ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING

Ciclo 38

**Settore Concorsuale:** 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
**Settore Scientifico Disciplinare:** ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

# Label-Efficient Bird's Eye View Perception from Monocular Cameras


*Presentata da:*
Henrique Piñeiro Monteagudo

*Coordinatore Dottorato:*
**Paola Salomoni**

*Supervisore:*
**Samuele Salti**

*Co-supervisori:*
**Francesco Sambo**
**Leonardo Taccari**
**Luigi Di Stefano**


---

Esame finale anno 2026

# Abstract

Assisted and automated driving systems have the potential to help prevent traffic accidents, which remain a leading cause of death worldwide. Vision-centric perception, due to its low cost, can unlock the deployment of these technologies at scale, amplifying their impact. This thesis, developed in collaboration with Verizon Connect, a company in the field of connected vehicles, studies perception using bird's eye view representations extracted from monocular cameras under real world constraints. These bird's eye view representations contain most of the information necessary for many tasks in driving and robotics while being compact, which makes them a common choice in the field. However, training machine learning models to produce them under full supervision requires expensive acquisition and annotation of three-dimensional data.

To deal with a lack of annotated three-dimensional data for our use case, we propose the first method to train bird's eye view semantic segmentation networks without direct bird's eye view supervision. We introduce two variations of this method based on neural density fields and monocular depth estimation networks and we validate it on standard benchmarks such as the nuScenes and the KITTI-360 datasets, beating baselines and showing its competitiveness as a pretraining methodology when some bird's eye view labels are available.

Standard benchmark datasets used as benchmarks in bird's eye view perception are acquired under very controlled conditions and lack the diversity observed in some real world applications e.g., in terms of camera mounting positions and viewpoints. Motivated by this example, which we observe in Verizon Connect's data, we introduce a new synthetic dataset with diverse viewpoints. We study the effect of these viewpoint variations in a bird's eye view semantic segmentation network trained on a single viewpoint, observing a decrease in performance. We evaluate a baseline approach combining several viewpoints at training time, observing a less pronounced decrease, suggesting that diverse training sets can help alleviating the impact of the viewpoint shift.

# Acknowledgments

There are so many people who have encouraged and supported me through carrying out this thesis and I'm very grateful to all of them: without you it wouldn't have been possible. If you ever spent some time with me during these years, taught me something, reviewed my work or just listened to me when things were not working out, I sincerely thank you, even if you are not explicitly mentioned here. As to raise a child, it takes a village to raise a PhD, and I strongly believe the social and community aspects are fundamental parts of the process even if it is, at times, a very lonesome endeavor.

I am immensely grateful to the two people who spent the most time – a resource I know was pretty scarce for both of them – supervising me during these years: Samuele Salti and Leonardo Taccari. This was not an easy project to supervise: lots of topics to learn and juggle in just three years. Regardless, you were always there for me and tried to take the best out of me, when the camera rotations, the coordinate system transforms and the renderings were working and when they were not. I thank you for all that you have taught me and for doing it while being great people. I would also like to give a special thank you to Francesco Sambo, who received me in Connect with open arms from the first moment, and always made sure things were smooth on all ends.

During these PhD years the place where I spent the most time has been, by a far margin, Verizon Connect's office in Florence. I would like to thank all the people with whom I worked and who taught me so much during this time, not only about science and engineering, but also about life, the Italian language and culture, or some unique way of cursing specific to Florence. Thanks to Aurel Pjetri, who helped me integrate from day one and was my mate on the day to day during these three and a half years. Thanks to Tommaso Bianconcini and Paolo Raiconi, who made me feel at home from the first day and always put a smile on my face. Thanks to Riccardo Turra, who was the first master's student I've ever supervised (and is now a colleague) during his internship, our work together is part of this thesis. A special thank you to Tomaso Trinci, with whom I've shared so many days at the office, and always has the patience to listen to my endless ramblings. Thanks to everyone I collaborated with and to everyone that made my time here

better: Matteo, Andrea, Niccolò, David, Ewa, Douglas, Davide, Stefano, Marco, Alessio, Chiara, and everyone else.

I would like to thank all my fellow Early Stage Researchers (and associates) in the SMARTHEP network: Carlos, Daniel, Danielle, Fotis, Jamie, Joachim, Kaare, Laura, Leon, Max, Micol, Patin, Sofia and Sten, it was a wild ride and sharing it with you was stimulating and very fun. Thanks to everyone who participated in the network and made it possible, specially to Caterina Doglioni, who coordinated it flawlessly, hosted me in Manchester and got me initiated in the analysis of high energy physics data.

I would also like to thank all the teachers and peers who motivated me to pursue this career. Thank you to all of my professors in the master's in computer vision who transmitted their interest in the subject. A special thank you to Pedro Arias and Joaquín Martínez who gave me my first opportunities at the University of Vigo. Thank you to Irea, you were there at the beginning of this and your example led me to pursue scientific work.

Finally, I would like to thank all of my family and friends back home in Galicia, who cheered for me during these years, always welcomed me back, came to visit and made me feel loved even from far away. A special thanks to my parents who gave me everything, provided me with an education that made me curious and critical and were always very supportive. Thanks to my sister Rosalía and my brother Lucas who are fundamental pillars in my life. Thanks to all my friends from Santiago and a special thank you to Fernando, Fernanda and Iago, I feel very grateful to have you in my life.

Grazas a todos! Grazie a tutti! Thank you all!

# Funding Acknowledgments

This work is part of SMARTHEP which received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement n. 956086

# Contents

# List of Figures

# Chapter 1

# Introduction

Road accidents remain a leading cause of death worldwide, claiming approximately 1.19 million lives annually [1]. Moreover, road accidents also cause a significant number of injuries. In addition to great human suffering, these accidents have a broad economic impact. In the United States of America the economic cost amounted to 1.6% of its gross domestic product in the year 2019 [2]. In the European Union (EU) 20,400 lives were lost to traffic accidents in 2023 [3]. In the same period 43,273 died in the United States from that cause [4]. To reduce this amount, the automotive industry, pressured by regulators, is pursuing automated safety systems. Advanced driver-assistance systems (ADAS) features especially designed for safety purposes such as forward collision warning and automatic emergency braking have been shown to reduce front-to-rear crash rates and front-to-rear injury crash rates [5]. Automated driving systems which aim to substitute human drivers also have great potential in this regard. Early data from totally automated (Level 4) driving systems deployed by Waymo as robotaxis show a reduction in vehicle crash rates across 11 crash types compared to human drivers [6], albeit only in the very limited geographical areas where they operate.

At the core of these automated systems is perception, the ability to build a robust understanding of the environment from sensor data. Although many of these systems currently rely on expensive sensor suites (including several units of sensors such as lidar and radar), the ubiquity and low cost of cameras make vision-based perception an important area of research. Vision-centric perception allows

for cheaper deployment and thus wider reach and enables redundancy in safety-critical applications. A challenge in this domain is creating representations of the 3D world that are suitable for downstream reasoning tasks like collision avoidance. This thesis focuses on the *bird's eye view* (BEV), an orthographic projection of the environment that acts as a powerful intermediate representation for automated and assisted driving.

## 1.1 Motivation

This PhD project was developed while working at Verizon Connect, a company specialized in software for the connected vehicles domain. One of their offerings, the most relevant for this thesis, is a video product. Video cameras are placed on the dashboard of customers' vehicles (these types of cameras are commonly referred to as *dashcams*) and capture footage from the vehicle's surroundings. One of the functions of this camera is to analyze the incoming footage and issue safety alerts so that the vehicle driver can respond accordingly. Enhancing the capabilities of these cameras and their future versions is one of the main motivations of this thesis. This particular application and hardware configuration provide a set of constraints and specifications that guide and ground the work in this thesis, and also differentiate it from concurrent and past work in the topic. We summarize this in three main points that will be recurring themes throughout this dissertation; first, we focus on *perception* rather than action tasks, as the devices we are interested in are add-ons that do not have the capability to operate the vehicles they are installed in. Second, we deal with scenarios in which *no three-dimensional data* (for example, from range-finding sensors such as lidar and radar) are available or are very limited, bringing our attention to self-supervision and simulation. Third, we concentrate on *lightweight* methods and representations as, to be effective, our systems should be suitable for real-time execution on constrained hardware on a vehicle.

The focus on lightweight methodologies is further emphasized by the project funding this work, SMARTHEP, an EU Marie Skłodowska-Curie Actions Innovative Training Network connecting researchers in the field of real-time data analysis for science, specifically particle physics, and industry applications. The theme of this network is applying machine learning to deal with streams of high amounts

of data which need to be processed in real time on constrained hardware.

## 1.2 Assisted and Automated Driving Systems

Many terms are used in the academic literature, the media and public discourse to refer to the total or partial automation of driving tasks and to automated systems that assist in the driving task: autonomous driving, advanced driving-assistance system, self-driving, driver assistance, etc. Nevertheless, it is not always clear what are these terms exactly describing, and they are often used to portray different concepts depending on the context. To contextualize our work in this wide ecosystem and avoid misnomers, we will give a brief overview of the terms used from a more technical standpoint, and state where our applications of interest lie within this framework.

A standardized taxonomy that has become popular for describing increasing automation in vehicle driving systems is that described by the six SAE levels of driving automation [7] (SAE is a professional and standards association) which are also an ISO standard [8], from no driving automation (Level 0) to full driving automation (Level 5). This taxonomy is valid for systems that perform automated driving on a *sustained* basis. However, some safety features such as automatic emergency braking operate only momentarily in a reactive manner, or warn or inform the human driver without undertaking the vehicle's motion control like forward collision warning. In a wider view of driving automation technology, assisted and automated driving features can be classified according to the "principles of operation framework" described in [9]. Under principle of operation A are warning and informing functions, which inform the driver, e.g. through an auditory signal, without acquiring the vehicle's motion control. Examples of these features are forward collision warning or lane departure warning. Principle of operation B categorizes sustained automated functions as described by the SAE levels of automation e.g., adaptive cruise control or a conditionally automated driving system (SAE Level 3). Finally, principle of operation C encompasses temporarily intervening functions. These functions act in critical or accident-prone situations to avoid or mitigate a crash or collision. Automatic emergency braking and automatic emergency steering are relevant examples that fall under this principle of

operation.

In the application motivating this thesis (dashcams), the device does not have the capability to control the vehicle, and thus our downstream tasks of interest fall under the principle of operation A. Consequently, in the remainder of this dissertation, we focus on technology for the perception and representation of a vehicle's environment, and not technology for path planning or vehicle control.

From the practical point of view, regardless of the final task at hand, be it a collision warning or a fully automated driving system, the underlying technology for perception at both the algorithmic and representation (computer vision and machine learning) and hardware levels (sensors like cameras, processors) has a significant overlap. This means that we can build on research for perception for fully automated driving systems or that on temporarily intervening functions, and our contributions can also be useful in those settings.

Three key aspects of the perception task for these systems are geometry, semantics, and motion. Driving is an inherently geometric task; understanding the scene geometry is crucial to avoid collisions and navigate the road environment. In the forward collision warning example, it is fundamental to understand the distance at which the obstacle in front of the ego-vehicle is. Semantics complements geometry to provide information on the category of objects and "stuff" (understood as non-countable entities: the road, sidewalks, vegetation, ...). This gives wider contextual information of key importance for driving-related tasks e.g., what part of the planar surface of the ground is drivable and which is a sidewalk or an opposite direction lane? Is the obstacle in front of the ego vehicle a car, a bicycle, a pedestrian, or a piece of trash? Finally, understanding motion lets us adjust our predictions in a dynamic environment. This can also easily be understood in the forward collision warning case. Consider the following scenario: a vehicle (semantics) is at a distance of 20 meters (geometry) of the ego-vehicle; being able to estimate their speed and our own (motion) makes the estimation of the collision risk much more tractable.

## 1.3 Why Bird's Eye View?

Why do we use bird's eye view representation for driving? As we have just motivated, driving-related tasks require a geometric understanding of the environment, but the perspective image space is not optimal for geometric reasoning as perspective projection discards depth information. The size of objects depends on the viewpoint of the camera and the object position with respect to it (objects that are closer to the camera occupy a significant higher amount of space in the image). Three-dimensional representations such as an occupancy voxel grid solve this issue, but the memory required to represent them scales cubically with the grid size and resolution. The bird's eye view is a compromise, offering some of the benefits of the explicit 3D representation in a more compressed format. The main advantage of the bird's eye view, compared to other possible projections of the three-dimensional space, is a physical constraint: all objects are gravity-bound, and the movement of all relevant agents participating in road scenes is mostly limited to the ground plane. Under certain reasonable assumptions (the most important of which is the ground being locally flat), the bird's eye view of a scene is a good approximation of the ground plane. Thus, the bird's eye view contains most of the relevant information for driving-related tasks while being compact. Additionally, the following properties of the bird's eye view are valuable for these tasks: it is translationally equivariant, which means that if a position of an object in the world changes (along the horizontal plane), its bird's eye view representation will change accordingly, contrary to what happens in a perspective view image. The bird's eye view is metric, so distances in the bird's eye view are representative of those in the real world. It is also a good medium for sensor and modality fusion. Moreover, if we use structured array-like representations, we can exploit methods designed to process images. This enables using techniques such as convolutional neural networks or image transformers already proven to be successful in vision tasks. While the topic of bird's eye view perception in driving has been widely studied in the last years [10] and others have written full dissertations on the topic [11], in this thesis we address a fundamental gap that we identify in the literature: applications where we do not have access to a high amount of three-dimensional data and where the camera setup is not fully under

our control. Addressing this particular gap is motivated by real-world constraints in an industrial application.

## 1.4 Contributions

The main two contributions of this thesis are the following:

- **RendBEV**: The proposal of the first training framework for bird's eye view segmentation networks without using bird's eye view ground truth labels which does not rely on lidar or precomputed pseudolabels.
- **VS-SIM**: The release of a synthetic dataset designed to isolate and benchmark viewpoint robustness in frontal-view cameras. An analysis of the effect of viewpoint variations on the performance of a bird's eye view semantic segmentation network.

## 1.5 Thesis Structure

The rest of this thesis is organized as follows:

- Chapter 2 goes into the details of constructing representations in the bird's eye view, from the geometric aspects to learning-based approaches, summarizing the vast literature in the topic.
- Chapter 3 presents our method for training bird's eye view semantic segmentation methods without using ground truth data. We provide an extensive experimental evaluation to validate the proposed methodology. The content of this chapter is partly drawn from the publications:
  - The conference paper: **H. P. Monteagudo**, L. Taccari, A. Pjetri, F. Sambo and S. Salti,"RendBEV: Semantic Novel View Synthesis for Self-Supervised Bird's Eye View Segmentation", in *Proceedings of the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Tucson, AZ, USA, 2025, pp. 535-544 [12]

  - The journal paper, extending the previous one: **H. P. Monteagudo**, L. Taccari, A. Pjetri, F. Sambo and S. Salti, "RendBEV: Semantic Perspective View Rendering as Supervision for Bird's Eye View Segmentation", in *IEEE Access*, 2026, vol. 14, pp. 12255-12272 [13]

- Chapter 4 introduces a simulated dataset to study the effect of viewpoint shifts, which are ubiquitous in the application that motivates this thesis, on existing bird's eye view semantic segmentation architectures. The content of this chapter is partly drawn from the paper:

  - R. Turra, M. Simoncini, **H.P. Monteagudo**, A. Pjetri, S. Salti and L. Taccari, "VS-Sim: A Synthetic Dataset for Viewpoint Shift Robustness" *In Proceedings of Image Analysis and Processing – ICIAP 2025* [14]

- Chapter 5 closes the dissertation, with a synthesis of the main insights contained in this work, along with an outlook into the future development of the field.

# Chapter 2

# Bird's Eye View Perception for Driving

The term bird's eye view (BEV) refers to an orthogonal, top-down projection of a three-dimensional scene into a two-dimensional representation. In the context of automated driving and robotics, BEV perception refers to the set of computer vision methodologies that use this representation at any stage of their processing pipeline. The increasing utilization of this approach in the literature (see Fig. 2.1) comes from its effectiveness in representing the environment within a geometrically consistent, metric space that is well suited for planning and reasoning. The role of the BEV representation is versatile and typically falls into one of three categories depending on the task. First, as an input representation: in tasks like motion forecasting, the inputs are often already in BEV, such as agent tracks and map data, and the goal is to predict future BEV states. Second, as an intermediate representation: many 3D object detection pipelines transform features from perspective images into a latent BEV space; these BEV features are then fed to the detection head, which outputs 3D bounding boxes. Third, as an output representation: for tasks like semantic mapping or BEV semantic segmentation, the goal of the entire pipeline is to produce a dense, top-down map of the environment as the final output.

In this chapter, we provide a comprehensive explanation of vision-centric bird's eye view perception for driving, starting from the sensors normally used to pro-

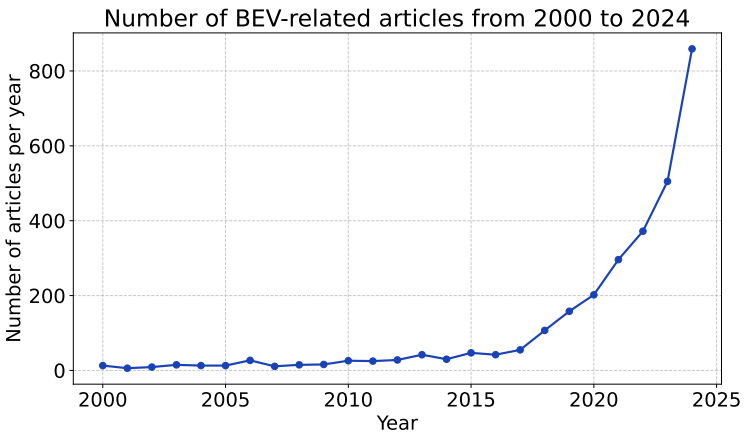


Figure 2.1: Number of published articles including the term "bird's eye view" and/or analogous together with terms related to automated driving or robotics. Source of data: Google Scholar.

duce the input data, following with the geometric foundations that support and formalize bird's eye view representations. Then, we classify representations along two axes: object-centric or grid-centric and the geometric space in which they are defined (e.g., the Euclidean three-dimensional space vs the perspective image space vs the bird's eye view space). Analyzing how representations evolve from the input to intermediate stages to the output according to these two axes helps us to better understand the underlying mechanisms. We later shift to methodologies for learning bird's eye view representations by using neural networks and review the literature on the topic, proposing to classify the methods according to two criteria: how the view transformation is performed inside the model and the task that the method aims to solve.

## 2.1 Sensors for Driving

The first step of any system aiming to automate or enhance driving-related tasks is to perceive the environment, and the first step for perception is sensing.

**Cameras** are passive light sensors in and near the visible spectrum, providing visual information on the scene. The road environment is designed to be navigated by humans, which use mostly visual information to do so, making cameras a natural

choice for perceiving the driving environment artificially. Nowadays, they are cheap sensors, and, thanks to their ubiquity, a huge volume of image data is available. The successes in the field of computer vision in the last decade, fueled by the deep learning revolution, have enabled many applications which where not previously possible with a camera-only setting. Cameras are routinely used in simple applications, from assisted parking [15], which can be as straightforward as providing the driver with a view of the space in the rear of the vehicle, to more complex systems such as forward collision warning [16], where the image data might be further processed to perform intermediary tasks such as object detection and tracking, and distance or time to collision estimation, all the way to fully automated driving systems (attempted in a vision-centric manner by companies such as Tesla and Wayve). The usual representation of the data produced by digital cameras, images, are ordered arrays of intensity values, usually in three channels corresponding to different color components (red, green and blue). The images produced by cameras are the main inputs for the methods described in this thesis.

**Li**ght **d**etection **a**nd **r**anging (**lidar**) systems are active light sensors, which use laser to directly perceive the distance to points in the scene by measuring the time difference in between emitting a laser beam to receiving a reflection. This geometric information is precious, as already mentioned, for driving related tasks, and something cameras cannot directly provide (as depth information is lost in the perspective projection process when the images are taken), which has resulted in many companies pursuing full driving automation to include them in their pipelines. However, lidars are more expensive than cameras, and require careful calibration and powerful data processing (as they generate large volumes of data). As a consequence, less lidar data is available for public usage. The main representation used for lidar data are point clouds, unordered sets of 3D points in the lidar reference frame, with possible additional values associated to each of the points, such as their reflectance. Spatio-temporal fusion of point clouds in a mobile platform, such as a road vehicle, requires further processing and integration with proprioceptive sensors and/or GPS.

**Ra**dio **d**etection **a**nd **r**anging (**radar**) is also an active sensor that uses radio waves instead of laser beams. In addition to their positions, the relative velocity

of objects can also be determined through processing radar data exploiting the Doppler effect, which has made it a common sensor in systems like automatic emergency braking. Radar data is usually represented similarly to lidar data: an unordered set of points given by their range, angle with respect to the radar sensor and their elevation, with possible additional values such as the Doppler velocity. Radar data is generally sparse.

**G**lobal **P**ositioning **S**ystem (**GPS**), provides a position and speed for the ego-vehicle using satellites. This makes it very useful for localization applications, which can enable the usage of precomputed maps of the environment. Precise GPS sensors are expensive and the signal can be absent or deteriorated in tunnels, urban, or forest environments.

**I**nertial **M**easurement **U**nits (**IMUs**) are devices that measure acceleration (through accelerometers) and angular velocity (through gyroscopes), which are fundamental for understanding the ego-vehicle's dynamics. This information can be used to perform temporal fusion, or, together with GPS and/or techniques like visual odometry and simultaneous localization and mapping (SLAM), to obtain ego poses, a fundamental input to many bird's eye view perception methods.

## 2.2 Representation

Cameras produce images, while lidars and radars produce point clouds. Both are *representations* of the environment. For certain downstream tasks, they might be sufficient; for example, a simple parking assist monitor might directly show the image captured by a rear camera and successfully aid the human driver in the parking task. A good representation is one that makes subsequent problem solving easier [17]. Solving a problem such as providing a reliable generic collision warning directly from an image is challenging, and a different representation might be desirable. This representation might encode the distances from the ego-vehicle to objects, for example.

We can describe two properties that are generally desirable for representations: compression and prediction [17].

**Compression.** Compressed representations have the obvious benefit of using less memory and requiring less compute, which is fundamental for any system designed to run in real time on constrained hardware. Additionally, compression can be a way to introduce invariance in a representation. We might want, for example, that, in our collision warning system example, neighboring cars are represented in the same way regardless of the meteorological and lighting conditions, the viewpoint from which they are being observed, or the color of their surfaces. A compressed representation of the environment could discard this information while retaining the higher-level, more abstract concepts that are most relevant for the task in its limited representational capacity; in our example, this could be the position (with respect to the ego-vehicle) and size of the surrounding objects and their classification as vehicles, while discarding information like the appearance and high-frequency texture and surface details present in a raw image.

**Prediction.** The point of generating a representation of the environment is to ease the execution of downstream tasks. Thus, we would like to generate representations which are good inputs for prediction tasks and facilitate decision making. Following our generic collision warning example, we would like to work with representations of the environment that let us predict that a collision is going to happen in the near future, so that we can issue a warning.

### 2.2.1 Classifying Representation

For the purpose of clarifying how different representations are used in the bird's eye view literature and in this thesis, we provide a taxonomy of representation around two different axes. This taxonomy is not exhaustive and does not intend to be general, but rather limited to our topic of interest.

#### Object-centric representation

Object-centric representations are valuable and prevalent in driving as many tasks require identifying objects as individual entities. A widespread object-centric representation – in driving and elsewhere – is the bounding box: in an image, a bounding box is a rectangle containing the object of interest. This is a standard

way of labeling objects in images, as well as representing the detected position in object detection pipelines. The image bounding box can be parametrized by four values. A common way to parametrize them is with the top-left corner together with the boxes' width and height. The bounding box can be generalized to allow for more flexibility or representation capability. For example, to allow for rotated rectangles, or to extend them to three-dimensional bounding boxes, and potentially expressing the extents of the object and its position with respect to the reference frame in metric units and not pixels. It is common practice to also associate semantic information (like the object class) to the geometric one given by the bounding box parametrization.

### Grid-based representation

An alternative way to describe the environment is grid-based representation. In this type of representation, the space is discretized in cells of a certain dimension and numerical values are associated with each cell. A grid-based representation of relevance for this thesis is the occupancy grid [18], originally introduced by A. Elfes for the tasks of robot mapping and navigation. In an occupancy grid, each cell stores a probabilistic estimate of its state (occupied or empty). This can be generalized to more descriptive or restrictive representations: this occupancy can be binarized or broadened to be a *semantic* occupancy, so that only objects or surfaces of a given category are taken into account, e.g. vehicles or the road. This can also be extended to a vector of values, and thus a multiclass segmentation of the cells can be performed.

### Coordinate frames of representation

In an orthogonal axis to the previous two categories offered to classify representation, we have the coordinate frames in which those representations are defined. As we have seen in the example of the bounding boxes, they can be two-dimensional or three-dimensional while still being object-centered bounding boxes. The same is true for grid-based representations. Occupancy grids might be represented in an image coordinate frame, in a three-dimensional Euclidean space, or e.g. in a different two-dimensional frame with some special properties: the bird's eye view.

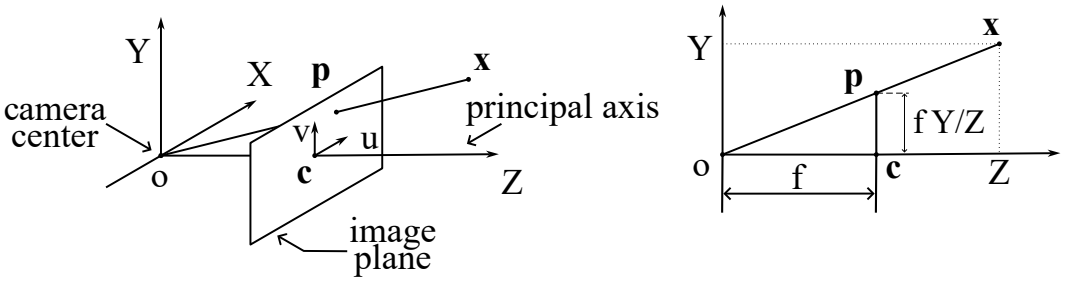


Figure 2.2: Perspective projection with the pinhole camera model. Figure adapted from [19].

