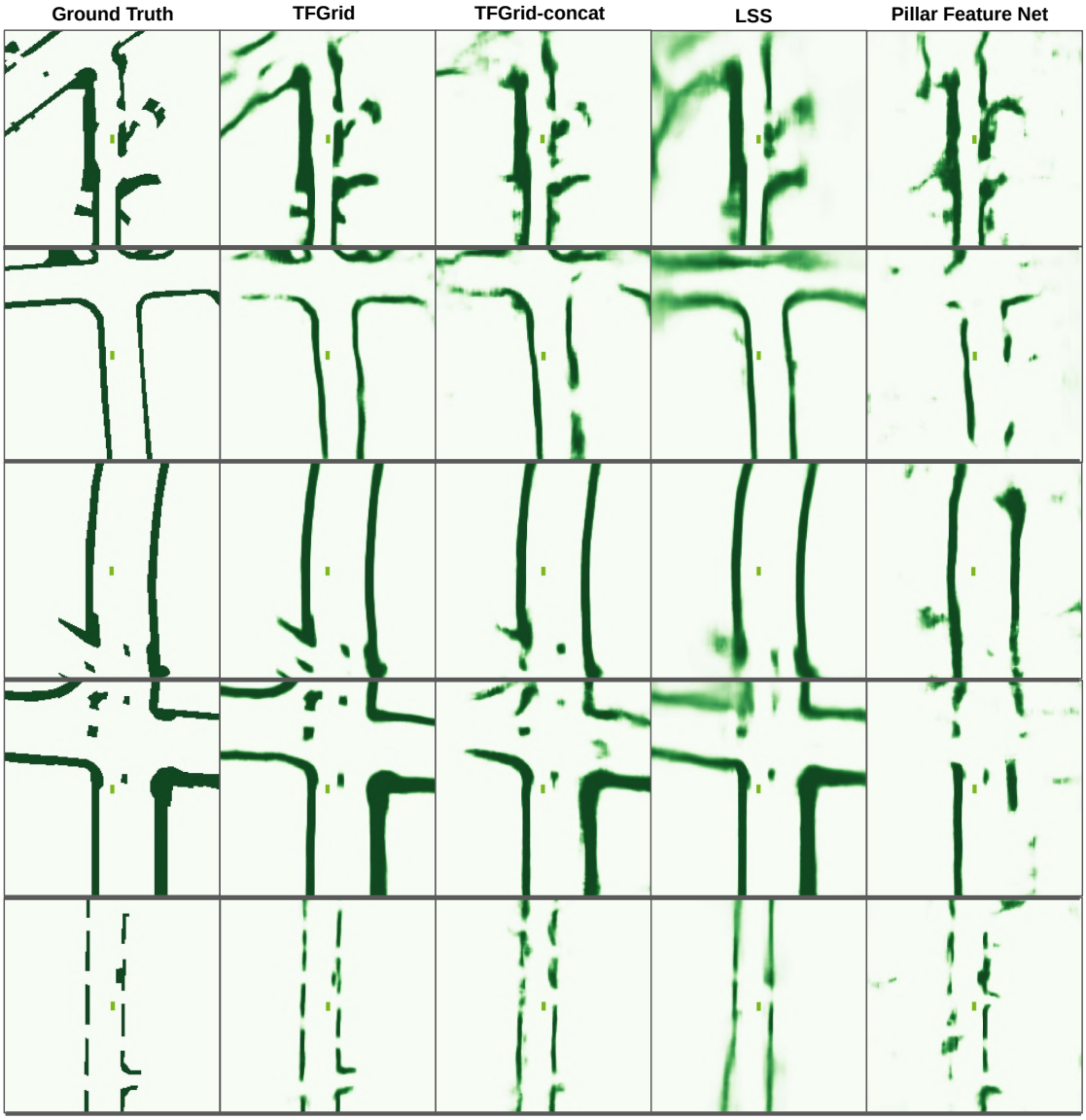
A.1 Vehicle Class



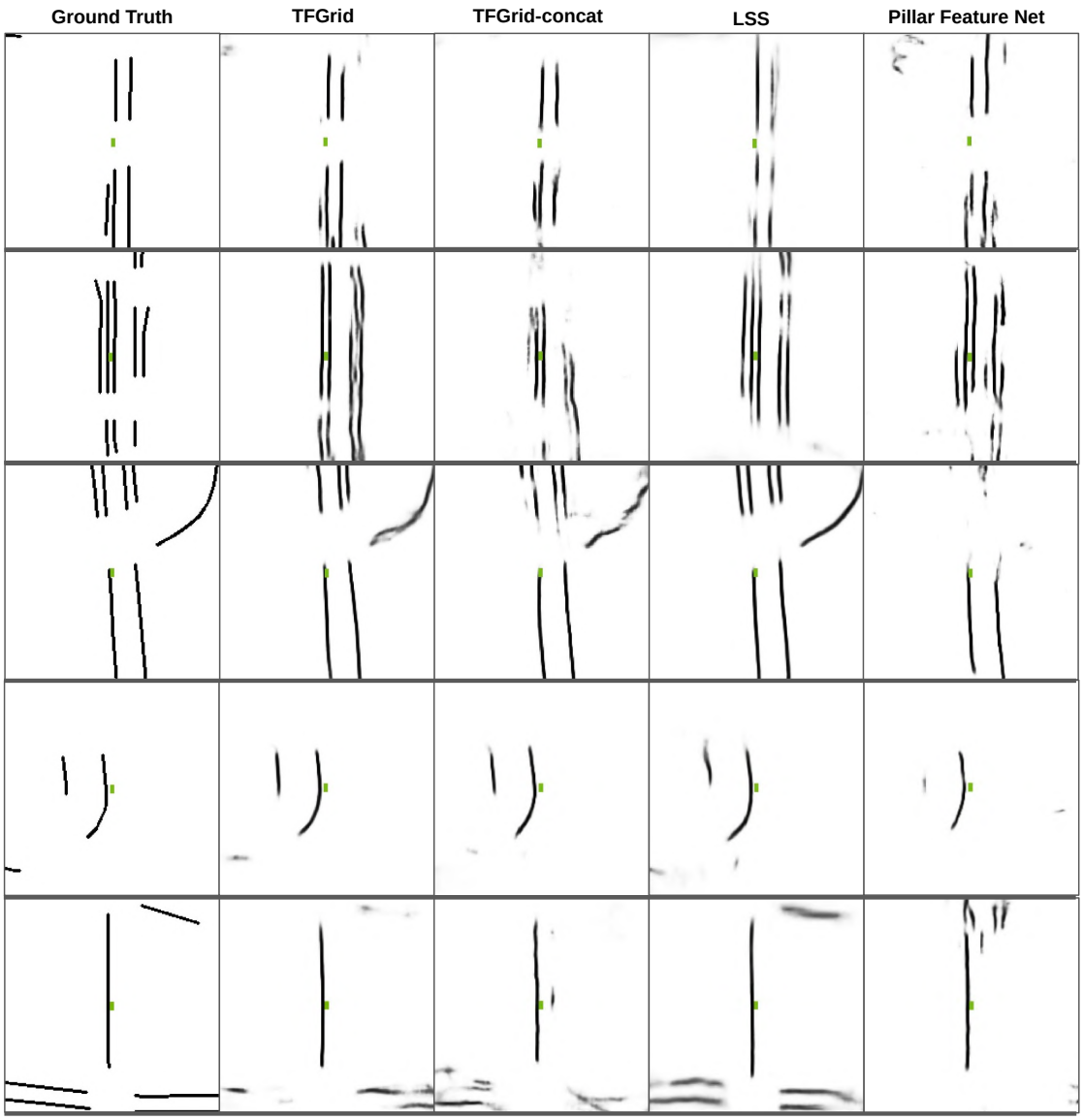
A.2 Drivable area Class



A.3 Walkway Class



A.4 Lane Divider Class



Appendix B

Semantic grid predictions with all classes combined