The bird's eye view is fundamentally a geometric concept; in the next section, we provide the foundations to let us formalize it and understand certain critical constraints.

## 2.3 Geometric Foundations

We present the geometry behind cameras, as our main sensor, and of the bird's eye view. This provides a more formal basis for our work and supports the understanding of the methods which are used and proposed in the rest of the thesis.

Let us begin with the basics of perspective projection and image formation. An image is a two-dimensional representation of a three-dimensional world [19]. The process that performs this transformation is a projection. In the pinhole camera model, three-dimensional points are projected to a plane through a central projection. Let the projection center be the origin of an Euclidean coordinate system (the camera coordinate frame), which we call the camera center, and the plane $Z = f$ the image plane, where $f$ is the camera focal length. The line from the camera center perpendicular to the image plane is called the principal axis of the camera; the point where the axis meets the image plane is the principal point. In this model, the image of a 3D point with coordinates $\mathbf{x} = (X, Y, Z)^T$ is obtained by finding the intersection of the image plane with a line that passes through the camera center and the 3D point itself. Figure 2.2 (left) shows a visualization of the process. By similar triangles (see Fig. 2.2 (right)), we obtain:

$$(X, Y, Z)^T \rightarrow (fX/Z, fY/Z)^T, \tag{2.1}$$

which describes a mapping from $\mathbb{R}^3$ to $\mathbb{R}^2$ Euclidean spaces. If the points are expressed as homogeneous vectors, used in the geometry of projective spaces ($\mathbb{P}^n$), and for which we refer the interested reader to [19] for an in-depth explanation, this central projection can be represented through a linear mapping and expressed by matrix multiplications as:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.2}$$

which assumes that the origin of the image coordinate frame lies at the principal point. This is often not the case in practice. Expressing the coordinates of the principal point in the image coordinate frame as $(c_u, c_v)^T$ we get the more general expression:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX + Zc_u \\ fY + Zc_v \\ Z \end{pmatrix} = \begin{bmatrix} f & & c_u & 0 \\ & f & c_v & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.3}$$

It is common to write:

$$K = \begin{bmatrix} f & & c_u \\ & f & c_v \\ & & 1 \end{bmatrix}, \tag{2.4}$$

and refer to $K$ as the camera calibration matrix or camera intrinsics matrix. Then, we can re-write Eq. (2.3) as:

$$\mathbf{p}_{\text{img}} = K \begin{bmatrix} I_3 & | & \mathbf{0} \end{bmatrix} \mathbf{x}_{\text{cam}}, \tag{2.5}$$

where $\mathbf{p}_{\text{img}}$ is a vector of coordinates in the image reference system, $\mathbf{x}_{\text{cam}}$ is a point in the three-dimensional space expressed in the camera reference system, and $I_3$ is the $3 \times 3$ identity matrix. Both $\mathbf{p}_{\text{img}}$ and $\mathbf{x}_{\text{cam}}$ are homogeneous.

Let us now switch from central projection to parallel projection and consider projection along the Z-axis. We can write this projection in a similar way as

Eq. (2.2):

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.6}$$

which describes an orthogonal projection along the Z-axis, dropping the Z-coordinate. If our reference system (or the position of our orthogonal camera) is chosen so that the Z-axis corresponds to the vertical coordinate, we get a mapping from a three-dimensional point in space to a two-dimensional horizontal plane i.e. the *bird's eye view*. In a more general case, we have a scaled orthogonal projection, which is defined by the matrix:

$$K_{\perp} = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.7}$$

where $k$ can be understood as the resolution of the BEV space, in $\mathrm{m}^{-1}$ if we are operating with metric units. We obtain BEV coordinates, rewriting Eq. (2.6) in concise form:

$$\mathbf{p}_{\mathrm{BEV}} = K_{\perp} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{\mathrm{cam}}, \tag{2.8}$$

Up until now, we have omitted – for simplicity – that points are in general expressed in a different coordinate system than that of the camera, the world coordinate system. Let $\tilde{\mathbf{x}}$ be an inhomogeneous vector expressed in the world coordinate system and $\tilde{\mathbf{x}}_{\mathrm{cam}}$ represent the same point in the camera coordinate system, then $\tilde{\mathbf{x}}_{\mathrm{cam}} = R(\tilde{\mathbf{x}} - \tilde{\mathbf{o}})$, where $R$ is an arbitrary $3 \times 3$ rotation matrix and $\tilde{\mathbf{o}}$ is the camera center expressed in the world coordinate frame. This can be concisely expressed in homogeneous coordinates as:

$$\mathbf{x}_{\mathrm{cam}} = \begin{bmatrix} R & -R\tilde{\mathbf{o}} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}, \tag{2.9}$$

where $\mathbf{x}$ is now expressed in the world reference frame. The parameters of $R$ and

$\tilde{\mathbf{o}}$ are normally referred to as the extrinsics parameters of a camera, and $\mathbf{t} = -R\tilde{\mathbf{o}}$ is commonly used to avoid making the camera center explicit.

If we know the coordinates of the three-dimensional points in our scene, obtaining a bird's eye view geometric representation of such points is then trivial by what we have just seen; we choose the position and orientation of our world coordinate system (which we place in the ground plane, with the Z-axis perpendicular to it and pointing upwards), our BEV resolution $k$, and we apply Eq. (2.8) and Eq. (2.9). This is done routinely in the literature, for example when a lidar (or lidars, together with their extrinsic parameters), which yields a set of 3D points with metric coordinates, is available.

In the main situation of interest for this thesis i.e. when we only have image data, we cannot directly obtain a bird's eye view. Due to the perspective projection, information about the depth of points has been lost: we cannot directly recover the 3D point that was imaged to an image point. For an image point $\mathbf{p_{img}}$, we can determine the set of points in space that map to this point, which is a ray through the camera center. This process is often referred to as backprojection or unprojection (of points to rays). We know two points (in camera coordinates) on this ray: the camera center $\mathbf{o} = \begin{pmatrix}0,0,0,1\end{pmatrix}^T$ and $\mathbf{x} = \begin{pmatrix}(K^{-1}\mathbf{p_{img}})^T, 0\end{pmatrix}^T$, which lies on the ray because it projects to $\mathbf{p_{img}}$. We verify that $\mathbf{x}$ lies on the ray by substituting in Eq. (2.5):

$$K \begin{bmatrix} I_3 & | & \mathbf{0} \end{bmatrix} \begin{pmatrix} K^{-1}\mathbf{p_{img}} \\ 0 \end{pmatrix} = \begin{bmatrix} I_3 & | & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{p_{img}} \\ 0 \end{pmatrix} = \mathbf{p_{img}}. \tag{2.10}$$

We can then write the line equation for the ray as the join of two points:

$$\mathbf{x}(\lambda) = \lambda \begin{pmatrix} K^{-1}\mathbf{p_{img}} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda K^{-1}\mathbf{p_{img}} \\ 1 \end{pmatrix}, \tag{2.11}$$

where we can interpret $\lambda$ as the Z-component of the point along the line, normally referred to as depth in this context; more explicitly, using Eq. (2.11) and

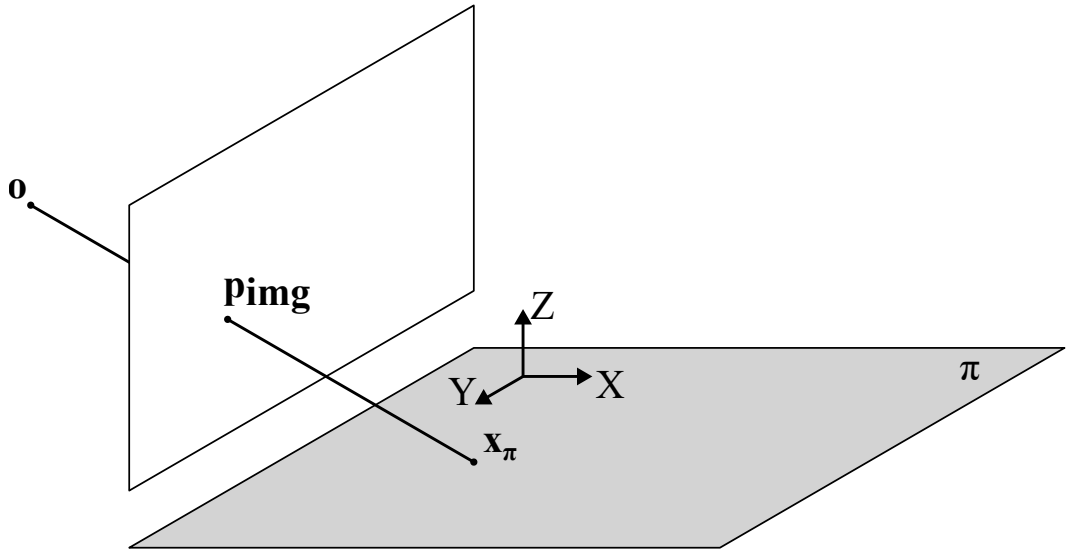


Figure 2.3: A scene point $\mathbf{x}_\pi$ lies on a known plane $\pi$ aligned with the world XY plane. Figure adapted from [19].

transforming to non-homogeneous coordinates:

$$\tilde{\mathbf{x}}(\lambda) = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \left( \lambda \begin{bmatrix} f^{-1} & & -c_u f^{-1} \\ & f^{-1} & -c_v f^{-1} \\ & & 1 \end{bmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \right) = \lambda \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \implies Z = \lambda. \tag{2.12}$$

Let us now consider the special case in which the scene points lie on a particular known plane. Take, for example, a scene where we place the world coordinate system so that its XY plane is aligned with the plane $\pi$ (with zero Z-coordinate) like in Fig. 2.3. We can write the expression mapping points in that plane to image points using Eq. (2.3) and Eq. (2.9):

$$\mathbf{p}_{\text{img}} = K \begin{bmatrix} I_3 & | & \mathbf{0} \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = K \begin{bmatrix} \mathbf{r_1}, \mathbf{r_2}, \mathbf{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \tag{2.13}$$

where $\mathbf{r_i}$ is $i$-th column of $R$. This mapping is a projective transformation or homography (from and to projective 2-space, i.e. $\mathbb{P}^2 \to \mathbb{P}^2$) described by the (in-

vertible) matrix:

$$H = K \left[ \mathbf{r_1}, \mathbf{r_2}, \mathbf{t} \right]. \tag{2.14}$$

Applying a warping based on this transformation to an image where all the imaged points lie on the selected scene plane, would thus result in a proper bird's eye view. One could still apply it to an image with some points belonging to the selected scene plane and others that do not. This would result in the points in the image which came from the plane being correctly mapped and the rest being distorted (see Fig. 2.4 for an example). This specific mapping is also known as inverse perspective mapping (IPM) [20] and it is commonly used as a baseline and as part of more complex methods to obtain bird's eye view representations, as we will see later in this chapter.

## 2.4 Learning Bird's Eye View Representations

Up until now, we have described common devices used to sense the environment for driving, classified representations used in driving and provided the geometric basis for the bird's eye view. Now, we will deal with *how* to obtain bird's eye view representations, focusing on methods that use perspective images as input and how those representations are used. Most of these methods use *learning-based* neural networks.

Most methods that produce a bird's eye view representation from images follow a similar pipeline: first, the input image or images are fed to a neural network, mapping them to a feature space. Second, these features are either lifted to three dimensions using a variety of methods we will now describe in more detail, or are further processed by a neural network to generate the final target outputs. If the features were lifted, the three-dimensional features are then collapsed to the bird's eye view space (using different methods), and are given as input to a neural network that performs the final output task (object detection, occupancy estimation, semantic segmentation, etc). Here, we offer a taxonomy along two axes: how the view transformation is performed and what is the downstream task being solved. Other criteria can be used to further categorize bird's eye view perception methods, primarily whether if and how temporal and view fusion is performed,

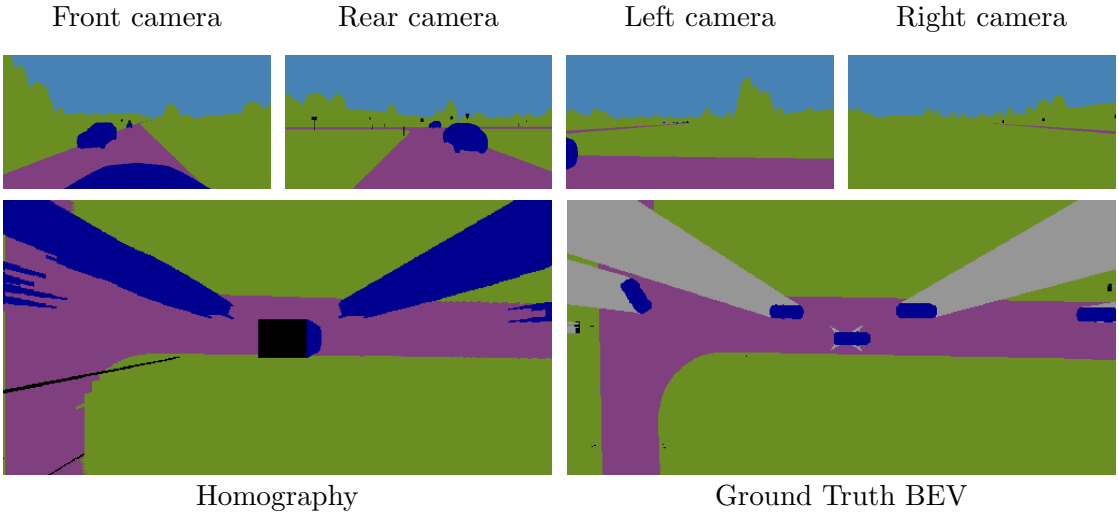


Figure 2.4: Example of an homography applied to semantic segmentation inputs. Distortion in the homography in non-flat regions (cars, in blue color) is easily observed by comparing with the ground truth BEV. Figure source: [21].

which we describe for some of the introduced methods, and modality or sensor fusion, i.e. combining images with other data such as lidar or radar point clouds. We leave such multi-modality methods out of our taxonomy, or only comment on the part of the method that works on images (if the other modalities are optional) as our focus in this thesis lies in camera-only methods.

### 2.4.1 View Transformation

The key step in bird's eye view perception pipelines is the view transformation [10]. As we have seen in the geometric foundations, it is not possible to directly transform image points to bird's eye view points without additional information about the scene. We can classify methods for bird's eye view perception based on how they perform this transformation.

#### Homography-based

These methods rely on warping input images or image features (obtained by passing images through a standard image encoder like a ResNet) to the ground plane using an homography. A common baseline approach in BEV semantic segmenta-

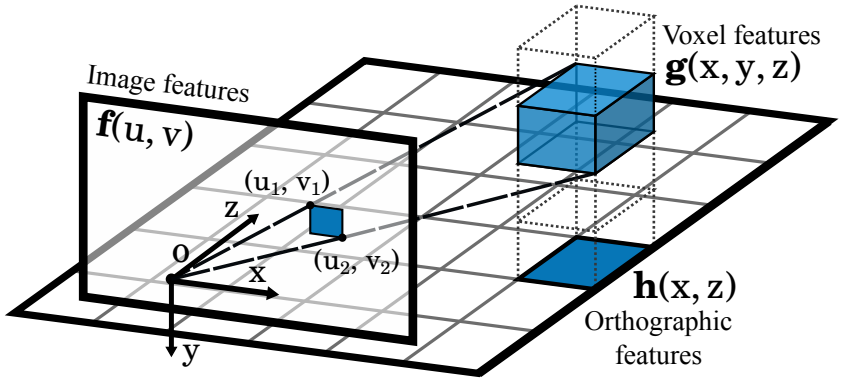


Figure 2.5: Overview of the Orthographic Feature Transform: voxel features are obtained by sampling image features over the projected area of the voxel. The bird's eye view feature map is then obtained by collapsing the vertical dimension. Figure source: [29].

tion (e.g. in [22–25]) is to generate a perspective view semantic segmentation of the image and then warp it using the inverse perspective mapping (IPM) [20]. This process results in extreme deformation of areas which do not belong to the ground plane in the original perspective image, like objects with non-zero height. This phenomenon can be visualized in the cars of Fig. 2.4. Computing the homography matrix to perform the warping requires knowledge about the camera intrinsic and extrinsic parameters (alternatively point correspondences can be used). Abbas et al. [26] propose to regress the necessary parameters to obtain the homography matrix with a convolutional neural network. They parametrize it with four geometric parameters: the horizon line and the vertical vanishing point. Another early work [27] uses an homography to warp the RGB perspective view image to the bird's eye view. The warped image is then fed to an object detection network which outputs oriented bounding boxes in the bird's eye view and is trained in a fully supervised manner. Some works like Cam2BEV [21], BEV feat stitch [28] or PanopticBEV [24] use the IPM to warp the features or part of the features in the perspective view to bird's eye view, while trying to account for its shortfalls through other mechanisms, like learning to mask the parts of the image that do not belong to the ground plane and processing them differently. The main weakness of these methods is the big distortion in not horizontally flat areas, and thus their performance is dependent on the way in how they tackle this limitation.

### Sampling-based

This family of methods relies mostly on the geometric foundations introduced in the previous section to lift two-dimensional features to three dimensions (and not directly to the BEV as with the homography) by sampling from them without learned parameters involved in the lifting process. Then the three-dimensional features are collapsed to the bird's eye view; for this other transformation learnable parameters are normally used. Roddick et al. introduce the Orthographic Feature Transform (OTF or OTFNet) [29], which goes from two-dimensional features obtained by a standard encoder such as ResNet to bird's eye view features through a process depicted in Fig. 2.5. The authors propose to build a voxel grid and populate the voxels with features by performing average pooling in the two-dimensional feature space inside a rectangular area. This rectangle is obtained by projecting the corners of each voxel to the perspective view using camera geometry (see Eq. (2.5)). After the voxel grid has been populated with features in this manner, it is collapsed to a two-dimensional representation (in terms of spatial dimension) by summing the features across the vertical dimension after multiplying them by a set of learned weights. They name the resulting representation the orthographic feature map. The orthographic feature map is passed to a "top-down network" which performs the final downstream task. This approach was the first to introduce this framework for methods working with monocular images only. The authors highlight that this framework allows the top-down network to process in the same way features that are close to the camera to those which are distant (while in the perspective image, closer regions represent a much larger area than distant ones). They state that the goal is to achieve a final feature representation that captures information about the 3D structure of the scene and not its 2D projection. This well-motivated idea of producing an orthographic (bird's eye view) feature map – inspired by 3D object detection methods for lidar data – which then is processed further by a downstream neural network was very influential in later methods. Hartley et al. introduce the Simple-BEV [30] architecture, where center points of a three-dimensional voxel grid are projected to the perspective feature space. The voxel grid is populated with features obtained by bilinearly sampling at the projected coordinates in feature space as depicted in Fig. 2.6 (right). Both

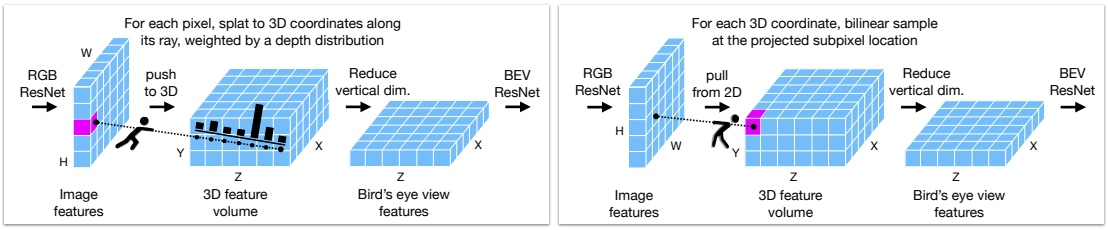


Figure 2.6: Two strategies for view transformation: pushing 2D features to 3D or pulling 2D features from 3D. Figure source: [30].

Simple-BEV and OTF follow a "pulling" strategy: starting from 3D coordinates, they project to the feature space and sample. As Simple-BEV takes multiple perspective images as input (surround cameras), a binary "valid" volume per camera is used to indicate whether each 3D coordinate landed inside the corresponding image limits and thus is inside the camera frustum. Then, a weighted average of the valid voxel features is performed and the vertical and channel dimensions are merged, yielding a high-dimensional bird's eye view feature map. These methods are simple and are able to provide competitive performance without bells and whistles. This makes them strong and commonly used baselines; we use a modified version of Simple-BEV to perform some of the experiments in this thesis. However, some of the more complex methods, as the multi-scale deformable-attention based lifting of BEVFormer [31] are able to obtain better performance as shown in the Simple-BEV article [30] at the cost of more learnable parameters and computational complexity.

### Purely learning-based

These methods directly try to produce a bird's eye view representation from an input image with no explicit geometric constraints, relying on ground truth data in the bird's eye view to guide the process.

An early work (2017) in this group, by Palazzi et al. [32], proposes a network which receives a monocular RGB image and bounding box coordinates as input with two separate branches processing the image and coordinates and finally decodes bird's eye view bounding box coordinates. They create a synthetic dataset using data generated with the Grand Theft Auto V video game to supervise the

objection detection task in the bird's eye view. MonoOccupancy [33] uses a Variational Encoder-Decoder architecture to simultaneously solve the view transformation and segmentation problems, i.e. the network takes as input a monocular RGB image and outputs a semantic-metric occupancy grid map, without any explicit geometric transformation in between. Weak BEV ground truth labels are generated for the Cityscapes dataset by reconstructing 3D point clouds from a disparity map obtained from stereo images and camera calibration, and assigning semantic labels from ground truth annotation in the 2D images to the reconstructed 3D points. This 3D semantic point cloud is projected to the ground plane, and used for supervision in the final task. MonoLayout [34] introduces a similar approach, but extends it to an *amodal* scene layout estimation, i.e. being able to reason behind occlusion boundaries. To do so, it leverages adversarial feature learning and performs sensor fusion to reduce noise in the training data. VPN [35] instead feeds multiple input views to a regular 2D encoder and then utilizes a two-layer MLP to perform the transformation to the bird's eye view. Projecting Your View Attentively (PYVA) [36] uses a cycled view projection in which fully connected neural networks transform image features to the top-down domain and back to image space. The original, top-down and fully cycled features are fed as keys, queries, and values, respectively to a cross-view transformer, the output of which is further processed to generate a bird's eye view segmentation. Overall, these methods suffer from a lack of inductive bias while using small-scale networks and limited amounts of data; this hinders their performance in comparison to other methods which explicitly take into account camera geometry in their modeling.

### Combining learning and geometry

This family of models combines learning and the usage of neural networks with built-in prior knowledge about camera geometry. We further subdivide this family into three different groups: polar transformation models, query-based models and depth-based models.

**Polar transformation models.** In the first group, Pyramid Occupancy Networks (PyrOccNet or PON) [22] is a pioneering work. An input image is processed by a ResNet50 backbone and features are extracted at different resolutions to form

a feature pyramid. Then, those image features are converted to bird's eye view features by a stack of proposed "dense transformer layers" (not related to the attention-based transformer architecture [37]). In these layers, image features are passed through a fully connected layer to reduce their number of channels, and then the height and feature dimension are merged. These "bottleneck features" are then expanded to polar BEV features by another fully connected layer. After that, they are resampled to cartesian coordinates in the bird's eye view space by applying perspective projection to the output grid positions to get the sampling positions in the polar space. Translating Images Into Maps (TIIM) [38] follows a similar approach, but proposes to cast the view transformation task as a sequence to sequence translation from vertical image columns in the perspective view to polar rays in the bird's eye view, and tackle the task with attention [37], implemented as a 1D transformer encoder across feature column elements, and 1D decoder which transform each encoded column into a polar ray of features; the feature rays are then concatenated and resampled to cartesian as in [22] to form the bird's eye view feature map. Optionally, temporal information is aggregated at this stage, by applying axial attention across the spatial and temporal axes of the bird's eye view feature maps corresponding to a sequence of frames.

**Query-based.** A group of models of which representative examples are DETR3D [39] and BEVFormer [31], which we can describe as query-based, bring the idea of reformulating detection and segmentation tasks with transformers from DETR [40] and Deformable DETR [41] to the field of BEV perception. DETR3D uses learnable object queries, which are decoded to 3D reference points via a neural network; these reference points are projected to image feature space using the camera projection matrices (intrinsics and extrinsics) where features are bilinearly sampled and added to the learnable queries, and multihead attention is then applied. BEVFormer introduces a grid of dense BEV queries $Q$ that is fed to a transformer-like architecture which alternates temporal self-attention, spatial cross-attention and feed forward layers. The spatial cross-attention layer works as follows: a pillar of points is lifted from each spatial location in the BEV query grid at a set of predefined anchor heights. Then, those 3D points are projected to a feature space (generated by feeding the perspective view input images to an image encoder) and

a feature is obtained by performing deformable attention [41] with the projected 2D point as reference point. This process is actually equivalent to the way the voxel grid in Simple-BEV [30] is populated with features, substituting simple bilinear interpolation by deformable attention. After several BEVFormer layers, the BEV feature map for the current timestep is produced as output and passed to segmentation or detection heads.

**Depth-based.** Finally, depth-based models predict depth or depth distributions, either using an external model, or learning depth implicitly inside the model. Lift, Splat, Shoot (LSS) [42] is a significant work in this category. LSS receives as input perspective view images from a multi-camera rig, and processes them in different stages. The first step is the "lift". For each pixel $u, v$ in the input image, a discrete number of discrete depths $d_i$ is associated to it, obtaining a number of $u, v, d_i$ points. In addition, a "context" feature and a distribution over the discrete depths is learned. A feature is obtained for each of the 3D points by computing the output product of the context feature and the depth distribution. If the depth distribution was instead a one-hot vector, it would be equivalent to a pseudolidar [43] approach. If the depth distribution was not learned, but defined uniform, it would be similar to the approaches described as sampling-based [29,30] with the important difference of those methods being based on "pulling" from 3D i.e. a 3D grid is defined, and then points from that grid are projected to the perspective view, while LSS is based on "pushing" to 3D i.e. starting from 2D features, they are pushed across rays passing through pixels; see Fig. 2.6 for a visual comparison of pushing and pulling. The second step of the LSS pipeline is splatting: the coordinates are transformed using known camera intrinsics and extrinsics to the unified ego-vehicle reference frame, the space is discretized to a voxel grid and features from points falling inside the same voxel are summed. Finally, the vertical dimension of the voxel grid is collapsed by merging the vertical and feature axes and obtaining the BEV feature map, which is passed to a different neural network for further processing and downstream task execution (including the third stage of Lift, Splat, Shoot, i.e. "shooting" different trajectories for motion planning). Other methods such as CaDNN [44] or BEVDepth [45] follow a similar paradigm, but explicitly supervise the predicted depth with ground truth data.

### 2.4.2 Tasks in Bird's Eye View Perception

We can also categorize bird's eye view perception methods by the task they are trying to solve or the output representation they provide. A basic example task is that of transforming a perspective view RGB image into a bird's eye view RGB image like in [26], usually with the goal of removing perspective distortion in the ground plane. A common application is parking assistance systems [15]. Here we will focus on two tasks: bird's eye view segmentation and 3d object detection.

#### Bird's eye view segmentation

The most relevant task for this thesis, as it is the one we will study in later chapters, is that of bird's eye view segmentation. Closely related and sometimes synonymous are the tasks of semantic occupancy grid prediction and semantic mapping. In general, we can describe it as the task in which a perspective view image (or a set of them, which could be a temporal sequence and/or images from a multi-camera rig) are converted into a grid-like top-down representation of the environment, in which each of the cells of the grid contains a value or vector of values indicating a score for a class being present in that cell. When more than one category is considered (e.g. road and vehicle), the specific modeling in each method determines wether the classes are mutually exclusive or two can be assigned to the same cell. This broadly translates into models that perform a softmax over the output vector for each cell, if classes are mutually exclusive, or a sigmoid, if they can coexist.

MonoOccupancy [33] outputs a $64 \times 64$ occupancy grid, with the size of each cell being $0.5 \times 0.5$m, and 4 possible classes: road, sidewalk, terrain (as ground classes) and a non free-space class (representing cells occupied by objects above the ground plane). MonoLayout [34] performs layout segmentation as classification in the road and sidewalk classes, with a separate branch of the model predicting vehicle occupancy. LSS [42] separately learns to segment the car or vehicle classes for objects, or the drivable area and lane boundary classes for road layour. PON [22] models the problems as multilabel classification across a bigger number of classes, e.g. for the nuScenes dataset: drivable, pedestrian crossing, walkway (sidewalk), carpark, car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bi-

cycle, traffic cone and barrier. BEVFormer [46] segments in four classes: car, vehicles, road and lane.

Some works try to predict not only present but future segmentation, normally for vehicles and/or pedestrians: FISHING Net [47] receives input data from multiple past timesteps and the present, and casts its task as performing vulnerable road user (pedestrian, cyclist, motorist), vehicle and background segmentation in the BEV, not only for the present timestep but also for a number of future ones. FIERY [48] (which performs view transformation in similar way to LSS) casts the task as present and future vehicle instance segmentation and works by predicting centerness, segmentation, offsets and future flows. It uses a probabilistic framework that allows the prediction of multimodal future trajectories. Other models of this type include StretchBEV [49] which uses a similar approach to FIERY with improvements that bring enhanced performance, BEVerse [50] which simultaneously tackles map segmentation, FIERY-like future instance segmentation and 3D object detection, and PowerBEV [51] which proposes to simplify previous methods by substituting recurrent components by a pure convolutional model.

### 3D object detection

BEV perception is routinely used to predict 3D bounding boxes. OTFNet [29], processes BEV features with a convolutional neural network and predicts, for each grid position a confidence score (which indicates the probability of a bounding box with center at those horizontal coordinates and fixed vertical coordinate corresponding to the camera height existing), a position offset which provides a finer localization for the bounding box, a dimension offset, which scales the bounding box dimension, and an angle vector, the one around the vertical axis (yaw). DETR3D [39] predicts BEV bounding boxes by regressing their position, size, heading angle (yaw) and velocity as well as (with a separate classification head) a category label. BEVFormer [31] implements a a custom deformable DETR-like head which is fed with bird's eye view features.

## 2.5 The Data Problem

Most of the works we have discussed up until now rely on full supervision with annotated ground truths BEVs to train the neural networks so that they learn how to produce bird's eye view features and perform downstream tasks. Bird's eye view ground truth annotated labels are cumbersome to obtain and generally require using three-dimensional data, obtained with range-finding sensors such as lidar and radar or stereo camera rigs paired with precise localization, and often, high-definition maps. The usual pipeline consists of capturing data with a rig of multiple sensors able to produce registered point clouds and images, which in a manual or semi-manual process are then annotated by humans to get ground truth labels such as 3D bounding boxes and semantic point clouds. Once this sort of labels are produced, bird's eye view labels for object detection can be produced by projecting 3D bounding boxes to the ground plane; object segmentation bounding boxes can similarly be obtained by projecting said bounding boxes to the ground plane and rasterizing them, while labels for the layout and other surface segmentation can be obtained by orthogonally projecting the semantic point cloud and post-processing it to densify it.

In the last years, thanks to the pioneering work of KITTI [52] and others, and due to the rising interest in academia and especially in industry in automated driving system technology, a number of datasets containing the necessary elements to produce bird's eye view labels was made public. However, a significant number of applications (such as the dashcams that motivate this thesis) fall out of the typical setting in how these datasets were acquired, with carefully calibrated multi-camera rigs, multiple lidars, etc. Additionally, scaling the amount of data acquired with these sensors to capture the tail of the distribution of driving scenes, where many relevant events (such as accidents and other safety critical events such as near misses) occur remains close to an impossible task – at least in the near future – as it requires deploying a huge amount of resources for a significant time.

### 2.5.1 Datasets for Supervised BEV Perception

The 3D object detection benchmark in KITTI [52] is one of the first extensively used in bird's eye view perception tasks, it consists of 7,481 training images and 7,518 test images as well as the corresponding point clouds, comprising a total of 80,256 labeled objects. It is used by e.g. the pioneering OTFNet [29] to generate its ground truth data for supervision, by creating Gaussian regions in the BEV around the horizontal coordinates of the object center. nuScenes [53] is one of the most used datasets in bird's eye view perception, which we also use in Chapter 3. It features image data coming from six cameras covering the whole $360^{\text{o}}$ around the vehicle, paired with lidar and radar. It provides annotation in the form of 3D bounding boxes for several different classes: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, traffic cone and barrier. It additionally includes human-annotated highly-precise semantic maps of relevant areas with 11 semantic layers (provided in the so called map expansion, released at a later date than the original dataset) including drivable area, pedestrian crossings, walkways, etc. PON [22] combines nuScenes' map data and 3D bounding boxes to create multilabel bird's eye view semantic segmentation labels paired with images, which they use for supervised learning. PanopticBEV [24] processes the dataset in a similar manner, but extends the annotation to panoptic segmentation, so that objects have individual ids associated with them. The KITTI-360 [54] dataset provides data captures by 4 cameras, a stereo pair at the front, and two $180^{\text{o}}$ field of view fisheye cameras on each side of the vehicle, paired with lidar data and precise localization. The dataset is annotated with 3D primitives at the point cloud level, yielding a semantic point cloud and 3D bounding boxes. PanopticBEV [24] obtains bird's eye view semantic segmentation labels from these data by accumulating semantic point clouds from different timesteps for static objects and projecting them to the ground plane, densifying the resulting sparse bird's eye view maps with morphological operators and projecting 3D bounding boxes to get object labels as illustrated in Fig. 2.7. SkyEye [25] uses the same method to generate ground truth data for evaluation, and a similar pipeline (described in the following subsection) to generate pseudolabels in which the 3D data are not used. The large scale Waymo Open Dataset [55] offers similar data, and is also used to generate

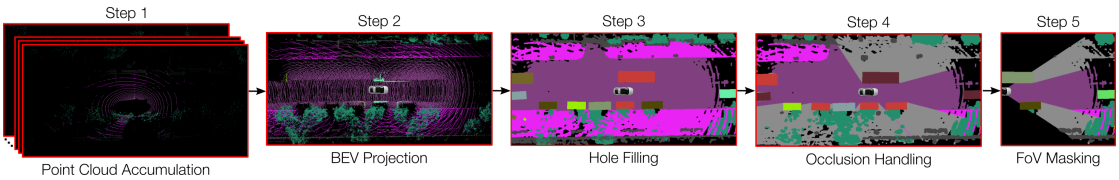


Figure 2.7: Pipeline for BEV semantic segmentation generation in PanopticBEV [24]. Figure source: [24].

labels e.g. for training in BEVFormer [31] or evaluation [25] in SkyEye.

### 2.5.2 Learning with Limited Data

Early works such as "Learning to map vehicles into the bird's eye view" [32] circumvented the lack of real data by exploiting photorealistic video games to generate data. A later method, Cam2BEV [21] uses software specifically created for driving simulation to create bird's eye view ground truth data. Simulators are powerful tools because they provide full control over the environment and sensor placement, as well as good quality ground truth with no human annotation. The open source CARLA [56] simulator, powered by the Unreal graphics engine provides many of the tools necessary to procedurally generate these data out of the box, and has become a popular tool for data generation in the automated and assisted driving research community (see e.g. [57–60].

MonoLayout [34] proposes to use monocular depth estimation and semantic segmentation models paired with odometry data (to perform temporal accumulation) to lift semantic point clouds for static classes, and then orthographically project them to obtain noisy bird's eye view pseudolabels. They then use these pseudolabels to train bird's eye view segmentation models. SkyEye [25] takes this concept further by proposing a pseudolabeling pipeline using a monocular depth estimation model, but also generating pseudolabels for dynamic objects. They do so by retaining only points consistent with the ground truth at a fixed timestep, clustering the points with the DBSCAN algorithm, then projecting the clusters orthographically to the BEV and finally fitting ellipses around each cluster using the RANSAC algorithm. SkyEye combines the idea of using bird's eye view pseudolabels for fully supervised training with a pretraining step based on predicting

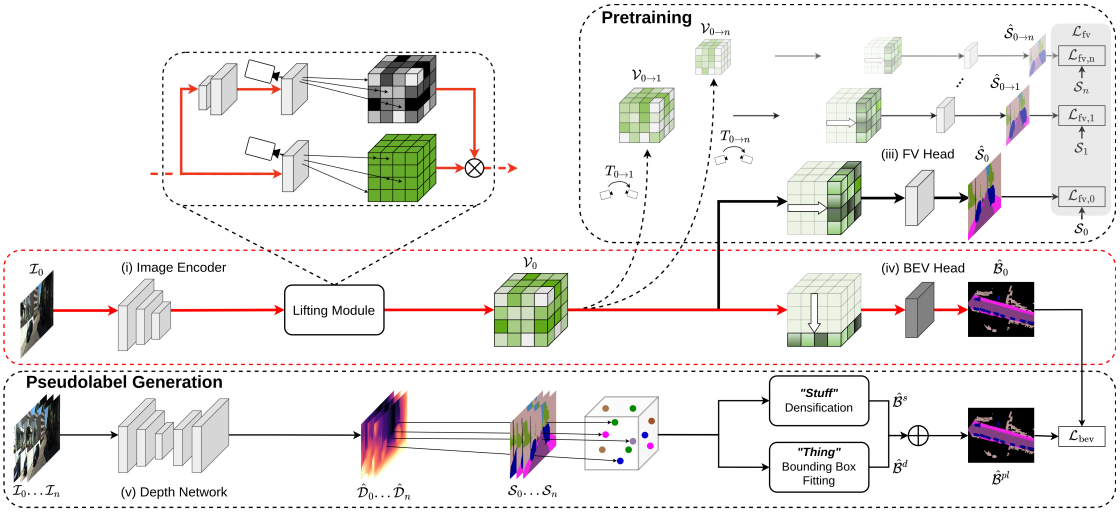


Figure 2.8: SkyEye network architecture, pseudolabel generation and training strategy overview. Figure source: [25].

present and future semantic segmentation in the perspective view from its intermediate three-dimensional feature volume. The network architecture and the training strategy are shown in Fig. 2.8. The lifting module for view transformation follows an implicit depth-based strategy. In Chapter 3 we use SkyEye's architecture, but we devise a pseudolabel-free training strategy that can be used to train the model without any type of BEV supervision and also as pretraining if ground truth labels are available.

LetsMap [61], proposes a similar pretraining strategy to obtain good three-dimensional or bird's eye view features, starting from a self-supervised learning image encoder (DINOv2 [62]), but predicting future RGB frames instead of semantics. With this pretraining, they are able to provide performance on-par with fully-supervised state of the art methods while limiting the amount of labeled data they use.

Semi-BEVseg [63] approaches the training of BEV semantic segmentation networks in low-annotation data regimes within a semi-supervised learning paradigm instead. The authors propose to use a teacher-student framework with consistency losses together with an augmentation explicitly designed for the monocular BEV segmentation task (a conjoint rotation). This allows them to exploit unlabeled data during training, enhancing the performance in experiments in the nuScenes

dataset.

Given our motivation and the lack of the necessary sensor data to produce fully supervised labels at our disposal, the methods we study are of this category. In Chapter 3 we explore how to train existing methods for bird's eye view semantic segmentation without any bird's eye view labels or pseudolabels. We perform this work on public datasets which do have these type of data available, so that we can evaluate our results and compare against existing methods. However, this analysis leaves out certain problems that are prevalent in Verizon Connect's internal dashcam data. One of these problems is the huge variability in terms of mounting positions of Verizon Connect's customers dashcams, with a lot of variance coming from the type of vehicle in which the camera is mounted and the installation process itself, which is often performed by the clients themselves. In Chapter 4 we explore the usage of simulation to create data mimicking viewpoint variations observed in Verizon Connect's real data, and test its effect on bird's eye view semantic segmentation models.

# Chapter 3

# RendBEV: Semantic Perspective View Rendering as Supervision for Bird's Eye View Segmentation

In this chapter, we introduce RendBEV, a method to supervise bird's eye view (BEV) semantic segmentation networks without using any BEV labels or pseudolabels. Our goal is to make training of models for BEV semantic segmentation possible even in scenarios where annotation is too expensive or plainly unfeasible like when only dashcam data is available. RendBEV requires only video sequences with camera poses to train a semantic segmentation BEV. At training time, we use its BEV prediction for a frame to render semantic perspective views from the

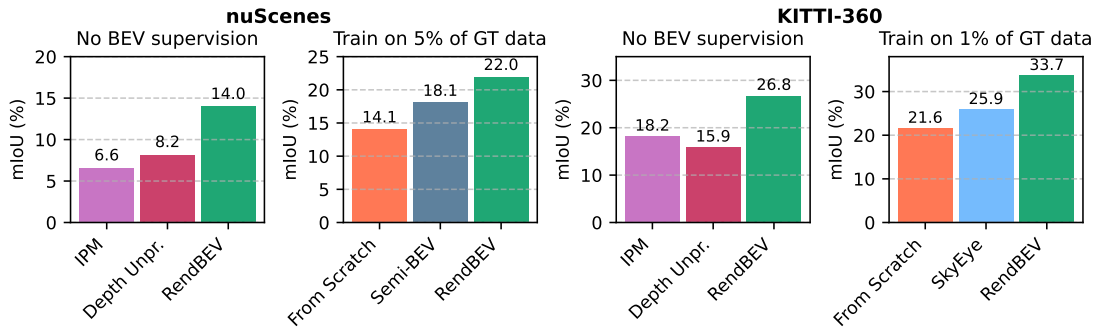


Figure 3.1: RendBEV's performance in zero and low annotated data regimes compared with baselines and competing methods

sequence. A loss function to update the BEV network can then be computed using perspective view targets. These targets, in scenarios where ground truth data are not available, can be obtained through off-the-shelf semantic segmentation models. Rendering of semantic segmentation in perspective view from the predicted BEV is made possible by recent progress in methods for metric depth estimation and novel view synthesis, which can yield good estimates of the scene geometry. In this chapter, we show that both types of method can be used for the reconstruction of perspective semantic views, which we then use to supervise the BEV semantic segmentation model. This leads to competitive results, which over-perform baselines by a significant margin, as we highlight in Fig. 3.1.

The main contributions of the work presented in this chapter are the following:

- The main contribution is the proposal and study of RendBEV, the first method – to the best of our knowledge – to train bird's eye view semantic segmentation networks that uses no BEV ground truth. Contrary to the most similar work to ours [25], we do not rely on precomputed BEV pseudolabels, supervising only in perspective view via rendered semantics.

- We provide extensive experimental results on the nuScenes [53] and KITTI-360 [54] datasets.

- We show that if a limited amount of BEV ground truth labels is available, our method can be used as pretraining, providing better results than methods specifically designed for semi-supervised learning in the nuScenes dataset and outperforming other pretraining methodologies in the KITTI-360 dataset.

## 3.1 Scene Reconstruction and Neural Fields

The prediction of scene geometry from monocular cameras has been a field of great interest and progress in recent years. Monocular depth estimation models learn to predict the depth for each pixel. Self-supervised depth-from-mono models [64] learn to do so in a self-supervised way by exploiting geometric consistency in image sequences. These methods cast the problem as novel view synthesis, predicting the appearance of a target image from the viewpoint of another im-

age and minimizing a photometric loss. Other models use full supervision with depth ground truths. Recently, big, fully-supervised models [65–68], trained on very large amounts of data from multiple domains, both synthetic and real, have made progress in enabling zero-shot monocular depth-estimation, with improved generalization capabilities. Some of these models like Metric 3D [66, 67] are able to provide *metric* depth estimation, i.e. not just relative or affine-invariant but directly in metric units. Metric 3D v2 achieves this by jointly training on large quantities of diverse depth and surface normal ground truth data as well as using a canonical camera space transformation module with which the scale ambiguity is resolved (with known camera focal length). In this work, we employ Metric3D v2's capability of metric depth estimation to provide us with information about the scene geometry and reconstruct semantics predicted by BEV semantic segmentation networks. This allows us to train them by shifting the supervision to the perspective view.

Inspired by Neural Radiance Fields (NeRFs) [69] and image-conditioned NeRFs [70], Behind the Scenes [71] proposes a method to obtain a more complete geometric representation of a scene than a per-pixel depth: an implicit density field. This field maps every point in the frustum of the input image to a density value. The architecture is composed of an image encoder-decoder and a MLP, which are jointly optimized by an image reconstruction loss. The reconstruction is obtained with a differentiable volume rendering formulation. S4C [72] extends [71] with an additional MLP that predicts class logits to tackle the task of semantic scene completion, with the addition of semantic segmentation pseudolabels as reconstruction targets. In this work, we leverage Behind the Scenes to perform differentiable volumetric rendering and create supervision, but we do not change its architecture to solve an additional task. In contrast, we use it to render class probabilities predicted by existing specialized BEV networks to make it possible to train them in a self-supervised way. A visual summary of RendBEV is provided in Fig. 3.2.

## 3.2 The RendBEV Method

RendBEV is based on one core idea: supervising a bird's eye view segmentation model without direct BEV supervision by sampling its output and rendering dif-

ferent semantic perspective views using the sampled values.

### 3.2.1 Problem Setting and Notation

We assume the following setting: we have access to a dataset of videos captured from a monocular camera. We identify the $k$-th frame of a video as $I^k \in \mathbb{R}^{3\times H\times W}$. We have access to the semantic segmentation $S^k$ of the frames, with a predetermined set of $C$ classes $\mathcal{C} = \{0, ..., C-1\}$. These can be the labeled GT if available or obtained from a trained model. Camera poses for each frame with respect to an arbitrary world reference frame $M^{k\to w} \in \mathbb{R}^{4\times 4}$ and intrinsic parameters $K \in \mathbb{R}^{3\times 3}$ of the camera are also known. We denote pixel locations in the images with $\tilde{\mathbf{p}} = (u, v)^T, \mathbf{p} \in \{1, ..., H\} \times \{1, ..., W\}$, or $\mathbf{p} = (u, v, 1)^T$ if homogeneous. We denote three-dimensional points as $\mathbf{x}$ when they are homogeneous and $\tilde{\mathbf{x}}$ when inhomogeneous.

At training time, we subsample videos in the dataset to obtain shorter sequences $\mathcal{T} = \{1, \dots, T\}$ of frames $I^t$, $t \in \mathcal{T}$. A BEV semantic segmentation network working on monocular RGB images takes as input a single reference image $I^t$ and outputs a bird's eye view segmentation logit map $\hat{B}^t \in \mathbb{R}^{C\times H_{BEV}\times W_{BEV}}$. RendBEV aims to (pre-)train any such network without access to BEV targets $B^t$ (neither labels nor pseudolabels) for supervision or any type of range-finding sensor data like lidar or radar. To do so, we propose to sample semantic values (logits or class probability values) from $\hat{B}^t$ and render semantic views $\hat{S}^{t'}$, where $t' \in \mathcal{T}$, with the sampled values (Sec. 3.2.2) by relying on a geometry module to provide 3D information of the scene (Sec. 3.2.3). Our training method is agnostic to the architecture of the BEV semantic segmentation network, as it works by sampling its final output, without accessing any intermediate representation.

### 3.2.2 Semantic Sampling

To perform the training in this constrained scenario, we propose to shift the supervision from the BEV to the perspective view (PV) by rendering semantic views $\hat{S}^{t'}$. Fig. 3.3 provides a visual overview of the process. Producing the views $\hat{S}^{t'}$ to match targets $S^{t'}$ is our pretext task. The model needs to learn a spatio-temporally consistent, correctly segmented BEV to improve in this pretext task. In this set-

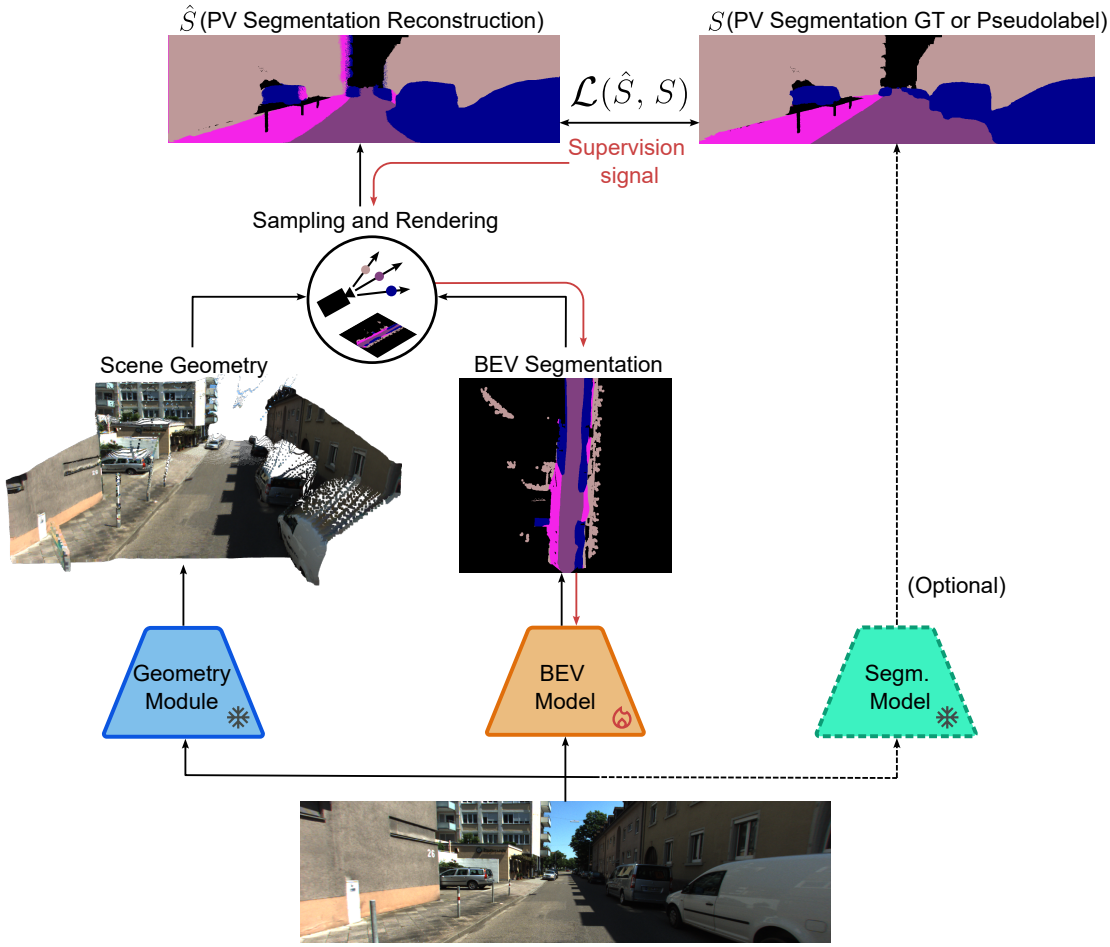


Figure 3.2: Visual summary of RendBEV: we supervise a bird's eye view (BEV) segmentation model by sampling its output and rendering a perspective view (PV) with which the loss function is computed. The loss function target can be the ground truth (GT) PV segmentation if available or a pseudolabel computed with an off-the shelf segmentation model. A geometry module, using pretrained 3D vision networks, provides the scene geometry for rendering.