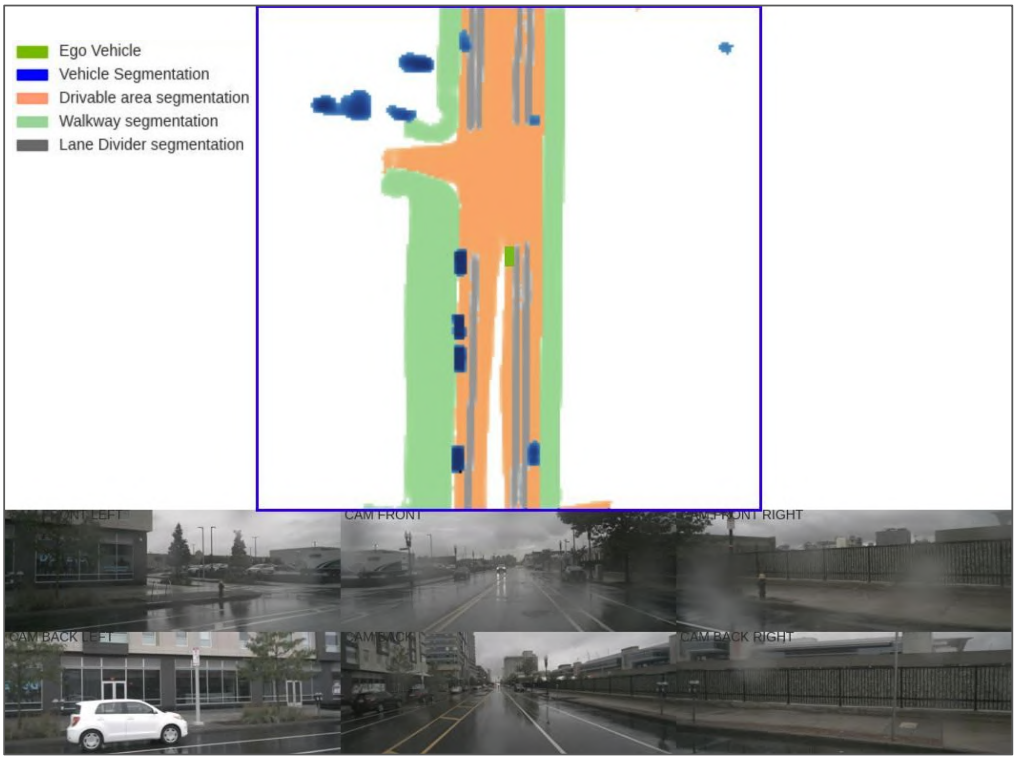
In this appendix are shown some examples of the top-view representations generated by the presented model a Transformer-based Lidar-RGB fusion network for semantic grid prediction. Each prediction has associated the images from the multiple cameras, where it can be appreciated the vehicles, road, walkway and lanes from the scene. Furthermore, some examples show the predictions in challenging scenes as Night and Rain.