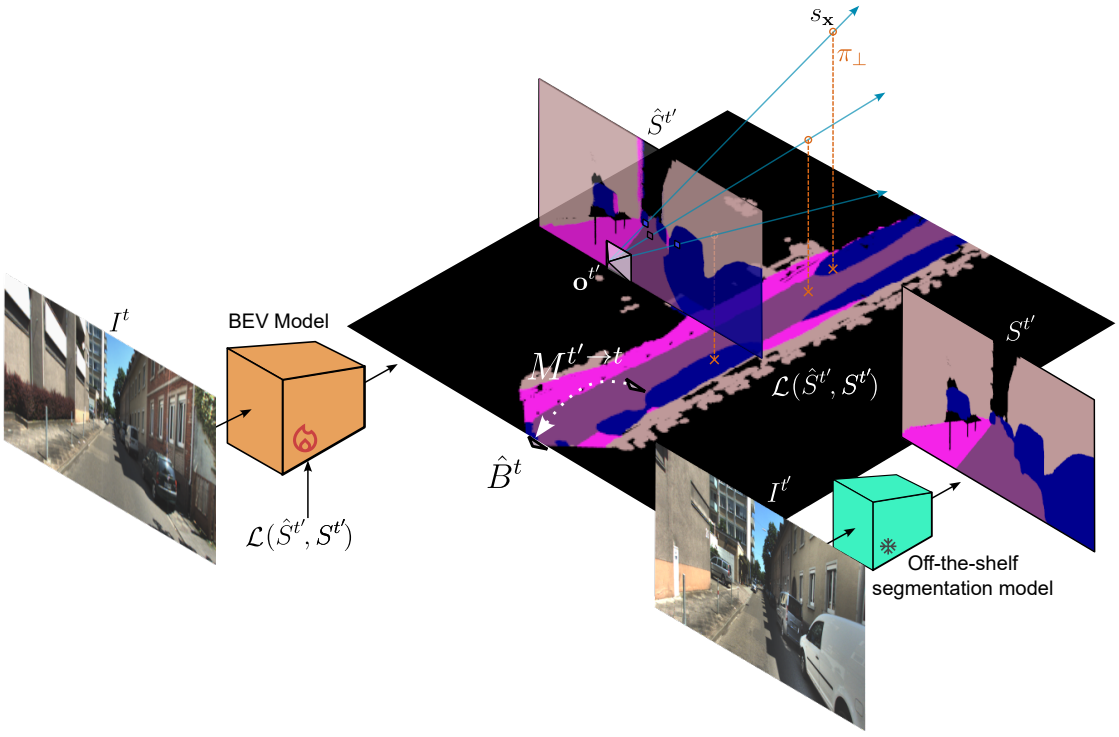


Figure 3.3: Overview of the core idea in RendBEV: we *render* the semantic segmentation $\hat{S}^{t'}$ of view $I^{t'}$ with class probability values $s_{\mathbf{x}}$ sampled from the *BEV* $\hat{B}^t$ prediction, enabling us to shift the supervision from the BEV to the perspective view by computing $\mathcal{L}(\hat{S}^{t'}, S^{t'})$

ting, we assume that the scene is locally static, so that we can reconstruct multiple views (at different temporal instants $t'$) from the BEV produced from a single view. This allows us to provide a richer supervision to the model, including more points in far areas and in regions occluded in the reference view. However, real-world data will include dynamic objects. The explicit handling of dynamic objects is currently an open research problem, which we leave for future work.

Given a sequence $\mathcal{T}$ of frames, and known camera intrinsics (pinhole camera model):

$$K = \begin{pmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

we process a single frame $I^t$ with our BEV semantic segmentation network, obtaining a BEV prediction $\hat{B}^t$. Next, for each image in the sequence, we unproject image pixels $\mathbf{p} = (u, v, 1)^T$ to rays with direction vector:

$$\mathbf{v} = K^{-1}\mathbf{p} = \begin{pmatrix} \frac{u-c_u}{f_u} \\ \frac{v-c_v}{f_v} \\ 1 \end{pmatrix}, \tag{3.2}$$

which we normalize to get a unit vector:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{||\mathbf{v}||}. \tag{3.3}$$

The equation for each 3D point $\tilde{\mathbf{x}}_\mathbf{p}$ along the ray at distance $\rho$ from the camera center $\mathbf{o}^{\mathbf{t}'}$(expressed in the camera coordinate frame of $t'$) is then:

$$\tilde{\mathbf{x}}_\mathbf{p}(\rho) = \rho\hat{\mathbf{v}}. \tag{3.4}$$

For each point, we transform its coordinates to those of the reference frame using camera poses

$$M^{t'\to t} = (M^{t\to w})^{-1} M^{t'\to w}, \tag{3.5}$$

and project the points to the BEV orthographically by dropping their vertical

component $y$ with the projection matrix:

$$\pi_{\perp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

Then, for each point $\mathbf{x_p}$, we obtain a vector of semantic values (either logits or class probabilities) $s_{\mathbf{x_p}}$ by sampling in $\hat{B}^t$ with the transformed coordinates. We express the whole operation in Eq. (3.7) (in homogeneous coordinates):

$$s_{\mathbf{x}_p(\rho)} = \hat{B}^t \langle \pi_{\perp} (M^{t' \to t} \mathbf{x_p}(\rho) \rangle, \tag{3.7}$$

where $\langle \cdot \rangle$ is a sampling operator.

After that, the semantics $\hat{S}_{\mathbf{p}}$ for pixel $\mathbf{p}$, are obtained by aggregating the sampled values along each ray. Let $m$ be the number of points sampled along a specific ray at distances $\rho_i$, $\mathbf{x}_i = \mathbf{x_p}(\rho_i), i = 1, \ldots, m$. We define $\alpha_i$ as the probability of the ray hitting a surface along the ray between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, and $T_i$ as the probability that the ray travels in free space before $\mathbf{x}_i$. We express the rendering along the ray as

$$\hat{S}_{\mathbf{p}} = \sum_{i=1}^{m} w_i s_{\mathbf{x_i}}, \tag{3.8}$$

where $w_i = \alpha_i T_i$ are referred to as rendering weights. In the following subsection, we present two ways to concretely realize this aggregation.

Finally, we supervise the BEV semantic segmentation network by computing a loss (e.g., a cross-entropy loss or a dice loss) with the rendered perspective views $\hat{S}^{t'}$ and the semantic perspective view targets $S^{t'}$:

$$\mathcal{L}(\hat{S}^{t'}, S^{t'}), \tag{3.9}$$

the specific loss used in each setting is presented in Sec. 3.3.2.

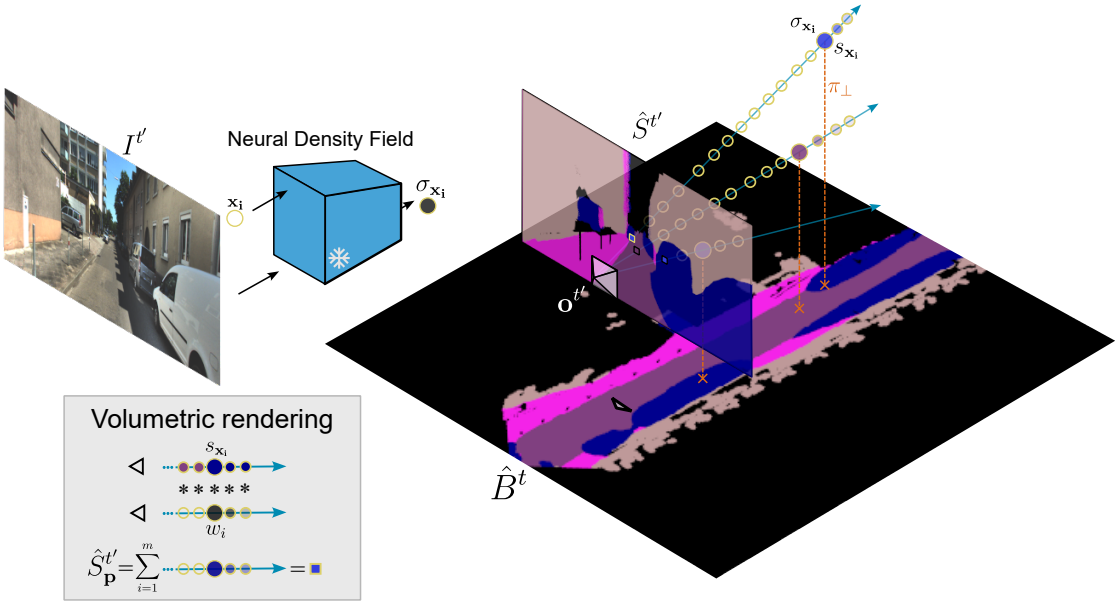


Figure 3.4: Reconstructing a semantic perspective view $\hat{S}^{t'}$ with volumetric rendering and a neural density field: we sample $\mathbf{x}_i, i = 1, \ldots, m$ across each ray and query volumetric densities $\sigma_{\mathbf{x_i}}$ from the neural density field. Rendering weights $w_i$ obtained from the volumetric densities are combined with the sampled semantics $s_{\mathbf{x_i}}$ and summed to obtain pixel values $\hat{S}^{t'}_{\mathbf{p}}$.

### 3.2.3 Rendering and Geometry Module

We use a pretrained model to provide information about the geometry of the scene so that we can render the perspective views. This model can be either a neural density field or a monocular depth estimation network.

#### Volumetric rendering with a neural density field

The first possibility is to use a neural density field such as Behind the Scenes [71], which is trained in self-supervised fashion on the target dataset in a preliminary step. Such model can be queried with a 3D point $\mathbf{x}_i$ and a feature map to provide a volumetric density value $\sigma_{\mathbf{x}_i}$. In this setting, we sample $m$ points uniformly in disparity with an added random noise factor along each of the rays we cast, following [71]. Different to [71] we query a density value $\sigma_{\mathbf{x}_i}$, by feeding the model with a feature map extracted from $I^{t'}$ from which the ray is cast for each sampled point $\mathbf{x}_i$. We also sample a semantic vector $s_{\mathbf{x}_i} = s_{\mathbf{x}_\mathbf{p}(\rho_i)}$ as described in Eq. (3.7). Then,

we use the following formulation (illustrated in Fig. 3.4) to perform volumetric rendering: let $\delta_i$ be the distance between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, we compute:

$$\alpha_i = 1 - \exp(-\sigma_{\mathbf{x}_i}\delta_i). \tag{3.10}$$

Given the previous $\alpha_j, j = 1, \ldots, i-1$, along a ray, we can compute the probability $T_i$ as:

$$T_i = \prod_{j=1}^{i-1}(1 - \alpha_j). \tag{3.11}$$

This is routinely used in novel view synthesis to decide the color $\hat{c}$ of a pixel by integrating colors of the 3D points $c_{\mathbf{x}_i}$ along the ray as

$$\hat{c} = \sum_{i=1}^{m} w_i c_{\mathbf{x}_i}, \tag{3.12}$$

where $w_i = \alpha_i T_i$. In our case, we are interested in rendering vectors of class probabilities (or logits), so we do:

$$\hat{S}_{\mathbf{p}} = \sum_{i=1}^{m} w_i s_{\mathbf{x}_i}, \tag{3.13}$$

and obtain a vector of semantics for each pixel in the target views.

**Depth-based rendering with a monocular depth estimation network.**

An alternative approach (illustrated in Fig. 3.5) consists of passing the sequence frames $I^{t'}$ through a pretrained monocular depth estimation network, which provides a depth estimate $d_{\mathbf{p}}^{t'}$ for each pixel. For each ray cast from the camera center $\mathbf{o}^{\mathbf{t}'}$ across a pixel $\mathbf{p}$, we thus get a single 3D point $\tilde{\mathbf{x}}_{\mathbf{p}} = \tilde{\mathbf{x}}_{\mathbf{p}}(d = d_{\mathbf{p}}^{t'})$ using a variation of Eq. (3.4)

$$\tilde{\mathbf{x}}_{\mathbf{p}}(d) = d\mathbf{v}. \tag{3.14}$$

Note that in this case, we are using the viewing direction vector $\mathbf{v}$ without normalizing it (as is done in Eq. (3.4)). This is due to the fact that the output of monocular depth estimation models is a z-depth, with the interpretation of depth

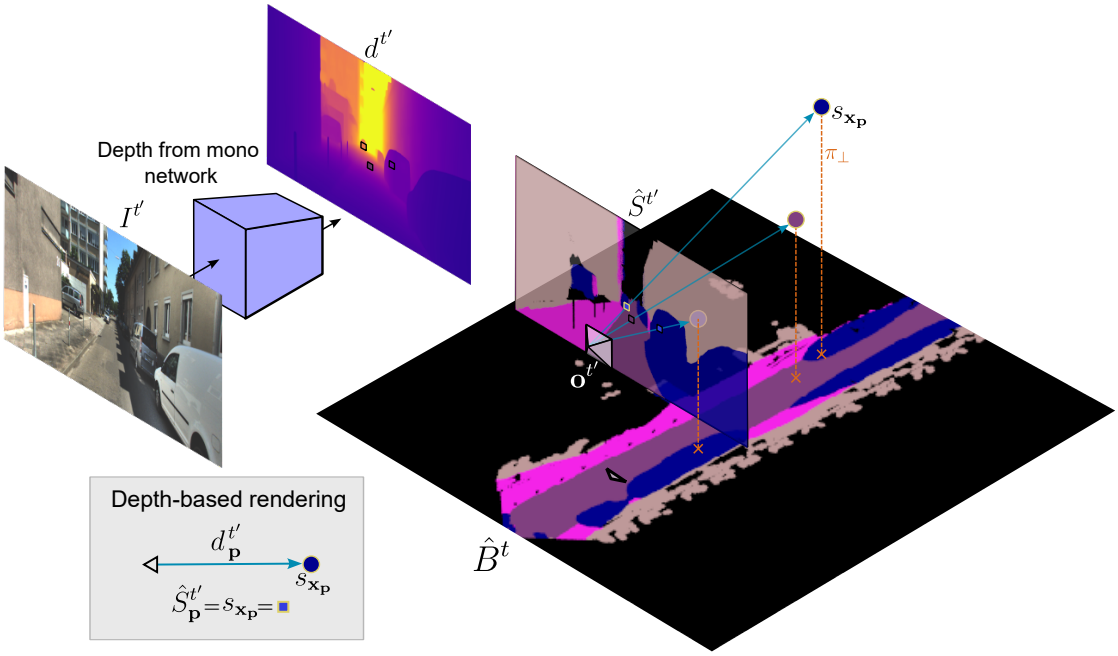


Figure 3.5: Reconstructing a semantic perspective view $\hat{S}^{t'}$ with depth-based rendering and a monocular depth model: a single point $\mathbf{x_p}$ is obtained across each ray at the depth $d^{t'}_{\mathbf{p}}$ given by the output of the monocular depth model at the pixel through which the ray was cast, the semantic value sampled from the BEV $s_{\mathbf{x_p}}$ can thus directly be mapped to the pixel value $\hat{S}_p$

seen in Eq. (2.12), and not a ray distance. This approach can be understood as a limit case of the previously introduced method based on volumetric rendering. In this case, the volumetric density $\sigma$ along a ray would take the shape of a step function, so that:

$$\sigma_i = \begin{cases} 0 & d_i < d^{t'}_{\mathbf{p}} \\ L & d_i \geq d^{t'}_{\mathbf{p}} \end{cases}, \tag{3.15}$$

where $L$ is a very high value and $d_i$ is the depth of point $\tilde{\mathbf{x}}_{\mathbf{p}}(d_i) = d_i\mathbf{v}$. Consequently, the rendering weights would be binarized, so that $w_i$ is zero for every point except the one before hitting a surface, where the ray terminates. This is equivalent to saying that all surfaces are either fully transparent or fully opaque.

We then sample a vector of class probabilities or logits $s_{\mathbf{x_p}(d)}$ for each of these points following the process described in Sec. 3.2.2. In this case the rendering process is a backward mapping, as we have a single 3D point corresponding to

each pixel, so that:

$$\hat{S}_{\mathbf{p}} = s_{\mathbf{x}_{\mathbf{p}(d)}} = \hat{B}^t \langle \pi_{\perp}(M^{t' \to t} \mathbf{x}_{\mathbf{p}}(d)) \rangle \tag{3.16}$$

Using this alternative has two main advantages: first, foundational depth models like Metric3Dv2 [67], trained on a variety of real world and simulated datasets have been shown to generalize to a variety of settings with good performance, enabling the application of our method to new scenarios without the need to previously train a neural density field (which might prove difficult in certain cases, like when images are corrupted due to environmental factors, e.g., rain or when the quality of the camera poses is not optimal). Second, it alleviates the computational burden. Instead of running the neural density field for each point across each ray and performing volumetric rendering, we can run the monocular depth estimation model on each frame $I^k$ in our training dataset in a preliminary step and store the results. The rendering process then reduces to reading the precomputed depth, unprojecting the pixels in the target views and sampling the BEV output. As our goal is to render class probability values or logits and not color, and we are only concerned with solid objects that we can safely consider "opaque" in the semantic domain, we assume this can be a good enough approximation for our application.

## 3.3 Experiments

### 3.3.1 Datasets

We perform training experiments on the nuScenes [53] and KITTI-360 [54] datasets. We generate ground truth BEVs from the nuScenes dataset following the instructions given by the authors of PON [22] and use the BEV version of KITTI-360 provided by the authors of SkyEye [25]. These ground truth BEVs are used for evaluation and fine-tuning experiments, but not when training with RendBEV.

All of our experiments are performed in a monocular setting using the frontal camera (both datasets feature a multi-camera rig), although our method can easily be extended to a multiple camera configuration.

### 3.3.2 Experimental Setting

We aim to evaluate the performance of RendBEV in a setting with a minimum amount of supervision. We explicitly avoid using lidar data or ground truth perspective view semantic segmentation data. Instead, we use models which have been trained on other datasets or in a self-supervised way. Metric3D v2 [67] is trained with full supervision on a large combination of datasets, which does not include KITTI-360 or nuScenes. Before the training experiments, we run Metric3D v2 over all the frontal images in both datasets and save the obtained depth maps, which allows to reduce the computation and memory footprint at training time. BehindTheScenes [71] is trained in a self-supervised way in a preliminary step.

We perform training experiments using RendBEV on the KITTI-360 [54] dataset using either Metric3D v2 [67] (DINO v2-reg ViT-giant backbone) for depth-based rendering or BehindTheScenes [71] for volumetric rendering. With these experiments we aim to evaluate the overall performance of models trained with our method and the impact of switching our rendering strategy. We use the BEV semantic segmentation network described in SkyEye [25] as our BEV model, except where explicitly stated. The input image size is 1408 x 384. We use as semantic segmentation targets in the perspective view pseudolabels produced by an off-the-shelf Panoptic-Deeplab [73] model trained on the Cityscapes [74] dataset (i.e. run zero-shot on KITTI-360). These pseudolabels were made public by the authors of S4C [72]. The KITTI-360 BEV segmentation dataset [25] (which we use for evaluation and fine-tuning experiments) and the pseudolabels are multiclass and single label: a single class (or none) is associated to each pixel. For training in KITTI-360, we use a weighted cross entropy loss, with class weights equal to the ones in [25]:

$$\mathcal{L}_{\text{KITTI-360}} = WCE(\hat{S}, S) \tag{3.17}$$

We perform training experiments on the nuScenes dataset [53] using Metric3D v2 and depth-based rendering. The architecture introduced in [25] is our BEV model. The input image size is 896 x 448. We generate our own semantic segmentation pseudolabels using the GroundedSAM2 pipeline [75–77] prompted with groups of text strings corresponding to the classes used to benchmark BEV semantic segmentation models in the dataset, see e.g. Tab. 3.1. With these experiments,

our purpose is to verify the applicability of our method to new scenarios with minimal supervision. This recipe (Metric 3D v2 for depth and GroundedSAM2 for semantic pseudolabels) can easily be ported to other datasets and benchmarks. The nuScenes BEV segmentation data and the pseudolabels we generate are multilabel, i.e., each pixel can be associated to more than one class simultaneously. For training in nuScenes we use a balanced binary cross entropy loss:

$$\mathcal{L}_{\text{nuScenes}} = BBCE(\hat{S}, S), \tag{3.18}$$

i.e. PyTorch's BCEWithLogitsLoss with positive weights. We compute the weights for each class with the square root of the number of negative examples divided by the number of positive examples of each class in the perspective view pseudolabels for training with RendBEV. For the fine-tuning and training from scratch experiments, we get the number of positive and negative examples in the set of available labels and compute the weights accordingly.

## 3.4 Experiments

### 3.4.1 No BEV Supervision Experiments

In this subsection, we present the experiments in the setting without BEV supervision. To the best of our knowledge, there are no other learning-based methods capable of providing results without using any BEV label or pseudolabel. We use two unsupervised baseline approaches: one via an IPM [20], which we compute using the known camera height and extrinsic parameters. The computed warping is applied to semantic segmentation pseudolabels to generate the target BEV. We provide an additional baseline based on depth unprojection, similar to the one proposed in [22], but with new perspective pseudolabels (the same we use with our method, produced with GroundedSAM 2) and depth maps from Metric3D v2 for the nuScenes dataset. For the KITTI-360 dataset we use the depths predicted by Metric3D v2 and the pseudolabels from the Panoptic DeepLab model. Each pixel is unprojected to a 3D point using the depth map and the camera calibration matrix. Then, these 3D points are projected to the ground plane, and the

Table 3.1: Quantitative results – in IoU (%) – of methods using no BEV supervision on the nuScenes [53] validation set (PON split [22]). IPM and Depth Unproj. stand for Inverse Perspective Mapping and Depth Unprojection respectively."Mean" indicates IoU across all classes. Driv., Cross., Sw., Cprk., C.V., Motor., Ped. stand for Drivable area, Crosswalk, Sidewalk, Carpark, Construction Vehicle, Motorcycle and Pedestrian respectively. Multilabel segmentation setting.

| Method | Layout | | | | Object | | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Driv. | Cross. | Sw. | Cprk. | Bus | Bike | Car | C.V. | Motor. | Trailer | Truck | Ped. | Cone | Barrier | |
| IPM | 56.7 | 2.3 | 15.0 | 0.0 | 6.3 | 0.1 | 5.7 | 1.6 | 0.3 | 0.6 | 3.0 | 0.3 | 0.4 | 0.4 | 6.6 |
| Depth Unproj. | 50.8 | 2.6 | 16.1 | 0.0 | 7.3 | 2.1 | 13.8 | 7.8 | 1.4 | 1.1 | 4.3 | 3.5 | 1.7 | 2.1 | 8.2 |
| RendBEV (M3D v2) | **62.4** | **12.0** | **16.2** | 0.0 | **20.8** | **8.2** | **23.1** | **11.1** | **8.6** | **6.5** | **14.0** | **5.3** | **4.4** | **2.7** | **14.0** |

semantic BEV is constructed by assigning the corresponding semantic values in the perspective view pseudolabels to those points. Due to the high resolution of the BEV in KITTI-360, we construct a coarser BEV (1/4 of the target resolution) and then upsample with nearest neighbors in order to obtain a denser result. We do not perform any filtering, and collisions (multiple classes ending in the same BEV pixel after projection) are not explicitly handled. For evaluation, we apply the occlusion and field of view masks present in the ground truth labels to the result of all methods.

In Tab. 3.1 we show the quantitative results of RendBEV and baselines on the nuScenes dataset, accompanied by qualitative results in Fig. 3.6. RendBEV beats the unsupervised baselines by a significant margin. Although the IPM and the depth unprojection give acceptable results for the drivable area, RendBEV is able to overperform both. The depth unprojection suffers from sparsity in its results, especially in far areas, and the inability to provide the shape of objects aside from their enclosing surface. The IPM works well for the road but produces huge shadows for any surface not contained in the ground plane e.g. vehicles or pedestrians, as can be observed in Fig. 3.6. The carpark class is zero for all methods, as our perspective view pseudolabeling pipeline failed to produce any results for the class. We still consider it during evaluation, i.e. we average over all classes to compute the mIoU, including the zero value in carpark, to facilitate the comparison with others using the benchmark. In the most relevant motor vehicles (car, bus, construction vehicles and truck) our method is capable of producing results above 10% IoU even in this challenging scenario, outperforming both baselines. For smaller

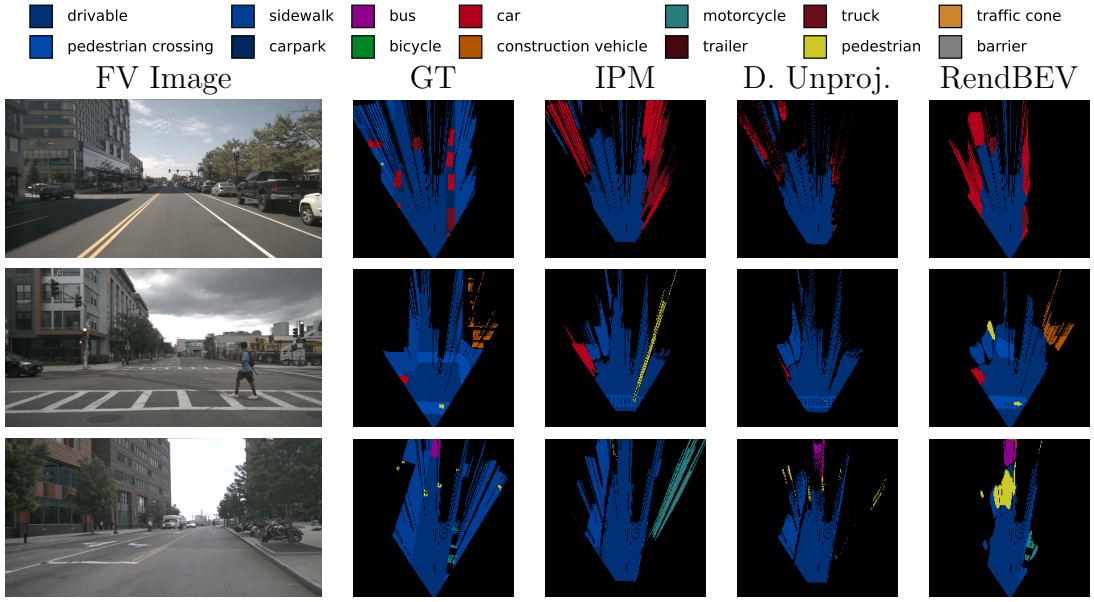


Figure 3.6: Qualitative results of a network trained with RendBEV and baselines for the setting with no BEV supervision in the nuScenes dataset. Field of view and occlusion masks from the ground truth are applied to all BEVs. PV stands for Perspective View, GT for Ground Truth, IPM for Inverse Perspective Mapping, Depth. Unproj. for Depth Unprojection, RendBEV corresponds to our method with Metric3Dv2 depth-based rendering. Best viewed with zoom.

vehicle classes (bike, motorcycle) which are intrinsically more difficult to segment – especially in the BEV as their footprint from above is always small, even if they are close to the camera, and thus comparatively bigger in the perspective view – the model trained with RendBEV provides about 8 % IoU while the baselines reach only about 2 %. Some common patterns in the behavior of the model when segmenting these objects can be observed in rows 2 and 3 of Fig. 3.6: objects closer to the camera and clearly delimited, such as the pedestrian crossing the street in the second row are properly segmented, but small localization errors hurt the IoU. Groups of these small objects, like the parked motorcycles to the right of the road in row 3 or the group of pedestrians standing and walking around a bus in the same image, get aggregated in a blob-like shape.

Tab. 3.2 contains the results for the no BEV supervision experiments in the KITTI-360 dataset, which we pair with qualitative results in Fig. 3.7. The data show similar trends to the ones observed in the previous table, with our model

Table 3.2: Quantitative results – in IoU (%) – using no BEV supervision on the KITTI-360 validation set. Depth Unproj. and IPM stand for Depth Unprojection and Inverse Perspective Mapping respectively.

| Method | Road | Sidewalk | Building | Terrain | Person | 2-Wheeler | Car | Truck | mIoU |
|---|---|---|---|---|---|---|---|---|---|
| IPM | 58.4 | 23.1 | 12.6 | 32.5 | 0.6 | 1.4 | 11.6 | 5.2 | 18.2 |
| Depth Unproj. | 40.8 | 18.4 | 16.8 | 23.1 | 2.9 | 2.2 | 17.6 | 5.3 | 15.9 |
| RendBEV (BtS) | 66.3 | 23.4 | 26.5 | 41.6 | 3.4 | 5.4 | **31.8** | 10.0 | 26.1 |
| RendBEV (M3D v2) | **67.3** | **25.2** | **27.3** | **42.1** | **4.8** | **6.4** | 31.3 | **10.1** | **26.8** |

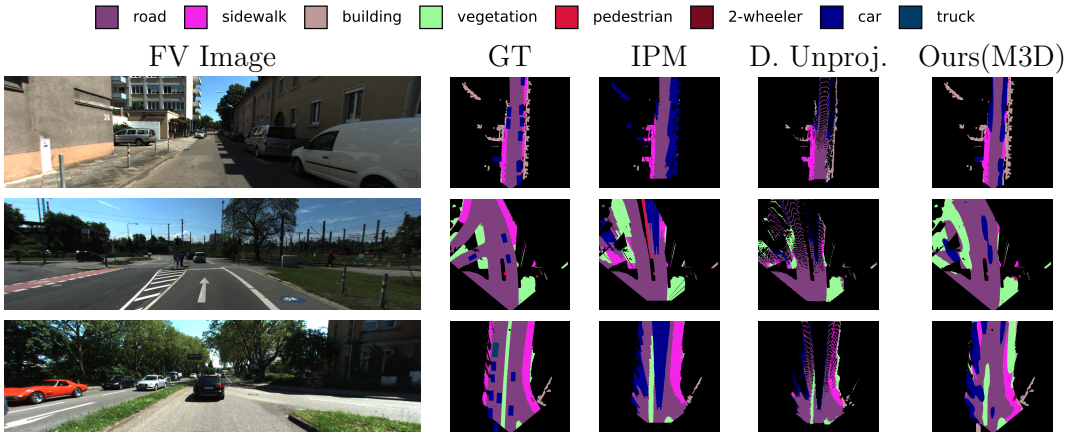


Figure 3.7: Qualitative results for the setting with no BEV supervision on the KITTI-360 dataset. FOV and unlabed pixel mask (from GT) applied to all BEVs.

beating the IPM and depth unprojection baselines, with a especially significant edge on the object classes. The depth unprojection baseline suffers from an even greater sparsity in this case, due to the resolution of the KITTI-360 images. These results also allow us to compare the performance provided by a model trained with RendBEV using the two different rendering strategies described in this work: the model using depth-based rendering and Metric3D v2 has slightly better general performance. Overall, the numerical results with both methods are very similar, showing that using a monocular depth estimation model to reconstruct the perspective views is a viable option, and thus unlocking the possibility of extending RendBEV to more scenarios.

Using Metric3D v2 and depth-based-rendering decreases computational constraints during training when compared to BehindTheScenes and volumetric rendering. A full 20-epoch training with Behind the Scenes takes about 8 days in

a single V100 GPU with 32GB memory, reduced to 1 day when training using depth-based-rendering with precomputed Metric3D v2 depths. Table 3.3 provides an overview of the compute profile of the training runs. The computation of these depth maps over all the frontal images in the KITTI-360 dataset takes approximately a day on the same hardware. At inference time, the network requires 1.1 GB of GPU memory and has an average latency of 103 ms in a V100 GPU (with a batch size of one image) for the nuScenes dataset. The measured average latency is 90.9 ms for the inference on KITTI-360 with a memory footprint of 2.1 GB.

Table 3.3: Training compute profile: peak memory, number of epochs, time to completion and batch size for each no BEV supervision experiment

| Experiment | Peak Mem. | Epochs | Time | BS |
|---|---|---|---|---|
| KITTI-360 (BtS) | 30.8 GB | 20 | 8 days | 5 |
| KITTI-360 (M3D v2) | 31.9 GB | 15 | 1.2 days | 8 |
| nuScenes (M3D v2) | 30.4 GB | 15 | 0.9 days | 8 |

### 3.4.2 Fine-tuning Experiments

With the aim of evaluating our method as a pretraining strategy when limited GT BEVs are available, we perform fine-tuning experiments at different degrees of annotated data availability, i.e., we first train a model using RendBEV and then fine-tune it using different subsets of the training dataset.

In Tab. 3.4 we report the results on the nuScenes dataset obtained using RendBEV as pretraining and then fine-tuning with different amounts of data available (as percentages of the nuScenes training set). To further compare our results, we also include the results of a semi-supervised learning methodology specifically designed for BEV segmentation, Semi-BEVseg [63], targeting training with limited amount of data in the same settings and other common semi-supervised learning baselines. Our results show that RendBEV is an effective pretraining methodology for low-annotated data regimes, providing better results in terms of overall performance than the semi-supervised learning methodology especially designed for this setting. Moreover, the best semi-supervised methodology (Semi-BEVseg), relying on data augmentations and a teacher-student framework during training is

mostly orthogonal to our methodology as it operates at a different level (during our "finetuning" stage) and through different mechanisms (data augmentations and symmetry). Both methods could potentially be combined by running RendBEV as pretraining and Semi-BEVseg as fine-tuning. The impact of the pretraining is particularly notable when lower amounts of data (5% of the training datasest or around 1700 annotated BEVs) are available.

Table 3.4: Quantitative results – in IoU (%) – of models trained with different percentages of data availability (percentage splits from Semi-BEVseg [63]) on the nuScenes [53] validation set (PON split [22]). "Mean" indicates IoU across all classes. Driv., Cross., Sw., Cprk., C.V., Motor., Ped. stand for Drivable area, Crosswalk, Sidewalk, Carpark, Construction Vehicle, Motorcycle and Pedestrian respectively.

| BEV (%) | Method | Layout | | | | Object | | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Driv. | Cross. | Sw. | Cprk. | Bus | Bike | Car | C.V. | Motor. | Trailer | Truck | Ped. | Cone | Barrier | |
| 5 | Π-Model [78] | 57.1 | 26.2 | 29.7 | 24.2 | 5.5 | 1.7 | 29.4 | **1.7** | 0.7 | 7.3 | 10.8 | 2.7 | 5.4 | 8.2 | 15.1 |
| | MT [79] | 56.7 | 26.8 | 30.4 | 25.3 | 6.8 | 1.2 | 30.4 | 0.9 | 0.5 | 7.5 | 11.4 | 3.0 | 6.9 | 8.4 | 15.4 |
| | CPS [80] | 57.0 | 25.4 | 29.4 | 24.0 | 5.9 | 0.3 | 29.3 | 0.3 | 0.1 | 6.6 | 10.5 | 2.0 | 5.0 | 7.8 | 14.5 |
| | UniMatch [81] | 57.0 | 25.2 | 29.3 | 24.0 | 5.7 | 0.3 | 29.2 | 0.4 | 0.1 | 6.4 | 10.4 | 1.8 | 4.8 | 7.2 | 14.4 |
| | Semi-BEVseg [63] | 59.3 | 29.8 | 33.8 | 26.1 | 9.8 | 2.7 | 34.6 | **1.5** | 1.6 | 10.7 | **14.7** | 5.9 | **9.6** | 12.5 | 18.1 |
| | RendBEV (M3D v2) | **71.1** | **30.4** | **35.4** | **31.9** | **28.8** | **8.0** | **36.3** | 0.7 | **7.8** | **11.1** | 14.1 | **7.4** | 7.0 | **18.2** | **22.0** |
| 10 | Π-Model [78] | 59.3 | 30.0 | 32.7 | 27.2 | 8.7 | 3.1 | 33.0 | 1.4 | 1.8 | 9.5 | 13.2 | 5.8 | 8.0 | 10.4 | 17.4 |
| | MT [79] | 58.6 | 29.7 | 31.8 | 27.4 | 8.9 | 2.8 | 33.7 | 1.9 | 1.8 | 11.4 | 15.6 | 5.1 | 9.4 | 10.8 | 17.8 |
| | CPS [80] | 58.7 | 27.0 | 30.3 | 26.0 | 7.7 | 1.3 | 32.0 | 1.2 | 0.5 | 10.1 | 14.6 | 3.0 | 8.0 | 8.7 | 16.4 |
| | UniMatch [81] | 58.8 | 27.2 | 30.2 | 26.4 | 8.0 | 1.2 | 32.4 | 1.7 | 0.7 | 10.0 | 14.7 | 3.2 | 8.5 | 8.9 | 16.6 |
| | Semi-BEVseg [63] | 60.8 | **31.9** | 35.7 | 27.4 | 13.8 | 5.2 | **36.4** | **2.8** | 4.0 | 13.9 | **17.3** | **7.8** | **11.4** | 12.9 | 20.1 |
| | RendBEV (M3D v2) | **71.7** | 31.6 | **36.4** | **32.5** | **29.0** | **9.1** | 36.2 | 1.7 | **9.0** | **18.5** | 14.5 | 6.3 | 4.8 | 14.3 | **22.5** |
| 20 | Π-Model [78] | 60.7 | 32.2 | 35.2 | 26.8 | 13.6 | 4.7 | 36.5 | 1.9 | 4.7 | 11.4 | 16.7 | 6.8 | 10.6 | 15.0 | 19.8 |
| | MT [79] | 60.4 | 32.8 | 36.0 | 29.8 | 11.9 | 4.4 | 36.2 | **4.4** | 3.7 | 12.9 | 17.6 | 7.8 | **11.1** | 15.1 | 20.3 |
| | CPS [80] | 59.4 | 31.7 | 35.2 | 29.2 | 10.9 | 3.5 | 35.0 | 3.6 | 2.7 | 12.0 | 17.0 | 7.0 | 10.2 | 13.6 | 18.4 |
| | UniMatch [81] | 59.4 | 31.3 | 35.0 | 29.0 | 10.5 | 3.2 | 35.0 | 3.5 | 2.6 | 11.8 | 16.8 | 6.8 | 10.0 | 13.6 | 18.2 |
| | Semi-BEVseg [63] | 61.5 | **34.0** | **37.0** | 30.6 | 16.8 | 6.9 | **38.5** | 3.4 | 6.7 | 14.3 | **20.4** | **9.7** | 10.7 | 15.8 | 21.9 |
| | RendBEV (M3D v2) | **72.0** | 32.3 | 36.4 | **31.8** | **25.3** | **7.1** | 37.4 | 0.9 | **8.7** | **17.4** | 17.6 | 7.3 | 8.2 | **20.7** | **23.1** |
| 40 | Π-Model [78] | 61.8 | 35.1 | 37.9 | 30.8 | 21.1 | 7.1 | 38.2 | 4.7 | 6.4 | 15.5 | 20.6 | 9.9 | 10.4 | 16.4 | 22.6 |
| | MT [79] | 61.5 | 34.9 | 37.9 | 31.6 | 18.5 | 8.3 | 38.4 | 3.2 | 7.6 | 16.2 | 20.0 | 10.5 | 10.9 | **16.7** | 22.6 |
| | CPS [80] | 59.5 | 33.0 | 36.0 | 29.5 | 17.2 | 6.2 | 37.3 | 1.6 | 5.7 | 13.1 | 18.0 | 8.5 | 8.9 | 15.0 | 20.6 |
| | UniMatch [81] | 59.6 | 33.0 | 35.7 | 29.4 | 17.0 | 6.0 | 37.3 | 1.6 | 5.6 | 12.8 | 18.0 | 8.4 | 8.9 | 15.0 | 20.5 |
| | Semi-BEVseg [63] | 62.8 | **36.0** | 38.9 | 31.5 | 21.6 | 7.6 | **39.6** | **5.1** | 6.8 | 18.3 | **22.7** | **11.1** | 11.1 | 16.1 | 23.5 |
| | RendBEV (M3D v2) | **73.1** | 33.1 | **39.0** | **34.3** | **28.5** | **9.7** | 38.1 | 2.7 | **9.3** | **20.7** | 19.9 | 7.0 | 3.8 | 16.5 | **24.0** |

Furthermore, in Tab. 3.5 we show the results obtained with a model pretrained with RendBEV (using both volumetric and depth-based rendering) in the KITTI-360 dataset and then fine-tuned on different percentages of available data. We compare it with the pretraining methodology proposed in the original SkyEye paper, where the architecture we are using was introduced. In both cases we show the results when pretraining is done using PV pseudolabels. We also show the results of a recent pretraining method, LetsMap. It is worth noting that the results of networks trained with SkyEye and RendBEV share the same underlying

Table 3.5: Quantitative results – in IoU (%) – of models trained with different percentages of data availability on the KITTI-360 validation set. Percentage splits defined in SkyEye [25].

| BEV (%) | Method | Road | Sidewalk | Building | Terrain | Person | 2-Wheeler | Car | Truck | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | LetsMap [61] | 72.9 | 37.8 | **43.7** | 38.3 | 0.9 | 2.6 | 30.6 | 10.9 | 29.7 |
| | SkyEye [25] | 70.7 | 31.1 | 32.4 | 40.1 | 0.0 | 0.0 | 29.1 | 4.0 | 25.9 |
| | RendBEV (BtS) | 73.2 | 37.2 | 36.1 | **46.8** | 4.3 | 7.5 | 37.7 | 13.8 | 32.1 |
| | RendBEV (M3D v2) | **75.5** | **39.6** | 36.6 | 44.9 | **6.6** | **8.5** | **42.9** | **15.0** | **33.7** |
| 10 | LetsMap [61] | 74.5 | **41.2** | **46.3** | 43.3 | 5.5 | **8.8** | 41.6 | **21.2** | **35.3** |
| | SkyEye [25] | 73.2 | 37.1 | 38.4 | 45.5 | 3.7 | 6.7 | 40.6 | 7.9 | 31.6 |
| | RendBEV (BtS) | 74.3 | 37.6 | 40.9 | 46.6 | 3.0 | 6.5 | 42.5 | 13.7 | 33.1 |
| | RendBEV (M3D v2) | **75.7** | **41.2** | 41.2 | **48.9** | **5.9** | 8.5 | **46.2** | 15.1 | **35.3** |
| 50 | LetsMap [61] | **76.5** | **42.7** | **49.2** | 41.5 | 3.4 | 8.6 | 38.8 | **19.4** | 35.0 |
| | SkyEye [25] | 72.5 | 36.9 | 39.4 | 45.1 | 3.6 | 7.5 | 41.2 | 9.7 | 32.0 |
| | RendBEV (BtS) | 73.6 | 38.9 | 42.5 | **47.6** | 3.7 | 5.9 | 43.4 | 13.3 | 33.6 |
| | RendBEV (M3D v2) | 74.7 | 40.4 | 39.8 | 46.7 | **7.7** | **11.6** | **47.0** | 13.6 | **35.2** |

architecture, making them more directly comparable than the results obtained by LetsMap, which uses a different one. In low-annotated data regimes, RendBEV provides higher performance boosts, beating SkyEye and LetsMap in mIoU. When the amount of available data is increased to 10 % of the KITTI-360 training set, RendBEV (using Metric3D v2) and LetsMap provide the best overall results. At 50 % of data from the KITTI-360 training dataset available, RendBEV(Metric3D v2) and LetsMap are the top performers with the overall performance being slightly better for RendBEV in this case.

Finally, we validate the effectiveness of our method as a pretraining strategy by training the same architecture we use in the previous experiments from scratch at different percentages of data availability and comparing it with models first trained with RendBEV and then fine-tuned on that same amount of data. In Tab. 3.6 we present the quantitative results for the nuScenes dataset. We provide some qualitative examples for the 5% and 20% splits to aid in the analysis of the results in Fig. 3.8. Pretraining with RendBEV yields a substantial boost in performance at all data splits, with a higher boost at lower data availability. Focusing in the split with lower amount of ground truth BEV labels (5%) we observe notable improvements in rarer classes such as pedestrian crossing, bus (from just 2.0 % IoU to 28.8 %), bike, motorcycle, pedestrian or trailer. This is illustrated e.g., in the second row of Fig. 3.8 where the pedestrian crossing structure is more complete and there are less hallucinated pedestrians and in the third row,

where the pretrained model predicted the far away vehicle on the left as bus while the model trained from scratch predicted it as car. The pretrained model was also able to segment 3 pedestrians (albeit not precisely located) while the model trained from scratch only hallucinated one at an incorrect location. With a higher amount of GT supervision e.g. 20 %, the performance boosts is less substantial, but the results are still overall better than those of the model trained from scratch, with the pretrained model being able to predict more difficult to segment objects such as the far away car on the left waiting in a pedestrian crossing in the first row.

We display the numerical results of experiments conducted in a similar fashion on the KITTI-360 dataset on Tab. 3.7: our method provides a high boost in performance in the 1% split (of more than 10 percentage points for both M3D and BtS) with the importance of the pretraining decreasing as more data becomes available. We present a comparison of qualitative results for the 1% and 10% splits in Fig. 3.9. The improvement in the results of the pretrained model with respect to the model trained from scratch is easy to visualize in Fig. 3.9, especially in the 1% split. The pretrained model is able to correctly perform important predictions close to the ego-vehicle that the model trained from scratch misses; clear examples are the parked car to the left in the first row, the pedestrian waiting to cross the street in front of the ego-vehicle on the second row or the biker under the shadow on the sidewalk on the right in the third row.

### 3.4.3 Ablation Studies

We investigate the effect of limiting the supervision to a single frame (the input one) instead of using a sequence, and of using a sequence composed of the current frame and the two immediately adjacent ones against using a longer sequence. We report results in Tab. 3.8. The best overall results are achieved with a longer sequence, encompassing eight frames, from the frame before the reference frame to the sixth after it. Using a shorter sequence composed of the reference frame and the immediately adjacent ones i.e., the previous and the following frames also provides better results than providing supervision by merely reconstructing the reference frame. We illustrate typical cases where the model trained with a sequence is

Table 3.6: Comparison of quantitative results – in IoU (%) – of models trained with different percentages of data availability (percentage splits from Semi-BEVseg [63]) from scratch and with RendBEV on the nuScenes [53] validation set (PON split [22]). "Mean" indicates IoU across all classes. Driv., Cross., Sw., Cprk., C.V., Motor., Ped. stand for Drivable area, Crosswalk, Sidewalk, Carpark, Construction Vehicle, Motorcycle and Pedestrian respectively. $\Delta$ is the difference between the mIoU of the model pretrained with that row's method and the same architecture trained from scratch.

| BEV (%) | Method | Layout | | | | Object | | | | | | | | | | Mean | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Driv. | Cross. | Sw. | Cprk. | Bus | Bike | Car | C.V. | Motor. | Trailer | Truck | Ped. | Cone | Barrier | | |
| 5 | from scratch | 67.1 | 18.1 | 27.3 | 29.9 | 2.0 | 0.1 | 27.5 | 0.3 | 0.5 | 1.8 | 7.6 | 1.2 | 1.7 | 11.7 | 14.1 | |
| | RendBEV (M3D v2) | **71.1** | **30.4** | **35.4** | **31.9** | **28.8** | **8.0** | **36.3** | **0.7** | **7.8** | **11.1** | **14.1** | **7.4** | **7.0** | **18.2** | **22.0** | +7.9 |
| 10 | from scratch | 70.2 | 26.9 | 30.7 | 30.9 | 14.4 | 0.1 | 30.3 | 0.0 | 1.2 | 6.0 | 10.4 | 3.5 | 2.9 | **18.4** | 17.6 | |
| | RendBEV (M3D v2) | **71.7** | **31.6** | **36.4** | **32.5** | **29.0** | **9.1** | **36.2** | **1.7** | **9.0** | **18.5** | **14.5** | **6.3** | **4.8** | 14.3 | **22.5** | +4.9 |
| 20 | from scratch | 71.2 | 30.2 | 33.2 | 28.9 | 10.5 | 0.4 | 33.2 | 0.0 | 3.0 | 14.6 | 13.7 | 3.5 | 4.9 | 16.8 | 18.9 | |
| | RendBEV (M3D v2) | **72.0** | **32.3** | **36.4** | **31.8** | **25.3** | **7.1** | **37.4** | **0.9** | **8.7** | **17.4** | **17.6** | **7.3** | **8.2** | **20.7** | **23.1** | +4.2 |
| 40 | from scratch | 71.5 | 26.0 | 33.8 | 26.1 | 10.9 | 2.0 | 31.5 | 0.7 | 2.8 | 11.9 | 13.9 | 3.8 | 3.4 | 14.9 | 18.1 | |
| | RendBEV (M3D v2) | **73.1** | **33.1** | **39.0** | **34.3** | **28.5** | **9.7** | **38.1** | **2.7** | **9.3** | **20.7** | **19.9** | **7.0** | **3.8** | **16.5** | **24.0** | +5.9 |

Table 3.7: Comparison of quantitative results – in IoU (%) – of models trained with different percentages of data availability on the KITTI-360 validation set with RendBEV and from scratch. $\Delta$ is the difference between the mIoU of an architecture trained with that row's method and the same architecture trained from scratch.

| BEV (%) | Method | Road | Sidewalk | Building | Terrain | Person | 2-Wheeler | Car | Truck | mIoU | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | from scratch | 61.0 | 22.7 | 27.8 | 23.7 | 0.0 | 0.0 | 31.3 | 6.3 | 21.6 | |
| | RendBEV (BtS) | 73.2 | 37.2 | 36.1 | **46.8** | 4.3 | 7.5 | 37.7 | 13.8 | 32.1 | +10.5 |
| | RendBEV (M3D v2) | **75.5** | **39.6** | **36.6** | 44.9 | **6.6** | **8.5** | **42.9** | **15.0** | **33.7** | **+12.1** |
| 10 | from scratch | 73.4 | 37.5 | 35.9 | 40.3 | 4.7 | 7.4 | 44.6 | 12.2 | 32.0 | |
| | RendBEV (BtS) | 74.3 | 37.6 | 40.9 | 46.6 | 3.0 | 6.5 | 42.5 | 13.7 | 33.1 | +1.1 |
| | RendBEV (M3D v2) | **75.7** | **41.2** | **41.2** | **48.9** | **5.9** | **8.5** | **46.2** | **15.1** | **35.3** | **+3.3** |
| 50 | from scratch | **75.3** | **40.6** | 41.8 | 45.3 | 2.9 | 6.6 | 45.5 | 13.5 | 33.9 | |
| | RendBEV (BtS) | 73.6 | 38.9 | **42.5** | **47.6** | 3.7 | 5.9 | 43.4 | 13.3 | 33.6 | -0.3 |
| | RendBEV (M3D v2) | 74.7 | 40.4 | 39.8 | 46.7 | **7.7** | **11.6** | **47.0** | **13.6** | **35.2** | **+1.3** |

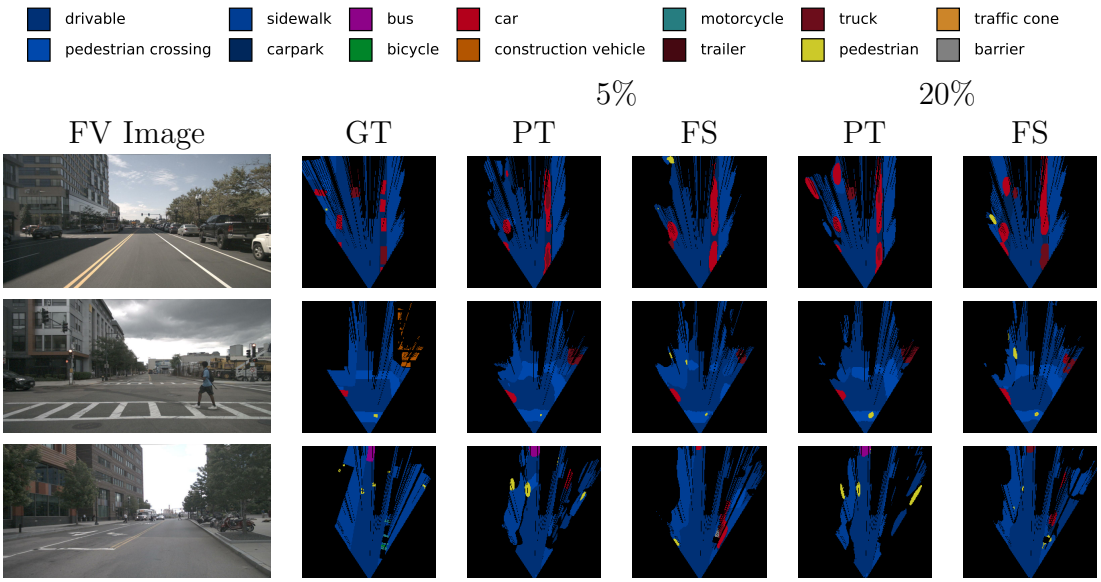


Figure 3.8: Qualitative results of models pretrained with RendBEV and then fine-tuned on 5% and 20% of the nuScenes training data and models trained from scratch on the same amount of data. PT stands for pretrained, FS stands for from scratch. FOV and occlusion masks applied to all BEVs.

able to perform better predictions in Fig. 3.10. Stronger supervision in far areas allows longer range predictions of the scene layout. In terms of computation, the training time for the three experiments is roughly the same (approximately 1 V100 GPU day), while the peak memory during training is 30.4GB, 27.7GB and 25.4GB for the models using the full sequence, the current and two immediately adjacent frames and just the current frame respectively.