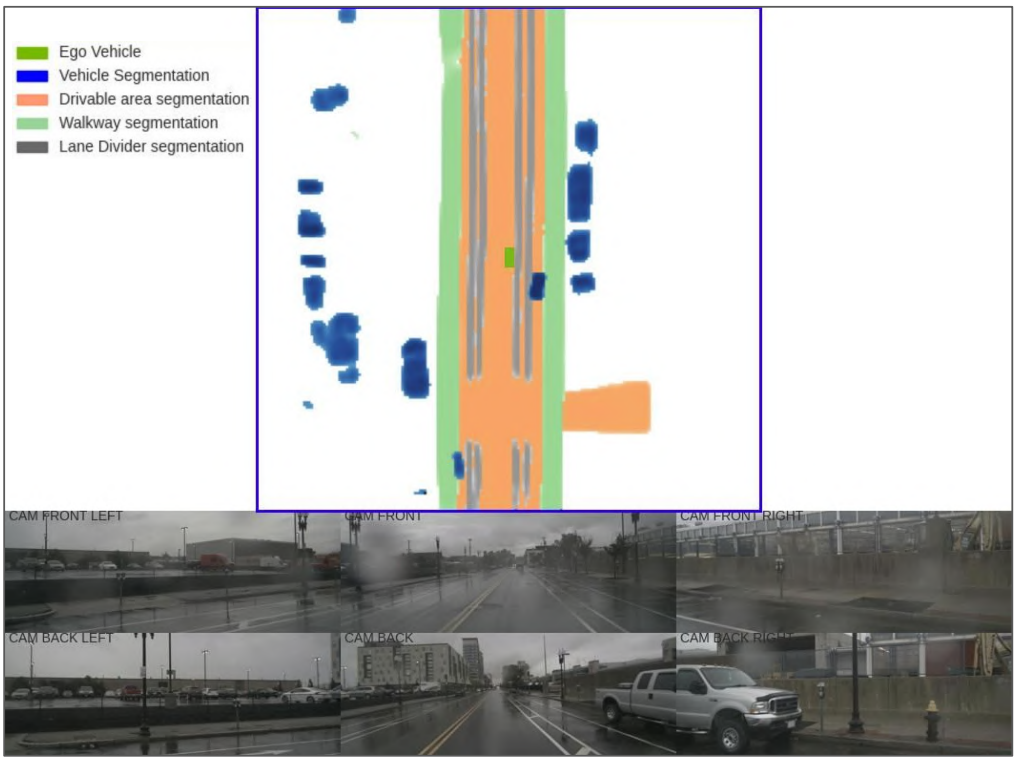
B.1 Semantic grids combined

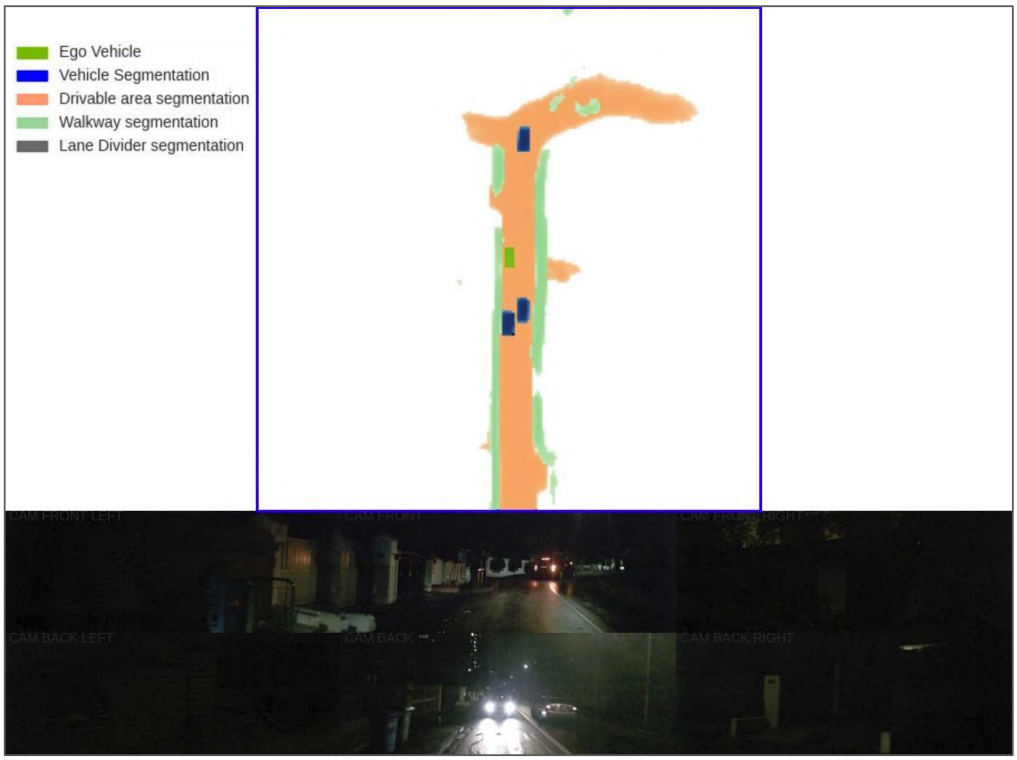

Ego Vehicle
Vehicle Segmentation
Drivable area segmentation
Walkway segmentation
Lane Divider segmentation


Ego Vehicle
Vehicle Segmentation
Drivable area segmentation
Walkway segmentation
Lane Divider segmentation
CAM BACK LEFT

B.2 Semantic grids combined at Night and Rain scenarios






Ego Vehicle
Vehicle Segmentation
Drivable area segmentation
Walkway segmentation
Lane Divider segmentation
CAM FRONT LEFT
CAM FRONT
CAM FRONT RIGHT
CAM BACK LEFT
CAM BACK
CAM BACK RIGHT


Ego Vehicle
Vehicle Segmentation
Drivable area segmentation
Walkway segmentation
Lane Divider segmentation
CAM FRONT LEFT
CAM FRONT
CAM FRONT RIGHT
CAM BACK LEFT
CAM BACK
CAM BACK RIGHT

Bibliography

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[2] Xuyang Bai et al. "TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection With Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 1090–1099.