We execute a supplementary set of experiments to validate RendBEV with a different architecture for the BEV semantic segmentation network. To this end, we modify Simple-BEV [30] and adapt it to our setting, by making it work with monocular frontal images only, increasing the number of classes in its semantic segmentation head and removing the auxiliary task heads. We train the network in the KITTI-360 dataset using the RendBEV method (with volumetric rendering and BehindTheScenes) and then fine-tune the model at different percentage splits of the dataset. To provide a baseline comparison, we train the same model from scratch on the same splits. We present the results obtained in these experiments in Tab. 3.9. The performance we reach while using RendBEV as a standalone

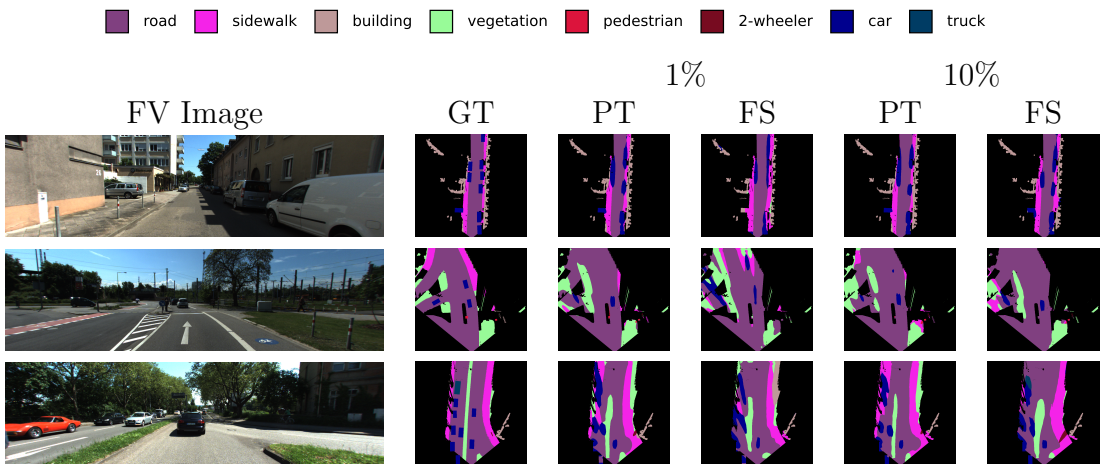


Figure 3.9: Qualitative results of models pretrained with RendBEV and then fine-tuned on 1% and 10% of the KITTI-360 training data and models trained from scratch on the same amount of data. PT stands for pretrained, FS stands for from scratch.

training is inferior to the one obtained with SkyEye's architecture as BEV semantic segmentation network, but the applicability of the method holds. When fine-tuning on available ground truth, RendBEV proves to be useful as pretraining in the lower annotation regimes. When the amount of ground truth data is high (in the 50% and 100% splits) the pretrained models obtain overall performances almost equal to the ones trained from scratch in terms of mIoU.

Table 3.8: Comparison of quantitative results – in IoU (%) – of RendBEV with a single frame or a sequence using no BEV supervision on the nuScenes [53] validation set (PON split [22]). "Mean" indicates IoU across all classes. Driv., Cross., Sw., Cprk., C.V., Motor., Ped. stand for Drivable area, Crosswalk, Sidewalk, Carpark, Construction Vehicle, Motorcycle and Pedestrian respectively. Multilabel segmentation setting.

| Setting | Layout | | | | Object | | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Driv. | Cross. | Sw. | Cprk. | Bus | Bike | Car | C.V. | Motor. | Trailer | Truck | Ped. | Cone | Barrier | |
| $t' = t$ | 53.7 | **12.5** | 14.3 | 0.0 | 21.1 | 4.5 | **27.4** | 7.6 | 8.9 | 3.4 | 14.5 | 5.3 | 2.5 | 2.5 | 12.7 |
| $t' \in \{t-1, \dots, t+1\}$ | 60.8 | 6.3 | **19.1** | 0.0 | **23.8** | 8.1 | 27.3 | 6.7 | **9.7** | 5.4 | **15.4** | **6.1** | 3.3 | 1.7 | 13.8 |
| $t' \in \{t-1, \dots, t+6\}$ | **62.4** | 12.0 | 16.2 | 0.0 | 20.8 | **8.2** | 23.1 | **11.1** | 8.6 | **6.5** | 14.0 | 5.3 | **4.4** | **2.7** | **14.0** |

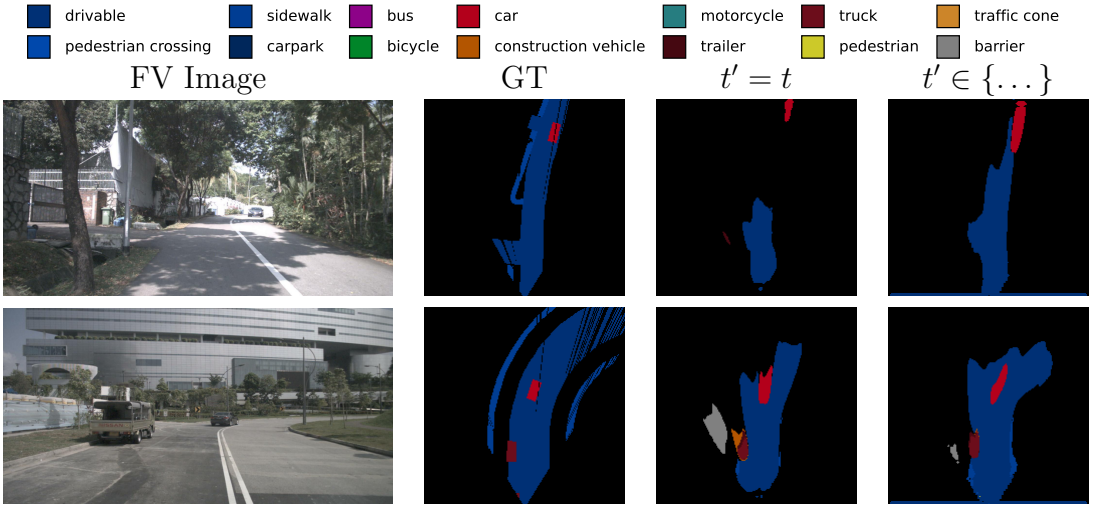


Figure 3.10: Qualitative results of models with no BEV supervision trained with the target equal to the input image $t' = t$ or with targets along a sequence $t' \in \{t-1, \dots, t+6\}$. Best viewed with zoom.

### 3.4.4 Robustness and Limitations

#### Robustness

We investigate the robustness of RendBEV to perturbations to the estimated depth. We do so by running Metric3D v2 Small (DINO2reg-ViT-Small backbone) to extract depth maps for the nuScenes dataset, and training a model using them instead of the Metric3D v2 Giant used in the original experiment. In addition, we run an experiment in which we apply additive Gaussian noise with zero mean and

Table 3.9: Study of the performance of RendBEV with Simple-BEV as BEV semantic segmentation network at different annotated data regimes. All scores are reported in the KITTI-360 dataset.

| BEV (%) | Method | Road | Sidewalk | Building | Terrain | Person | 2-Wheeler | Car | Truck | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | RendBEV | 65.5 | 30.3 | 29.5 | 38.46 | 1.9 | 2.5 | 30.9 | 7.2 | 25.8 |
| 1 | – | 57.5 | 23.12 | 19.3 | 21.4 | 0.1 | 0.1 | 18.2 | 1.5 | 17.7 |
| | RendBEV | **68.8** | **34.7** | **32.8** | **38.7** | **2.2** | **3.1** | **34.3** | **5.2** | **27.5** |
| 10 | – | 70.4 | 34.4 | 30.3 | 35.4 | 0.3 | 0.8 | 34.4 | **10.0** | 27.0 |
| | RendBEV | **70.7** | **36.1** | **36.3** | **40.0** | **1.7** | **4.9** | **35.8** | 5.7 | **28.9** |
| 50 | – | **72.1** | 35.5 | 34.9 | 37.4 | 1.0 | 1.5 | **38.6** | **11.6** | 29.1 |
| | RendBEV | 70.7 | **36.0** | **36.7** | **40.4** | **1.7** | **5.2** | 36.6 | 6.1 | **29.2** |

a standard deviation of 0.5m to every depth value during training.

We report the results in Tab. 3.10. We observe a very limited performance degradation, slightly higher when substituting the depth maps with the ones coming from the smaller model. These results suggest that our model is robust to mild perturbations in depth. Computing depth maps for the 34,139 training images from nuScenes requires 4 GPU hours with the Small Metric3D v2 model compared to 24 GPU hours with the Giant one. Using the smaller model could be an alternative when scaling to very large datasets and/or under tight computational budgets.

In order to simulate less precise camera poses (which could be the output of a visual inertial odometry or SLAM pipeline), we run four different experiments in which we apply a random translation and/or rotation to each of the poses in a sequence (the noise is different for each sequence element). Each of the components of the translation vector is sampled independently from a Gaussian distribution with zero mean and standard deviation $\sigma = 0.1$m or $\sigma = 0.5$m. For the rotation noise, we sample each of the three angles independently from a Gaussian distribution with zero mean and standard deviation $\sigma = 1°$ or $\sigma = 2.5°$. We report the results in Tab. 3.11. We observe small performance losses when only applying translation noise. Adding rotation noise on top of the translation noise has a more pronounced impact on performance, which indicates that our method is more susceptible to noise in the camera orientation than in its position.

Table 3.10: Quantitative results – in IoU (%) – of RendBEV under different perturbations to the depth. M3D v2-G → M3D v2-S means that depth maps from Metric3D v2 Small are used instead of the M3D v2 Giant ones used in the original experiment. $\pm\mathcal{G}(\sigma)$ represents additive Gaussian noise with zero mean and standard deviation $\sigma$. Results using no BEV supervision on the nuScenes [**?**] validation set (PON split [22]). "Mean" indicates IoU across all classes. Driv., Cross., Sw., Cprk., C.V., Motor., Ped. stand for Drivable area, Crosswalk, Sidewalk, Carpark, Construction Vehicle, Motorcycle and Pedestrian respectively. Multilabel segmentation setting.

| Depth perturbation | Layout | | | | | Object | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Driv. | Cross. | Sw. | Cprk. | Bus | Bike | Car | C.V. | Motor. | Trailer | Truck | Ped. | Cone | Barrier | |
| ✗ | 62.4 | 12.0 | 16.2 | 0.0 | 20.8 | **8.2** | 23.1 | **11.1** | 8.6 | **6.5** | **14.0** | 5.3 | 4.4 | 2.7 | **14.0** |
| M3D v2-G → M3D v2-S | 59.6 | **15.9** | **17.2** | 0.0 | **22.1** | 3.8 | 25.5 | 10.6 | **9.4** | 2.7 | 13.9 | **5.5** | 3.2 | 1.5 | 13.6 |
| $\pm\mathcal{G}(\sigma = 0.5)$ | **65.3** | 13.1 | 13.6 | 0.0 | 20.9 | 6.0 | **25.7** | 10.1 | 9.2 | 4.0 | 11.6 | 5.0 | **5.5** | **3.0** | 13.8 |

Table 3.11: Quantitative results – in IoU (%) – of RendBEV with noise applied to the poses. $\mathcal{G}(\sigma)$ represents Gaussian noise with zero mean and standard deviation $\sigma$. Results using no BEV supervision on the nuScenes [53] validation set (PON split [22]). "Mean" indicates IoU across all classes. Driv., Cross., Sw., Cprk., C.V., Motor., Ped. stand for Drivable area, Crosswalk, Sidewalk, Carpark, Construction Vehicle, Motorcycle and Pedestrian respectively.

| Pose perturbation | | Layout | | | | Object | | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Translation | Rotation | Driv. | Cross. | Sw. | Cprk. | Bus | Bike | Car | C.V. | Motor. | Trailer | Truck | Ped. | Cone | Barrier | |
| ✗ | ✗ | 62.4 | 12.0 | 16.2 | 0.0 | 20.8 | **8.2** | 23.1 | **11.1** | 8.6 | **6.5** | **14.0** | 5.3 | 4.4 | 2.7 | **14.0** |
| $\mathcal{G}(\sigma = 0.1\text{m})$ | ✗ | 63.3 | 11.8 | 18.1 | 0.0 | **27.0** | 6.9 | 24.2 | 5.9 | 7.7 | 3.0 | 12.5 | **5.5** | 3.8 | 2.2 | 13.7 |
| $\mathcal{G}(\sigma = 0.5\text{m})$ | ✗ | **65.3** | 13.1 | 13.6 | 0.0 | 20.9 | 6.0 | **25.7** | 10.1 | 9.2 | 4.0 | 11.6 | 5.0 | **5.5** | **3.0** | 13.8 |
| $\mathcal{G}(\sigma = 0.1\text{m})$ | $\mathcal{G}(\sigma = 1°)$ | 63.5 | 15.4 | **18.9** | 0.0 | 21.9 | 5.5 | 21.3 | 6.2 | **10.0** | 4.9 | 10.8 | 3.8 | 1.9 | 1.9 | 13.3 |
| $\mathcal{G}(\sigma = 0.5\text{m})$ | $\mathcal{G}(\sigma = 2.5°)$ | 61.8 | **15.6** | 14.0 | 0.0 | 17.9 | 4.5 | 19.0 | 8.3 | 8.2 | 3.5 | 11.9 | 3.6 | 2.8 | 1.4 | 12.3 |

Table 3.12: Quantitative results – in IoU (%) – using no BEV supervision on the KITTI-360 validation set when using Ground Truth Perspective View (GT PV) semantic segmentation and pseudolabels from an off-the-shelf model.

| Method | GT PV | Road | Sidewalk | Building | Terrain | Person | 2-Wheeler | Car | Truck | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|
| RendBEV (M3Dv2) | ✓ | **68.6** | **33.5** | **29.3** | **43.7** | **5.3** | 5.3 | **31.8** | 8.3 | **28.2** |
| RendBEV (M3Dv2) | ✗ | 67.3 | 25.2 | 27.3 | 42.1 | 4.8 | **6.4** | 31.3 | **10.1** | 26.8 |

We evaluate the robustness of RendBEV (with depth-based rendering and Metric3D v2) to the quality of the perspective view semantic segmentation supervision by performing a training on the KITTI-360 dataset – where ground truth perspective view segmentation is available – and comparing it with the results obtained when training with noisy pseudolabels obtained with an off-the-shelf model. We report the results in Tab. 3.12. Using ground truth semantic segmentation in the perspective view slightly increases performance, although the only class with a significant increase in performance is sidewalk. The usage of perspective view pseudolabels from an off-the-shelf model is thus a viable option, although ground truth labels provide the best downstream performance if available.

### Limitations

The main limitation of networks trained with our method is the performance on moving objects, especially in scenes crowded with them. Indeed, using multiple frames in training improves general performance, as seen in Tab. 3.8, but relies on a the assumption of a locally static scene, that might not always be valid. In

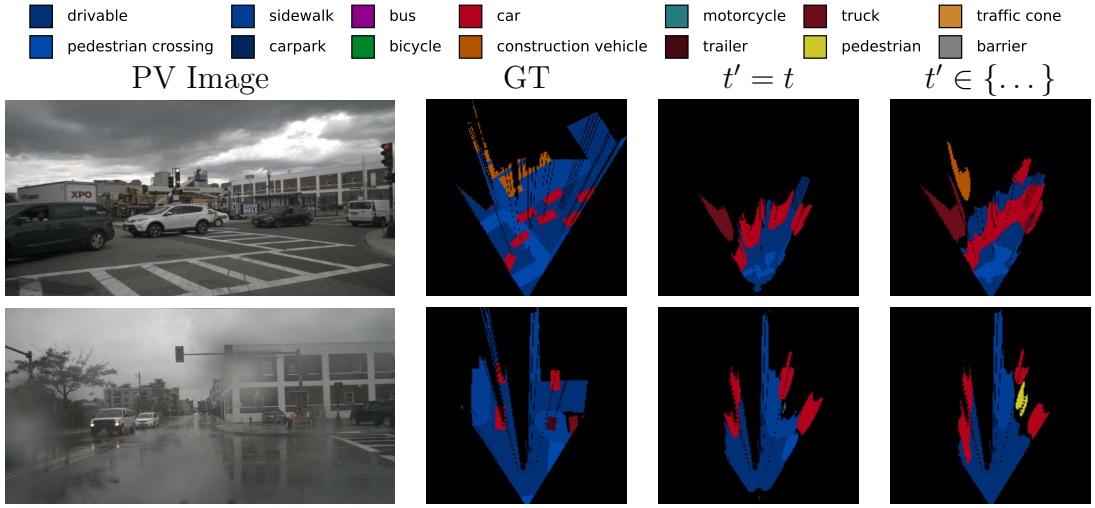


Figure 3.11: Qualitative results of RendBEV (Metric3D v2) on the nuScenes dataset with no BEV supervision in the standard setting $t' \in \{t-1, \dots, t+6\}$ in dynamic scenes. Ground Truth (GT) and results when only rendering the present frame during training $t' = t$ (and thus dropping the static scene assumption) shown for reference.

Fig. 3.11 we show the behavior of a network trained with RendBEV when dealing with highly dynamic scenes. We provide the results of a network trained only by rendering the reference frame $t$ instead of multiple frames. In the first row, we can view a failure case on dynamic objects in the bird's eye view segmentation, where vehicles are blurred in the direction of their perceived movement. When a group of moving vehicles are close together, this results in the individual vehicles merging into a single blob. This does not happen when only a single reference frame is used, although other problems are present, such as worse segmentation through occlusions and in the longer range, as well as stronger shadow effects in the depth direction. In the second row, we can observe similar behavior on an input image with adverse weather conditions and moving objects in the longitudinal direction instead of the traversal direction.

## 3.5 Conclusion

In this chapter we have presented RendBEV, a method for training bird's eye view semantic segmentation models without using bird's eye view ground truth labels. Contrary to similar works, we don't rely on precomputed pseudolabels. Our method relies on shifting the supervision to the perspective view by rendering it with values sampled from the output of the bird's eye view network to be trained. The rendering pipeline exploits either a pretrained neural density field or a foundation monocular depth model to provide information on the scene geometry. Extensive experimental work on the nuScenes and KITTI-360 dataset validates our approach. We have also shown how RendBEV can be used as pretraining when limited BEV ground truth labels are available, leading to a significant boost in performance compared to training from scratch with those labels, and beating or being on-par with competing methods. The main limitation of our method is in dealing with dynamic object and scenes, which we plan to address in future work.

## 3.6 Future Work

Several research ideas and challenges related to RendBEV have remained unexplored, or the results we obtained are not mature enough for discussion in the main chapter body. We summarize them here.

**Motion modeling.** As we have stated, dealing with dynamic objects and scenes is the main limitation of our RendBEV training methodology. We hypothesize that dropping the static scene assumption while modeling object motion could lead to increased performance and robustness in dynamic scenes. While other works have explored explicit motion modeling in the bird's eye view under full supervision, applying this without direct bird's eye view supervision remains an open challenge. A possible way to model this motion would be with a bird's eye view flow field. This flow could be predicted by the bird's eye view network and will allow to link timesteps enabling to use RendBEV without the static scene assumption. It could be explicitly supervised by using two-dimensional optical flow pseudolabels provided by an off-the shelf model in a similar way to which the semantics are

supervised. Another possibility, which could be combined with the previous one, would be the prediction of the BEV semantic segmentation of future timesteps as well as the present one. Each future timestep would be supervised only by its corresponding perspective view, or by multiple views combined with the flows and a consistency constraint. These ideas, aside from the potential to improve performance on dynamic objects, would enable the prediction of the future position of objects. This future prediction is very useful for downstream tasks which can enhance safety in driving such as collision avoidance and accident anticipation.

**Explicit object modeling.** A limitation of RendBEV is the lack of definition in the shape of objects and overall "blobiness" of object shapes. Integrating priors on the shape of objects, e.g. a car almost always being an oriented rectangle in the bird's eye view, is not easy in a semantic segmentation framework, without the concept of objects. Segmentation networks naturally learn this shape from data when supervised with ground truth labels, but this is not the case with the supervision provided by RendBEV. Explicitly considering objects as separate entities would help in integrating these priors. One example is to switch from a semantic segmentation framework to an object detection framework. If the task is cast as predicting oriented bounding box parameters by regression, the shape prior is implicit. This is indeed one of the most popular tasks in the bird's eye view perception literature (BEV or 3D object detection). However, then it is not straightforward to lift the supervision to the perspective view to avoid having to use ground truth BEV supervision as in RendBEV. One possibility of doing this would be to differentiably rasterize the predicted oriented bounding boxes and then sample the raster to produce the semantic perspective views.

# Chapter 4

# A Synthetic Dataset for Viewpoint Shift Robustness

Computer vision models for tasks related to assisted and automated driving systems as well as driving scene understanding, including bird's eye view segmentation, but also object detection, or depth estimation are expected to work reliably in different domains and on a long tail of challenging conditions.

In many practical scenarios, due to either the lack of a sufficient amount of data or to the difficulties and costs associated to data annotation, it is common to use *off-the-shelf* models trained on public datasets, possibly fine-tuned on a small dataset for the target domain. However, the target domain can be significantly different from the one on which models have been developed and evaluated, and the performance gap can be wide.

The largest and most popular driving datasets, such as KITTI [52], nuScenes [53], Waymo [55], Cityscapes [74], are acquired either from a single vehicle with a very specific camera setup, or from multiple vehicles with a camera setup which is as close as possible to a single pre-defined one. This single-setup acquisition introduces an intrinsic domain shift problem when switching to applications where the installation setup of a camera or the vehicle type, and, thus, the *viewpoint*, may be significantly different from the one in existing public datasets on which models are typically trained.

Indeed, in a real-world application, a frontal camera could be installed in several

Figure 4.1: Video frames from real-world frontal dashcams (Verizon Connect) with different viewpoints.

different positions: if it is an integrated camera, it is often behind the rear-view mirror, while dashcams can be installed virtually anywhere on the windshield or on the dashboard. The installation height is also directly related to the type of vehicle: comparing passenger cars with particularly tall vehicles, such as heavy duty trucks, the difference between the camera height could be in the order of 2 meters. Another source of variability is the camera orientation, which can vary depending on the vehicle (cameras on tall vehicles need to point down to capture footage in front of the vehicle) and the quality of installation; this quality might be suboptimal e.g., for aftermarket devices like dashcams, where the installation is often performed by non-professionals. As a result, different camera setups can have different yaw, pitch and roll angles, height from the ground, and horizontal displacement, as shown in Fig. 4.1.

Although other domain shifts on driving scenes, such as adverse weather conditions or image corruptions, have been studied in the scientific literature [82], the robustness of neural network models to camera viewpoint shift is an understudied problem in the literature with the exception of the concurrent work [57]. Building annotated datasets with data from multiple viewpoints covering most of the

scenarios that might come up in practice is difficult, costly and time-consuming. Such difficulties, however, can be overcomed with synthetic data, which allows us to control for confounders (other sources of uncertainty or domain shift), isolate the effect of the viewpoint shift, and streamline evaluation.

The main contributions of the work in this chapter are the following:

- We present VS-Sim[1], a dataset of synthetic driving scenes acquired from different viewpoints, with the aim of raising awareness on the problem and speeding up scientific research towards *viewpoint robustness*.
- We estimate the impact of the viewpoint shift in driving scenes on bird's eye view semantic segmentation, in which the position of the camera with respect to the scene is crucial.
- We show that a BEV semantic segmentation model trained on a single viewpoint transfers poorly to different ones, especially if the pitch angle changes, and that training on multiple viewpoints yields a model that generalizes better on viewpoints that were not used during training. This indicates that including training data from diverse viewpoints is a possible solution to mitigate performance degradation from viewpoint shifts.

## 4.1 Related Work

### 4.1.1 Domain Generalization in Road Scenes

Domain adaptation and domain generalization are well-known problems for driving adjacent tasks. Driving scenes in the wild are known to have a wide range of conditions and contain a long tail of non-nominal events that may introduce a significant domain shift with respect to popular datasets: examples are adverse weather conditions [83], rare safety-critical scenarios [84], as well as different scene geometry [57, 85]. A simple yet effective approach is to accurately curate datasets so that they better align to the data distribution of the target domain, as well as using data augmentation techniques to cope with the scarcity of such data [83, 86].

[1]The link to download the dataset can be found in the project page: henrique.gal/VS-SIM

This is not always possible or sufficient, and there is a wide body of research on techniques to either adapt to a target domain [87,88] or to train models with better generalization capabilities [89].

### 4.1.2 Datasets with Multiple Viewpoints

Compared to other sources of domain shift, such as e.g. weather or lighting conditions, capturing data with different viewpoints requires changing the physical position and/or orientation of the sensors, which might be challenging in many cases. Most driving image datasets, such as nuScenes [53], KITTI-360 [54] or Waymo [55] contain data acquired from rigs with several viewpoints that provide a 360° coverage of the scene. However, these data cannot be directly used to study the robustness to viewpoint shifts of cameras that have the same function (e.g., frontal view). If we want to study the performance of a model trained on images from a frontal view camera installed in a sedan car in optimal conditions (no roll and yaw, low pitch) when tested on a suboptimal installation (e.g. a slightly rotated camera) or images from a camera installed in a different vehicle type (e.g. a van, SUV, or heavy duty truck instead of a sedan), the necessary data is different than the surround camera footage in the aforementioned datasets.

Few real-world datasets have driving scenes from multiple frontal viewpoints. A notable example is Mapillary Vistas [90], that contains road images from an extremely diverse set of cameras across the world, with very diverse geographies, environmental conditions and viewpoints. The only available annotations are for perpective view semantic segmentation. BDD100K [91] includes 100k annotated images captured by mobile phones used as dashcams. The dataset contains annotations for object detection and some semantic segmentation tasks. Note that in both BDD100K and Mapillary Vistas, camera extrinsics are not known.

Several synthetic datasets in the literature include data acquired from different viewpoints, although robustness to viewpoint shifts is usually not the main focus of the dataset. Virtual KITTI [92] is a synthetic replica of KITTI [52], that includes some variations of the orientation of frontal camera ($\pm 15°$ in the yaw angle). IDDA [93] has data from 5 different vehicles. The main difference is the camera height and the shape of the vehicle hood that is visible in the image. SYN-

THIA [94] is another large-scale sythetic dataset focused on semantic segmentation of urban scenes. It includes 2 frontal cameras with a horizontal displacement, and images from other 6 side/rear cameras. A notable exception is the dataset described in [57], concurrent work to the content of this chapter, with synthetic data for a total of 36 frontal viewpoints, with perturbations in pitch, yaw, height and depth.

### 4.1.3 Viewpoint Robustness

Several articles have shown that tasks such as object detection or semantic segmentation are generally quite robust to variability in camera pose, although there might be some degradation of performance due to the different appearance that some objects might have from orientations not seen in the training set. [92] reports experiments on Virtual-KITTI around multiple object tracking with two non-standard camera orientations ($\pm 15^\circ$ in the yaw angle), showing a degradation of $\sim 5\%$ in Multiple Object Tracking Accuracy. The work in [93] reports experiments on IDDA about the impact of viewpoint shifts on semantic segmentation of the frontal view image. Their results show that semantic segmentation predictions for an unseen viewpoint are mainly affected by the presence of a different vehicle hood, whose pixels are misclassified. Overall, models do not appear to be heavily affected by the viewpoint shift (much less than other perturbations such as weather), nor do domain adaptation methods improve much (up to 4%). The work in [95] shows that semantic segmentation does not degrade much, on average around 3% of mIoU, also in very different viewpoints from what was seen during training. The work in [85] proposes a domain adaptation method to reduce the gap due to viewpoint shift for the detection of pedestrians in different real-world datasets (Cityscapes [96] and MOT [97]) in which they appear under significantly different perspectives.

Arguably, the viewpoint shift is more relevant for tasks that require some kind of 3D understanding of the world, such as depth estimation or bird's eye view reconstruction. In the monocular depth estimation community, much effort in the last few years has gone towards foundation models that work zero-shot to estimate depth across multiple domains and geometries [98–100], though we know

of no work that attempts to directly measure the effect of different viewpoints. In tasks where the aim is to reconstruct a bird's eye view representation, to the best of our knowledge the only relevant existing work is [57] (concurrent work), in which the authors measure the impact of viewpoint shift in BEV reconstruction of surrounding vehicles. On their synthetic dataset, they train a model on a standard "source" viewpoint and test it on data from several different viewpoints, unseen during training. They show how a small change in the viewpoint significantly affects BEV reconstruction. Moreover, they propose a viewpoint augmentation technique involving novel view synthesis to transform images to a specific target viewpoint.

In this work, we study *viewpoint robustness* in bird's eye view semantic segmentation by measuring the performance drop due to viewpoint shift in several realistic positions for camera installations. Complementing concurrent work [57], we include the case of a non-negligible roll angle, and consider multiclass dense BEV semantic segmentation rather than only the segmentation of vehicles. We look at the problem through the lens of domain generalization, rather than domain adaptation as in [57], thus under the assumption that the target domain is not known. We show that our baseline approach, training a BEV model on a mix of different viewpoints, helps to obtain a model with better generalization capabilities.

## 4.2 The VS-Sim Dataset

We generate the VS-Sim dataset using CARLA [56], a popular open-source simulator for research on automated driving systems based on the Unreal engine. We use three types of virtual sensors to acquire the raw RGB data and the annotations: the RGB camera, acting as a road-facing dashcam to collect frontal view images; the depth camera, which generates depth maps corresponding to the RGB images; the semantic segmentation camera, used to generate semantic segmentation ground truth both for perspective views and for the bird's eye view. An example of a RGB frame and the corresponding annotations from the other virtual sensors are presented in Fig. 4.2.

We place frontal view cameras inside the cabin of the ego-vehicle, in positions

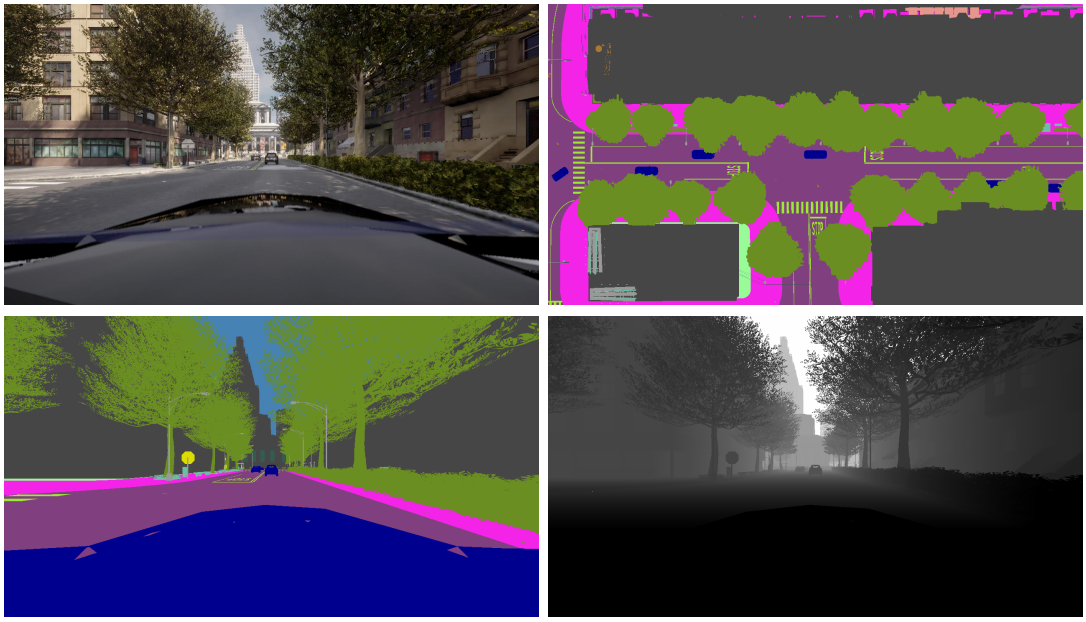

Figure 4.2: Frontal RGB image and pixel-level ground truth for BEV semantic segmentation, semantic segmentation, depth.

that commonly occur in real Verizon Connect's dashcam installations: one in a default position (top-center, above the rear-view mirror), one on the top-left corner of the windshield, and one at the bottom touching the dashboard. Moreover, we include some common angle perturbations: a roll by 5° to the right and four different pitch rotations (-10°, -5°, 5°, 10°) with respect to the default position. For each position, we capture RGB, depth and semantic segmentation. All the virtual cameras simultaneously record the same scene. The available viewpoints are summarized in Fig. 4.3 and Table 4.1.

We configure the intrinsic parameters of the frontal cameras to resemble the dashcams deployed by Verizon Connect, with a resolution of $1280 \times 720$ and a horizontal field of view of 120°. In Fig. 4.4, we report a sample set of images with the various viewpoints. To capture BEV ground truth, we position a semantic segmentation camera with the same resolution and a FOV of 10° at a height of 650 meters above the scene. With this configuration, we aim to minimize perspective distortion and approximate an orthographic camera, which is not available in CARLA. Semantic segmentation masks contain pixel-level information for a

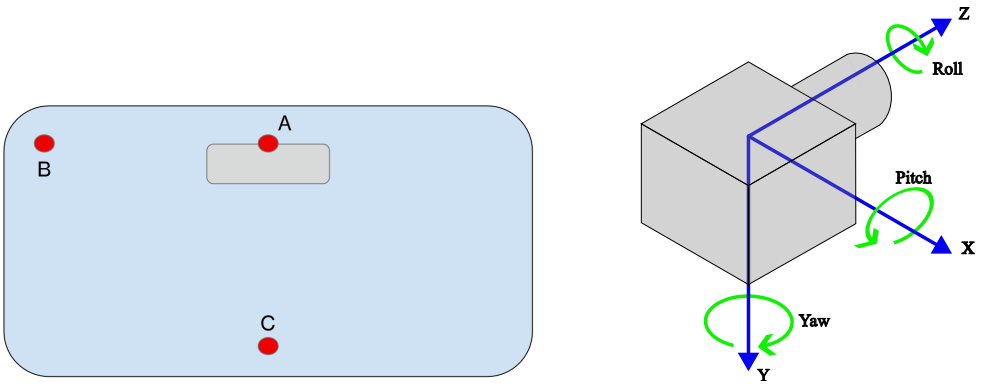


Figure 4.3: Positions of the cameras on the windshield and reference system.

Table 4.1: Summary of the available viewpoints.

| Name | Position | Roll | Pitch | Yaw |
|---|---|---|---|---|
| Default | Top-center of windshield (A) | 0 | 0 | 0 |
| Bottom | Bottom center of windshield (C) | 0 | 0 | 0 |
| Top-Left | Top-left of windshield (B) | 0 | 0 | 0 |
| Roll | Top-center of windshield (A) | $+5°$ | 0 | 0 |
| Pitch$^{--}$ | Top-center of windshield (A) | 0 | $-10°$ | 0 |
| Pitch$^{-}$ | Top-center of windshield (A) | 0 | $-5°$ | 0 |
| Pitch$^{+}$ | Top-center of windshield (A) | 0 | $+5°$ | 0 |
| Pitch$^{++}$ | Top-center of windshield (A) | 0 | $+10°$ | 0 |

total of 11 classes: road, sidewalk, building, wall, vegetation, terrain, occlusion, person, two-wheeler, car, and truck, analogous to categories in similar real-world datasets [52, 53].

To provide sufficient diversity, data are captured in seven different CARLA maps, namely Town 1, 2, 3, 4, 5, 7, and 10. For each map, we generate 100 different 20-second trajectories, varying the position of the vehicle in the map and the surrounding objects. In total, using a sensor sampling rate of 1Hz, the dataset consists of a total of 14,000 samples for each of the 8 viewpoints, each one including RGB images and frontal view annotations (depth and semantic segmentation), and a mask with BEV semantic segmentation ground truth.

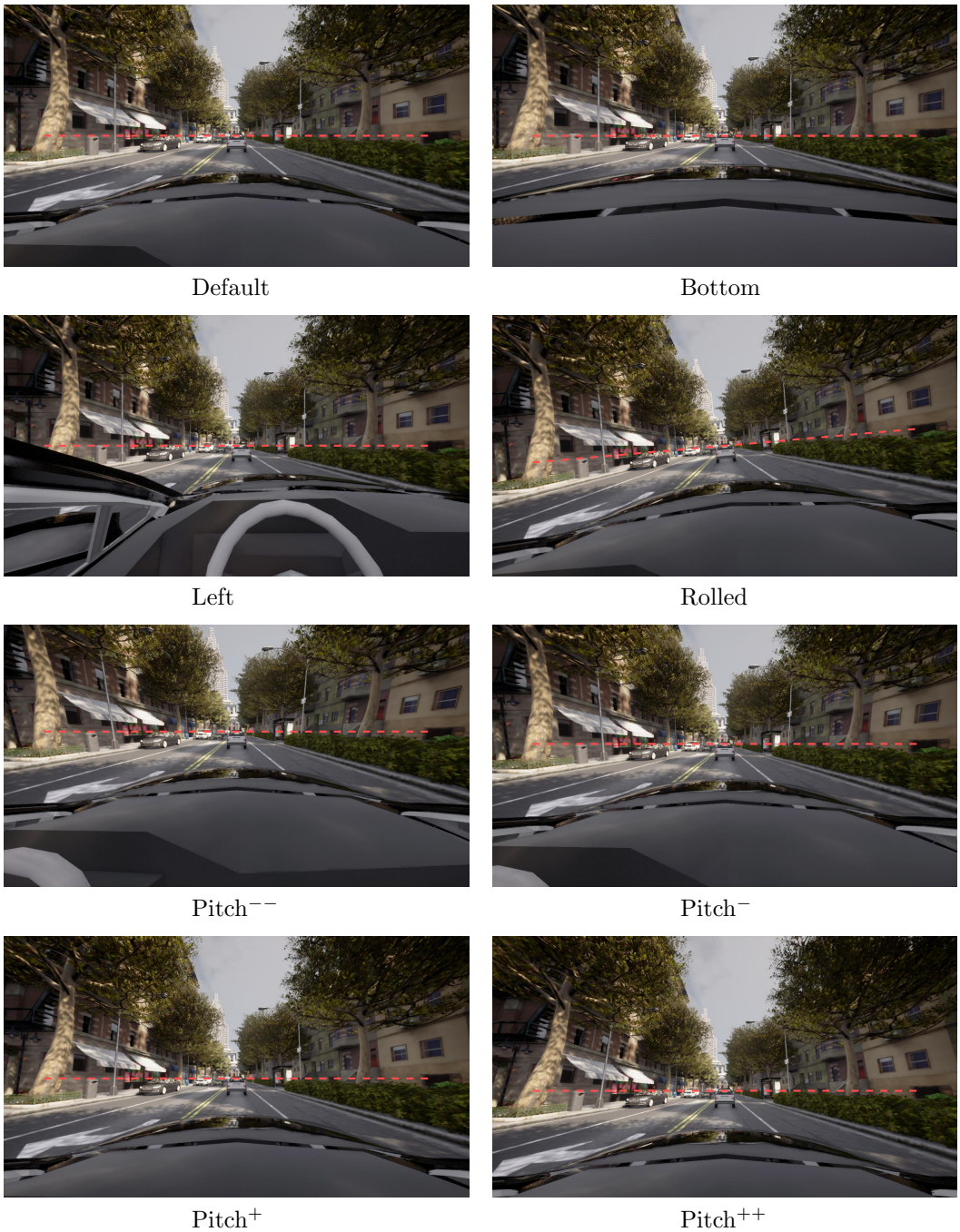


Figure 4.4: Viewpoints included in the dataset for each location. Horizon highlighted by a dashed line.

## 4.3 Experimental Evaluation

For a fair representation of model performance, all experiments are carried out on spatially separated train/test sets, splitting each town in an area used for training and one used for testing, so that images extracted in the same part of the map belong to the same split. The train set is composed of a total of 10,466 images per position while the test set is composed of 1,491 images per position. In the following sections training is always performed on the aforementioned train split of the specified position and evaluation is performed on the test split.

### 4.3.1 BEV Semantic Segmentation

We use Simple-BEV [30] as the BEV semantic segmentation network for our experiments. Simple-BEV is a minimal yet high performance architecture composed of three stages. First, an image encoder processes the RGB image. Then, a parameter-free 3D lifting mechanism fills a 3D feature volume by bilinearly sampling the encoder features in positions obtained by projecting the voxel centers using the camera instrinsics. Finally, those features are collapsed along the vertical direction and fed to a UNet-like network with a semantic segmentation head that generates the final BEV semantic segmentation. We slightly modify the network for our setting by switching to a multiclass segmentation task from the original monoclass paradigm and eliminating the auxiliary task heads.

### 4.3.2 Measuring Performance Drop from Viewpoint Shift

To estimate the effect of the viewpoint shift on a BEV model for semantic segmentation, we train what we call a "baseline model" on the Default position. Then, we measure the quality of the predicted BEV on acquisitions from other viewpoints. The results are presented in Tab. 4.2.

Compared to the performance on the default viewpoint that the model was trained on, the results show a drop in performance for all viewpoints, although with very different levels of degradation. On the Bottom and Top-Left positions, despite the ego-vehicle occluding a bigger part of the frames (see Fig. 4.4), the network demonstrates quite good generalization, resulting in a modest decrease in

Table 4.2: Performance – in IoU (%) – on BEV semantic segmentation of a baseline model trained on the Default position, and tested with different camera viewpoints. Best is in bold, second best underlined. Sidew. stands for sidewalk, Veget. stands for vegetation.

| Test viewpoint | Road | Sidew. | Build. | Wall | Veget. | Terrain | Car | Truck | mIoU | Δ |
|---|---|---|---|---|---|---|---|---|---|---|
| Default | **81.20** | **61.61** | **63.45** | **3.67** | **66.31** | **42.45** | **42.45** | **21.81** | **48.23** | – |
| Bottom | <u>79.27</u> | 55.96 | 61.60 | 2.45 | 65.03 | 38.99 | 30.84 | <u>12.62</u> | <u>43.22</u> | -10.4% |
| Top-Left | 78.81 | <u>56.25</u> | <u>62.25</u> | <u>3.18</u> | <u>65.71</u> | <u>39.79</u> | 27.19 | 8.57 | 42.72 | -11.4% |
| Roll | 58.56 | 36.86 | 39.99 | 1.31 | 52.68 | 26.79 | <u>31.63</u> | 10.90 | 32.34 | -32.9% |
| Pitch$^{--}$ | 34.31 | 25.10 | 32.97 | 0.49 | 44.19 | 22.3 | 1.45 | 0.18 | 20.12 | -58.3% |
| Pitch$^{-}$ | 55.48 | 22.00 | 21.21 | 0.37 | 37.94 | 15.99 | 1.38 | 0.71 | 19.38 | -59.8% |
| Pitch$^{+}$ | 30.85 | 23.43 | 26.50 | 0.35 | 33.56 | 21.91 | 0.09 | 0.00 | 17.09 | -64.6% |
| Pitch$^{++}$ | 41.73 | 14.38 | 10.97 | 0.09 | 16.16 | 10.05 | 0.13 | 0.00 | 11.69 | -75.8% |

mIoU (around 10%). The largest drop occurs for classes with rather small objects (car, truck), where even small localization errors have a huge impact on the IoU.

The Rolled viewpoint has a significant impact on the performance, impacting the reconstruction especially on pixels that are outside the central portion of the BEV (i.e., far from the vanishing point). This is quite evident looking at Fig. 4.5, which compares the prediction of the baseline model on the RGB input from the default viewpoint and the prediction on the rolled RGB input.

Finally, the largest decrease occurs when the viewpoint is pitched. BEV models, such as Simple-BEV, typically construct a 3D feature volume aligned with the camera reference system. If the camera optical axis is not parallel to the ground plane – which is the case when there is pitch – reducing the vertical dimension of this volume results in a plane which is tilted with the respect to the ground plane. This causes the output BEV to be deformed and explains the degradation in the results. In Fig. 4.6 it is easy to appreciate that even slight rotations along the pitch axis cause a drastic degradation of the quality of the prediction.

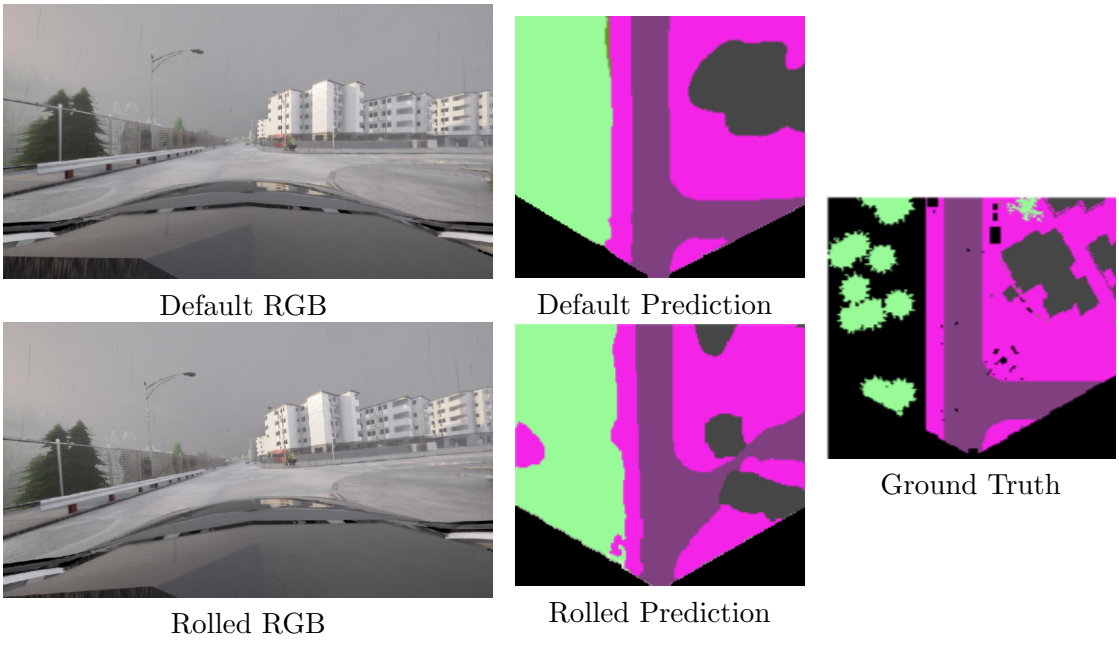


Figure 4.5: Example of BEV reconstructions for a non-default viewpoint. On the left, the RGB image from the default position and the rolled viewpoint, respectively. In the middle, prediction of the baseline model on the two images. On the right, the BEV ground truth.

### 4.3.3 Increasing Viewpoint Robustness on Multiple In-domain Viewpoints

We run a second set of experiments by training the same Simple-BEV architecture on data acquired from a mix of positions. We refer to it as *mixed model*. We hypothesize that augmenting the training dataset with different viewpoints could enhance the model generalization and mitigate the degradation due to viewpoint shifts. Our aim with these experiments is to validate this hypothesis.

We train the model on a dataset composed of a mix of viewpoints (Default, Top-Left, Bottom, Rolled) and evaluate on a test set that includes only positions seen at training time. From the results in Tab. 4.3, we observe that the mixed model can perform almost equally well across all those viewpoints. The only major difference is the class "truck", which drops to almost zero for all viewpoints. However, note that the number of trucks in the dataset is very small.

In Fig. 4.7 we compare the results obtained on all viewpoints of the baseline

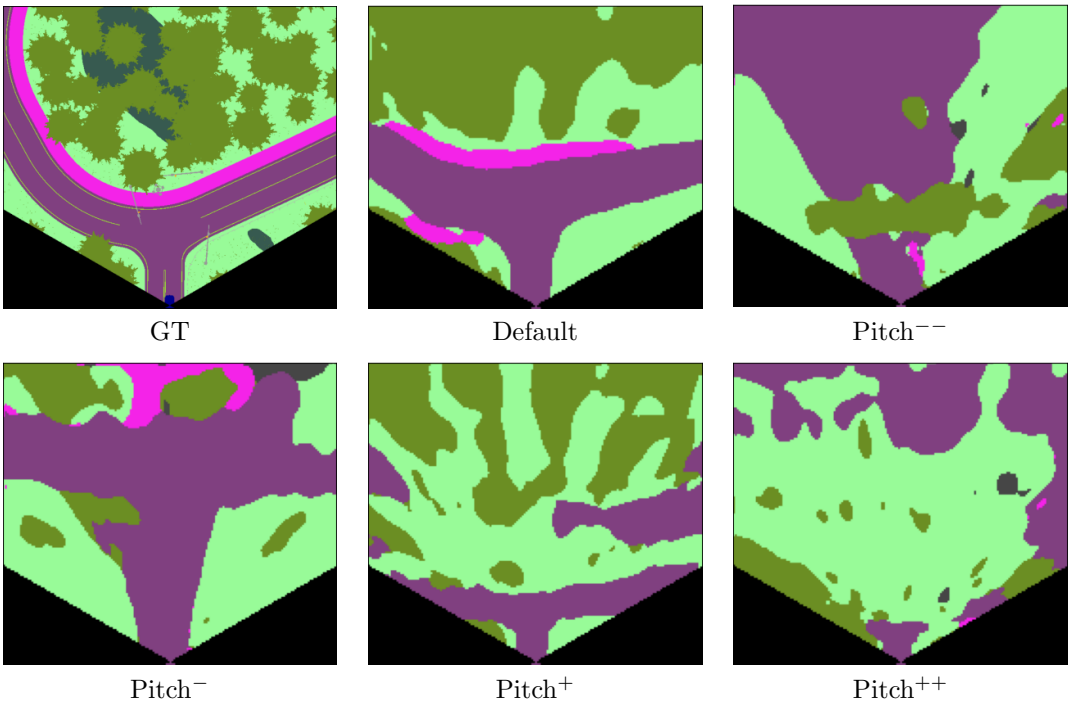


Figure 4.6: Example of BEV maps predicted by the baseline model on viewpoints with different pitch.

model (trained only on the Default viewpoint) and of the mixed model (trained on Default, Bottom, Top-Left, Roll). The mixed model does significantly better on all positions compared to the baseline model, except for the default viewpoint, indicating that targeting a single viewpoint is preferable, if possible. Still, this experiment shows that a BEV reconstruction model can learn to perform its task on multiple viewpoints, if they do not differ too much and they are provided at training time.