[3] Daniel Balouek et al. "Adding Virtualization Capabilities to the Grid'5000 Testbed". In: *Cloud Computing and Services Science*. Ed. by Ivan I. Ivanov et al. Vol. 367. Communications in Computer and Information Science. Springer International Publishing, 2013, pp. 3–20. ISBN: 978-3-319-04518-4. DOI: 10.1007/978-3-319-04519-1_1.

[4] Holger Caesar et al. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[5] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[6] A. Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989), pp. 46–57. DOI: 10.1109/2.30720.

[7] Özgür Erkent et al. "Semantic Grid Estimation with a Hybrid Bayesian and Deep Neural Network Approach". In: *IROS 2018 - IEEE/RSJ International Conference on Intelligent Robots and Systems*. Madrid, Spain: IEEE, Oct. 2018, pp. 1–8. URL: https://hal.inria.fr/hal-01881377.

[8] Özgür Erkent et al. "Semantic Grid Estimation with a Hybrid Bayesian and Deep Neural Network Approach". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 888–895. DOI: 10.1109/IROS.2018.8593434.

[9] Juncong Fei et al. "PillarSegNet: Pillar-based Semantic Grid Map Estimation using Sparse LiDAR Data". In: *IEEE Intelligent Vehicles Symposium (IV): 11-17 July 2021, online*. 32nd IEEE Intelligent Vehicles Symposium. IV 2021 (Online, July 11–17, 2021). Institute of Electrical and Electronics Engineers (IEEE), 2021, 838–844. ISBN: 978-1-72815-394-0. DOI: 10.1109/IV48863.2021.9575694.

[10] Andreas Geiger et al. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[11] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[12] Noureldin Hendy et al. "Fishing net: Future inference of semantic heatmaps in grids". In: *arXiv preprint arXiv:2006.09917* (2020).

[13] Lukas Hoyer et al. “Short-term prediction and multi-camera fusion on semantic grids”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[15] Alex H Lang et al. “Pointpillars: Fast encoders for object detection from point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705.

[16] Yingwei Li et al. “Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17182–17191.

[17] Zhiqi Li et al. “BEVFormer: Learning Bird’s-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers”. In: *arXiv:2203.17270* (2022).

[18] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[19] Zhijian Liu et al. “BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird’s-Eye View Representation”. In: *arXiv preprint arXiv:2205.13542* (2022).

[20] Chenyang Lu, Marinus Jacobus Gerardus van de Molengraft, and Gijs Dubbelman. “Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 445–452.

[21] Bowen Pan et al. “Cross-view semantic segmentation for sensing surroundings”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4867–4873.

[22] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[23] Jonah Philion and Sanja Fidler. “Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D”. In: *Proceedings of the European Conference on Computer Vision*. 2020.

[24] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. “Multi-Modal Fusion Transformer for End-to-End Autonomous Driving”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 7073–7083. DOI: 10.1109/CVPR46437.2021.00700.

[25] Charles R Qi et al. “Frustum pointnets for 3d object detection from rgb-d data”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 918–927.

[26] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[27] Thomas Roddick. “Learning Birds-Eye View Representations for Autonomous Driving”. PhD thesis. University of Cambridge, 2021.

[28] Thomas Roddick and Roberto Cipolla. “Predicting Semantic Map Representations From Images Using Pyramid Occupancy Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[30] Lukas Rummelhard, Amaury Nègre, and Christian Laugier. "Conditional Monte Carlo Dense Occupancy Tracker". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 2485–2490. DOI: 10.1109/ITSC.2015.400.

[31] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[32] Avishkar Saha et al. "Translating images into maps". In: *ICRA 2022*. 2022. URL: https://www.amazon.science/publications/translating-images-into-maps.

[33] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. "Pointrcnn: 3d object proposal generation and detection from point cloud". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 770–779.

[34] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[35] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[36] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[37] Sourabh Vora et al. "Pointpainting: Sequential fusion for 3d object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4604–4612.

[38] Chunwei Wang et al. "PointAugmenting: Cross-Modal Augmentation for 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11794–11803.

[39] 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).

[40] Yunpeng Zhang et al. "BEVerse: Unified Perception and Prediction in Birds-Eye-View for Vision-Centric Autonomous Driving". In: *arXiv preprint arXiv:2205.09743* (2022).

[41] Brady Zhou and Philipp Krähenbühl. "Cross-view Transformers for real-time Map-view Semantic Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13760–13769.

[42] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499. DOI: 10.1109/CVPR.2018.00472.