### 4.3.4 Increasing Viewpoint Robustness on Unseen Viewpoints

Our last experiment aims to measure wether training on multiple viewpoints can help generalize to viewpoints that were *not seen* at training time. We consider again the mixed model trained on a mix of viewpoints (Default, Top-Left, Bottom, Rolled) and evaluate on a test set of unseen viewpoints (Pitch$^{--}$, Pitch$^{-}$, Pitch$^{+}$,

Table 4.3: Performance – in IoU % – on BEV semantic segmentation of a model trained on a mixture of viewpoints (Default, Bottom, Top-Left, Roll) and tested on those (in-domain) viewpoints. Veget. stands for vegetation.

| Test viewpoint | Road | Sidewalk | Building | Wall | Veget. | Terrain | Car | Truck | mIoU |
|---|---|---|---|---|---|---|---|---|---|
| Default | 82.83 | 60.67 | 66.01 | 1.85 | 69.29 | 45.23 | 33.00 | 1.51 | 45.05 |
| Bottom | 80.96 | 57.01 | 62.21 | 1.20 | 68.68 | 43.38 | 29.46 | 0.29 | 42.90 |
| Top-Left | 82.61 | 60.64 | 65.16 | 2.01 | 69.03 | 44.41 | 32.55 | 1.40 | 44.73 |
| Roll | 82.85 | 60.78 | 66.16 | 1.90 | 69.38 | 45.10 | 31.07 | 1.60 | 44.86 |

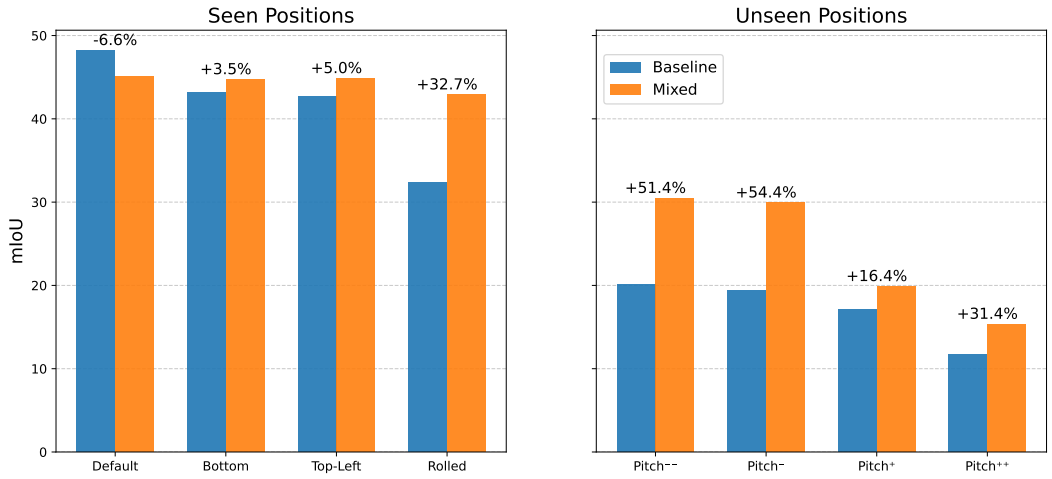


Figure 4.7: Comparison of the results of the baseline model (in blue, trained on Default only) and of the mixed model (in orange, trained on Default, Bottom, Top-Left, Roll). Results on viewpoints seen (left) and unseen (right) during training.

Pitch$^{++}$). The results are reported in Tab. 4.4.

The performance is not competitive with the results obtained by the same model on viewpoints that were seen in training (Tab. 4.3). The very low scores on small objects, especially car, show that localization remains very imprecise. However, in Fig. 4.7 we can observe that the training on multiple viewpoints was able to increase the viewpoint robustness, even for unseen perturbations. More specifically, the results for negative pitch perturbations are rather good, and show that the mixed model generalized significantly better than the Baseline. We hypothesize that this result is due to the similarity between the appearance of objects in the bottom camera pose (seen in training) and this viewpoint, reducing the domain shift between the train and test sets.

Table 4.4: Performance – in IoU (%) – on BEV semantic segmentation of a model trained on a mixture of viewpoints (Default, Bottom, Top-Left, Roll) and tested on unseen viewpoints ($\text{Pitch}^{--}$, $\text{Pitch}^{-}$, $\text{Pitch}^{+}$, $\text{Pitch}^{++}$). Veget. stands for Vegatation.

| Test viewpoint | Road | Sidewalk | Building | Wall | Veget. | Terrain | Car | Truck | mIoU |
|---|---|---|---|---|---|---|---|---|---|
| $\text{Pitch}^{--}$ | 63.80 | 40.25 | 49.26 | 0.48 | 56.62 | 32.79 | 0.57 | 0.00 | 30.47 |
| $\text{Pitch}^{-}$ | 66.02 | 34.90 | 49.40 | 0.66 | 56.62 | 29.40 | 2.18 | 0.33 | 29.94 |
| $\text{Pitch}^{+}$ | 38.11 | 27.95 | 35.64 | 0.19 | 36.64 | 20.58 | 0.02 | 0.00 | 19.89 |
| $\text{Pitch}^{++}$ | 48.62 | 16.79 | 18.92 | 0.06 | 26.97 | 11.41 | 0.14 | 0.00 | 15.36 |

## 4.4 Conclusion

With VS-Sim, we release a synthetic dataset of road scene images that can be used to study the robustness to viewpoint shifts of modern computer vision methods. This stems from the observation on a real-world application of an extreme diversity of viewpoints not present in most public datasets. We also provide the first analysis of viewpoint robustness for bird's eye view multiclass dense semantic segmentation from a monocular camera.

The experimental results show that a BEV model trained on data in a single position is significantly affected even by small variations in viewpoint. Among the camera positions we studied, translations turned out to affect performance the least, while changes of orientation have a larger impact. Even small positive rotations along the pitch axis (i.e., lower horizon) greatly reduce mIoU scores across all classes, and completely destroy the ability to localize smaller objects.

Future research can exploit the dataset for analysis on other tasks such as depth estimation, frontal-view semantic segmentation, and object detection. The dataset could also be used for research on data augmentations that introduce different points of view (e.g., using simulated rotations or novel view synthesis) to improve viewpoint robustness without the necessity to capture data from different physical configurations.

# Chapter 5

# Conclusion

This dissertation focuses on advancing bird's eye view (BEV) perception from monocular cameras, particularly in contexts where annotated training data is scarce. While the BEV offers a compact and geometrically coherent representation advantageous for driving systems, its adoption is slowed by a data bottleneck. Traditional training methods require ground truth derived from expensive 3D sensors like lidar or radar, making data collection and annotation costly, time-consuming, and difficult to scale, especially for capturing rare but crucial events. In many real-world scenarios, such as the dashcam application central to this thesis, these 3D data are unavailable. Furthermore, the diversity of camera mounting positions encountered in large-scale deployments introduces substantial viewpoint shifts, creating robustness challenges that current datasets and methods often overlook. This thesis tackles these fundamental problems by exploring BEV perception under data constraints and investigating the impact of viewpoint variability.

Our main contribution addresses the data challenge. While bird's eye view representations are desirable for our application, the necessary data to perform the typical fully supervised training is unavailable in our setting. This led us to the proposal and study of a new method, RendBEV, to train bird's eye view segmentation networks without bird's eye view labels, which we introduced in Chapter 3. This is the first method to train these networks without direct bird's eye view supervision, be it labels or pseudolabels. RendBEV renders semantic perspective views with values sampled from the bird's eye view network output. In order to do so, it

uses a geometry module to provide estimates of the scene geometry, which makes rendering possible. We have shown two possible instantiations of our method with different geometry modules that lead to different rendering strategies. The first of them is based on a neural density field; it provides volumetric density estimates for 3D points, and is used together with a volumetric rendering formulation. The second instantiation relies on a monocular metric depth estimation model for the scene geometry and is paired with a simpler, more efficient, depth-based rendering formulation. We have shown how RendBEV can be used to train existing models such as SkyEye [25] and SimpleBEV [30] on the KITTI-360 [54] and nuScenes [53] datasets, outperforming baselines by a significant margin in the setting in which no bird's eye view supervision is available. We have also shown how our method can be used as a pretraining method when some amount of ground truth bird's eye view labels are available, leading to a performance boost compared to training from scratch on the same data, and outperforming or being on-par with competing methodologies targeting this setting. Finally, we have extensively studied the robustness and limitations of RendBEV through additional experiments and analyses. The main limitation of RendBEV lies in dealing with dynamic scenes and objects which we plan to address in future work through explicit motion and object modeling.

An additional contribution of this thesis addresses the problem of viewpoint robustness. We have studied the viewpoint diversity problem by generating a synthetic dataset mimicking common variations observed in our real data and evaluating the robustness of bird's eye view semantic segmentation networks to these variations in Chapter 4. Simulation enables us to generate data with ground truth labels without deploying real lidar and radar sensors and having to perform manual annotation. We would not be able to access these data otherwise. Through our experiments, we have shown that bird's eye view semantic segmentation networks trained in a single position exhibit declining performance when the position varies at test time. We have also demonstrated that a simple strategy in which more viewpoints are added to the training set improves the viewpoint generalization capability, highlighting the importance of using viewpoint-diverse training sets, which are not common in the bird's eye view perception literature.

In conclusion, we have advanced the study of bird's eye view segmentation per-

ception in settings closer to our real application, in which conditions are less ideal than in the usual benchmarks in the literature. These contributions demonstrate the feasibility of BEV perception even with low-cost sensors and limited labels, paving the way for wider adoption. We hope our work provides a strong foundation on top of which stronger models and more complete pipelines for bird's eye view perception with limited labels and more in-the-wild settings can be built.

## The Future of Vision-Centric BEV Perception

Despite significant progress, many challenges remain open in vision-centric bird's eye view perception. Although it continues to be a popular research topic, the current literature is dominated by fully supervised training frameworks, relying on three-dimensional ground truth data. While this might be a valid framework to develop methods for certain real-world applications with standarized sensor specifications, such as fully automated driving platforms with callibrated rigs, it is not applicable for others such as dashcams.

Even within controlled environments, the reliance on full supervision hinders **scalability**: different vehicle types or models with different camera rigs often mean that datasets need to be captured and annotated for each specific setting, which is prohibitively expensive. Promising research directions to address this challenge are label-efficient and generalizable models. In the adjacent field of 3D occupancy prediction several recent works propose label-efficient ways to train networks, some via rendering semantic views [72, 101, 102] in a similar manner to RendBEV, or exploiting vision-language alignment in models like CLIP enable open vocabulary segmentation [103].

However, these methods still mostly use the standard benchmarks for training and evaluation; generalization across datasets and to more uncontrolled settings remains underexplored. A notable exception in bird's eye view mapping (only static/layout classes) which addresses the generalization and robustness concerns is Map It Anywhere [104]. They introduce a data engine which exploits crowd-sourced images paired with OpenStreetMap to obtain diverse image-BEV pairs at scale automatically and propose a camera-agnostic baseline segmentation network. We believe bridging these perspectives – label efficiency and generalizable

approaches – will be key to unlock real-world deployment at scale, and thus maximize the impact of BEV perception.

# Bibliography

[1] World Health Organization, "Factsheet: Road Traffic Injuries," 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[2] L. J. Blincoe, T. R. Miller, J.-S. Wang, D. Swedler, T. Coughlin, B. Lawrence, F. Guo, S. G. Klauer, and T. A. Dingus, "The Economic and Societal Impact of Motor Vehicle Crashes, 2019 (Revised)," no. DOT HS 813 403, Feb. 2023. [Online]. Available: https://rosap.ntl.bts.gov/view//78698

[3] European Road Safety Observatory. Brussels, European Commission, Directorate General for Transport, "Annual statistical report on road safety in the EU, 2025," 2025. [Online]. Available: https://road-safety.transport.ec.europa.eu/document/download/17d70e9c-d9c4-4273-b497-41b61194e808_en?filename=ERSOnext_AnnualReport_20250227.pdf

[4] Centers for Disease Control and Prevention, National Center for Health Statistics. National Vital Statistics System, "Mortality 2018-2023 on CDC WONDER Online Database, released in 2024. Data are from the Multiple Cause of Death Files, 2018-2023, as compiled from data provided by the 57 vital statistics jurisdictions through the Vital Statistics Cooperative Program." [Online]. Available: https://wonder.cdc.gov/controller/datarequest/D158;jsessionid=343547D330EE00359E71BC9B177F

[5] J. B. Cicchino, "Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates," *Accident Analysis & Prevention*, vol. 99, pp. 142–152, 2017.

[6] K. D. Kusano, J. M. Scanlon, Y.-H. Chen, T. L. McMurry, T. Gode, and T. Victor, "Comparison of Waymo Rider-Only crash rates by crash type to human benchmarks at 56.7 million miles," *Traffic Injury Prevention*, 2025, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15389588.2025.2499887. [Online]. Available: https://doi.org/10.1080/15389588.2025.2499887

[7] "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE, Standard J3016_202104, 2021. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/

[8] ISO/SAE International, "ISO/SAE PAS 22736:2021 - Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," https://www.iso.org/standard/73766.html, 2021, Publisher: ISO and SAE International.

[9] E. Shi, T. M. Gasser, A. Seeck, and R. Auerswald, "The Principles of Operation Framework: A Comprehensive Classification Concept for Automated Driving Functions," *SAE International Journal of Connected and Automated Vehicles*, vol. 3, no. 12-03-01-0003, pp. 27–37, 2020.

[10] Y. Ma, T. Wang, X. Bai, H. Yang, Y. Hou, Y. Wang, Y. Qiao, R. Yang, and X. Zhu, "Vision-Centric BEV Perception: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[11] T. Roddick, "Learning Birds-Eye View Representations for Autonomous Driving," Ph.D. dissertation, Apollo - University of Cambridge Repository, 2021. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/330203

[12] H. Piñeiro Monteagudo, L. Taccari, A. Pjetri, F. Sambo, and S. Salti, "RendBEV: Semantic Novel View Synthesis for Self-Supervised Bird's Eye View Segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

[13] H. Piñeiro Monteagudo, L. Taccari, A. Pjetri, F. Sambo, and S. Salti, "Rend-BEV: Semantic perspective view rendering as supervision for bird's eye view segmentation," *IEEE Access*, vol. 14, pp. 12 255–12 272, 2026.

[14] R. Turra, M. Simoncini, H. Piñeiro Monteagudo, A. Pjetri, S. Salti, and L. Taccari, "VS-Sim: A Synthetic Dataset for Viewpoint Shift Robustness," in *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*. Springer Nature Switzerland, 2025, pp. 507–519.

[15] C.-C. Lin and M.-S. Wang, "A Vision Based Top-View Transformation Model for a Vehicle Parking Assistant," *Sensors*, vol. 12, no. 4, pp. 4431–4446, 2012.

[16] E. Dagan, O. Mano, G. P. Stein, and A. Shashua, "Forward collision warning with a single camera," in *IEEE Intelligent Vehicles Symposium, 2004*, 2004, pp. 37–42.

[17] A. Torralba, P. Isola, and W. Freeman, *Foundations of Computer Vision*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2024. [Online]. Available: https://mitpress.mit.edu/9780262048972/foundations-of-computer-vision/

[18] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[20] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological Cybernetics*, vol. 64, no. 3, pp. 177–185, Jan. 1991. [Online]. Available: https://doi.org/10.1007/BF00201978

[21] L. Reiher, B. Lampe, and L. Eckstein, "A Sim2Real Deep Learning Approach for the Transformation of Images from Multiple Vehicle-Mounted Cameras

to a Semantically Segmented Image in Bird's Eye View," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[22] T. Roddick and R. 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, pp. 11 135–11 144.

[23] Q. Li, Y. Wang, Y. Wang, and H. Zhao, "HDMapNet: An Online HD Map Construction and Evaluation Framework," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4628–4634.

[24] N. Gosala and A. Valada, "Bird's-Eye-View Panoptic Segmentation Using Monocular Frontal View Images," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1968–1975, Apr. 2022.

[25] N. Gosala, K. Petek, P. L. J. Drews-Jr, W. Burgard, and A. Valada, "Sky-Eye: Self-Supervised Bird's-Eye-View Semantic Mapping Using Monocular Frontal View Images," in *Proceeding of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2023, pp. 14 901–14 910.

[26] S. A. Abbas and A. Zisserman, "A Geometric Approach to Obtain a Bird's Eye View from an Image," in *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 4095–4104.

[27] Y. Kim and D. Kum, "Deep Learning based Vehicle Position and Orientation Estimation via Inverse Perspective Mapping Image," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 317–323.

[28] Y. B. Can, A. Liniger, O. Unal, D. Paudel, and L. Van Gool, "Understanding Bird's-Eye View of Road Semantics using an Onboard Camera," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3302–3309, 2022.

[29] T. Roddick, A. Kendall, and R. Cipolla, "Orthographic Feature Transform for Monocular 3D Object Detection," *British Machine Vision Conference (BMVC)*, 2019.

[30] A. W. Harley, Z. Fang, J. Li, R. Ambrus, and K. Fragkiadaki, "Simple-BEV: What Really Matters for Multi-Sensor BEV Perception?" in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 2759–2765.

[31] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 1–18.

[32] A. Palazzi, G. Borghi, D. Abati, S. Calderara, and R. Cucchiara, "Learning to Map Vehicles into Bird's Eye View," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 233–243.

[33] C. Lu, M. J. G. van de Molengraft, and G. Dubbelman, "Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 445–452, 2019.

[34] K. Mani, S. Daga, S. Garg, S. S. Narasimhan, M. Krishna, and K. M. Jatavallabhula, "MonoLayout: Amodal scene layout from a single image," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 1689–1697.

[35] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4867–4873, 2020.

[36] W. Yang, Q. Li, W. Liu, Y. Yu, Y. Ma, S. He, and J. Pan, "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. 15 536–15 545.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[38] A. Saha, O. Mendez, C. Russell, and R. Bowden, "Translating Images into Maps," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 9200–9206.

[39] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries ," in *Conference on Robot Learning*. PMLR, 2022, pp. 180–191.

[40] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 213–229.

[41] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," in *International Conference on Learning Representations (ICLR)*, 2021.

[42] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 194–210.

[43] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger, "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," in *CVPR*, 2019.

[44] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander, "Categorical Depth Distribution Network for Monocular 3D Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 8551–8560.

[45] Y. Li, Z. Ge, G. Yu, J. Yang, Z. Wang, Y. Shi, J. Sun, and Z. Li, "BEVDepth: Acquisition of Reliable Depth for Multi-view 3D Object Detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 1477–1485.

[46] C. Yang, Y. Chen, H. Tian, C. Tao, X. Zhu, Z. Zhang, G. Huang, H. Li, Y. Qiao, L. Lu *et al.*, "BEVFormer v2: Adapting Modern Image Backbones to Bird's-Eye-View Recognition via Perspective Supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 830–17 839.

[47] N. Hendy, C. Sloan, F. Tian, P. Duan, N. Charchut, Y. Xie, C. Wang, and J. Philbin, "FISHING Net: Future Inference of Semantic Heatmaps In Grids," Jun. 2020, arXiv:2006.09917 [cs].

[48] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall, "FIERY: Future Instance Prediction in Bird's-Eye View from Surround Monocular Cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 273–15 282.

[49] A. K. Akan and F. Güney, "StretchBEV: Stretching Future Instance Prediction Spatially and Temporally," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[50] Y. Zhang, Z. Zhu, W. Zheng, J. Huang, G. Huang, J. Zhou, and J. Lu, "BEVerse: Unified Perception and Prediction in Birds-Eye-View for Vision-Centric Autonomous Driving," *arXiv preprint arXiv:2205.09743*, 2022.

[51] P. Li, S. Ding, X. Chen, N. Hanselmann, M. Cordts, and J. Gall, "PowerBEV: A Powerful Yet Lightweight Framework for Instance Prediction in Bird's-Eye View," Jun. 2023, arXiv:2306.10761 [cs].

[52] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[53] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[54] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3292–3310, Mar. 2023.

[55] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2446–2454.

[56] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[57] T. Klinghoffer, J. Philion, W. Chen, O. Litany, Z. Gojcic, J. Joo, R. Raskar, S. Fidler, and J. M. Alvarez, "Towards Viewpoint Robustness in Bird's Eye View Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 8515–8524.

[58] T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari, and F. Yu, "SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 21 371–21 382.

[59] A. Kloukiniotis, A. Papandreou, C. Anagnostopoulos, A. Lalos, P. Kapsalas, D.-V. Nguyen, and K. Moustakas, "CarlaScenes: A synthetic dataset for odometry in autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 4520–4528.

[60] Z. Song, Z. He, X. Li, Q. Ma, R. Ming, Z. Mao, H. Pei, L. Peng, J. Hu, D. Yao *et al.*, "Synthetic Datasets for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 1847–1864, 2023.

[61] N. Gosala, K. Petek, B. Ravi Kiran, S. Yogamani, P. Drews-Jr, W. Burgard, and A. Valada, "Letsmap: Unsupervised representation learning for label-efficient semantic bev mapping," in *Proceedings of the European Conference on Computer Vision (ECCV)*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Springer, 2024, pp. 110–126.

[62] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "DINOv2: Learning Robust Visual Features without Supervision," 2023.

[63] J. Zhu, L. Liu, Y. Tang, F. Wen, W. Li, and Y. Liu, "Semi-Supervised Learning for Visual Bird's Eye View Semantic Segmentation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 9079–9085.

[64] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3827–3837.

[65] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[66] W. Yin, C. Zhang, H. Chen, Z. Cai, G. Yu, K. Wang, X. Chen, and C. Shen, "Metric3D: Towards Zero-shot Metric 3D Prediction from A Single Image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 9043–9053.

[67] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, and S. Shen, "Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation,"

*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10 579–10 596, 2024.

[68] A. Bochkovskii, A. Delaunoy, H. Germain, M. Santos, Y. Zhou, S. R. Richter, and V. Koltun, "Depth Pro: Sharp Monocular Metric Depth in Less Than a Second," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://arxiv.org/abs/2410.02073

[69] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, p. 99–106, Dec. 2021. [Online]. Available: https://doi.org/10.1145/3503250

[70] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural Radiance Fields from One or Few Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[71] F. Wimbauer, N. Yang, C. Rupprecht, and D. Cremers, "Behind the Scenes: Density Fields for Single View Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 9076–9086. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.00876

[72] A. Hayler, F. Wimbauer, D. Muhle, C. Rupprecht, and D. Cremers, "S4C: Self-Supervised Semantic Scene Completion With Neural Fields ," in *International Conference on 3D Vision (3DV)*, 2024, pp. 409–420. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/3DV62453.2024.00133

[73] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[74] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic

Urban Scene Understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[75] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "SAM 2: Segment Anything in Images and Videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714

[76] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," *arXiv preprint arXiv:2303.05499*, 2023.

[77] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, "Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks," 2024.

[78] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning." in *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

[79] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1195–1204.

[80] X. Chen, Y. Yuan, G. Zeng, and J. Wang, "Semi-Supervised Semantic Segmentation with Cross Pseudo Supervision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[81] L. Yang, L. Qi, L. Feng, W. Zhang, and Y. Shi, "Revisiting Weak-to-Strong Consistency in Semi-Supervised Semantic Segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7236–7246.

[82] N. Drenkow, N. Sani, I. Shpitser, and M. Unberath, "A Systematic Review of Robustness in Deep Learning for Computer Vision: Mind the gap?" 2022.

[83] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming," 2020.

[84] L. Taccari, F. Sambo, L. Bravi, S. Salti, L. Sarti, M. Simoncini, and A. Lori, "Classification of crash and near-crash events from dashcam videos and telematics," in *International Conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2460–2465.

[85] V. Vidit, M. Engilberge, and M. Salzmann, "Learning Transformations To Reduce the Geometric Shift in Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17 441–17 450.

[86] L. Kong, S. Xie, H. Hu, L. X. Ng, B. R. Cottereau, and W. T. Ooi, "RoboDepth: Robust Out-of-Distribution Depth Estimation under Corruptions," in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

[87] D. Shenaj, E. Fanì, M. Toldo, D. Caldarola, A. Tavera, U. Michieli, M. Ciccone, P. Zanuttigh, and B. Caputo, "Learning Across Domains and Devices: Style-Driven Source-Free Domain Adaptation in Clustered Federated Learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 444–454.

[88] Y. Man, L.-Y. Gui, and Y.-X. Wang, "DualCross: Cross-Modality Cross-Domain Adaptation for Monocular BEV Perception," in *IROS*, 2023.

[89] S. Wang, X. Zhao, H. Xu, Z. Chen, D. Yu, J. Chang, Z. Yang, and F. Zhao, "Towards Domain Generalization for Multi-view 3D Object Detection in Bird-Eye-View," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 13 333–13 342.

[90] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 4990–4999.

[91] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2636–2645.

[92] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual Worlds as Proxy for Multi-Object Tracking Analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[93] E. Alberti, A. Tavera, C. Masone, and B. Caputo, "IDDA: A Large-Scale Multi-Domain Dataset for Autonomous Driving," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5526–5533, 2020.

[94] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.

[95] P. Testolina, F. Barbato, U. Michieli, M. Giordani, P. Zanuttigh, and M. Zorzi, "SELMA: SEmantic Large-Scale Multimodal Acquisitions in Variable Weather, Daytime and Viewpoints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7012–7024, 2023.

[96] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[97] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "MOT20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[98] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler, "Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[99] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[100] V. Guizilini, I. Vasiljevic, D. Chen, R. Ambruș, and A. Gaidon, "Towards Zero-Shot Scale-Aware Monocular Depth Estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 9233–9243.

[101] Y. Huang, W. Zheng, B. Zhang, J. Zhou, and J. Lu, "SelfOcc: Self-Supervised Vision-Based 3D Occupancy Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 946–19 956.

[102] S. Boeder, F. Gigengack, and B. Risse, "GaussianFlowOcc: Sparse and Weakly Supervised Occupancy Estimation using Gaussian Splatting and Temporal Flow," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2025, pp. 24 943–24 954.

[103] ——, "LangOcc: Open Vocabulary Occupancy Estimation via Volume Rendering," in *International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 200–210.

[104] C. Ho, J. Zou, O. Alama, S. M. J. Kumar, B. Chiang, T. Gupta, C. Wang, N. Keetha, K. Sycara, and S. Scherer, "Map It Anywhere (MIA): Empowering Bird's Eye View Mapping using Large-scale Public Data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.