Towards Improving the Robustness and Generalizability of Camera-Based Bird's Eye View Segmentation Models

by

Merve Rabia Barin

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

July 17, 2025

Towards Improving the Robustness and Generalizability of Camera-Based Bird's Eye View Segmentation Models

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Merve Rabia Barin

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Asst. Prof. Fatma Güney (Advisor)

_____

Prof. Hazım Kemal Ekenel

_____

Asst. Prof. Pınar Yanardağ

Date: _____

To my family — for whom I would give everything, and without whose support this would mean nothing.

ABSTRACT

Towards Improving the Robustness and Generalizability of Camera-Based Bird's Eye View Segmentation Models
Merve Rabia Barin
Master of Science in Computer Science and Engineering
July 17, 2025

Bird's Eye View (BEV) representations are essential for many autonomous driving tasks, as they provide a structured understanding of the 3D spatial layout of the environment. Extracting BEV from camera images offers a cost-effective and scalable alternative to LiDAR. However, vision-only BEV models come with their own challenges: they often lack robustness when encountered with corrupted inputs (e.g., noisy images or camera failures) and struggle to generalize due to insufficient coverage of existing datasets. Both robustness and generalization are important for making autonomous driving systems safe and reliable for their deployment in real-world. In this thesis, we address both of these challenges. First, to improve the robustness of BEV perception, we propose utilizing the strong and generalizable features of large vision foundation models by integrating them into BEV perception using a parameter-efficient adaptation technique. Our experiments show improved robustness under various input corruptions, with additional gains observed by scaling the model size and input resolution. We also demonstrate faster convergence and reduced parameter requirements, highlighting the efficiency of our approach. Second, to tackle the issue of limited dataset coverage and the lack of flexible input control in multiview driving scene generation, we propose a guided layout generation pipeline that targets moderately challenging and underrepresented scene configurations. By analyzing the distribution of real-world driving scenes, we guide a generative model to synthesize scenes in these specific regions of the distribution. While augmenting the training set of BEV perception models with these guided samples does not yet yield performance gains, the generated layouts closely follow real-world configurations and mostly fall within the target region. This suggests promising potential for targeted synthetic data augmentation as a future direction for improving the generalization of BEV perception models.

ÖZETÇE

Kuş Bakışı Görünüm (Bird's Eye View) Segmentasyon Modellerinin Sağlamlığını ve Genelleme Yeteneğini Geliştirme
Merve Rabia Barin
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans
17 Temmuz 2025

Kuş Bakışı Görünüm, çevrenin üç boyutlu mekânsal düzenini yapılandırılmış bir şekilde sunması nedeniyle, otonom sürüşteki pek çok görev için kritik öneme sahiptir. Araç kameralarından kuş bakışı görünüm çıkarmak, LiDAR tabanlı yöntemlere kıyasla daha uygun maliyetli ve ölçeklenebilir bir alternatif sunar. Ancak yalnızca görsel verilere dayanan kuş bakışı görünüm üreten modeller bazı zorluklarla karşılaşmaktadır: Bu modeller, bozulmuş girdilerle (örneğin bulanık kamera görüntüleri veya kamera arızaları) karşılaştıklarında yeterince dayanıklı değildir ve mevcut veri kümeleri yeterli çeşitliliğe sahip olmadığından genelleme konusunda da yetersiz kalmaktadır. Dayanıklılık ve genelleme, otonom sürüş sistemlerinin gerçek dünya koşullarında güvenli ve güvenilir bir şekilde çalışabilmesi için kritik faktörlerdir. Bu tezde, her iki soruna da çözüm sunmayı hedefliyoruz. İlk olarak, kuş bakışı görünüm üreten modellerin dayanıklılığını artırmak amacıyla, büyük görsel temel modellerin (vision foundation models) güçlü ve genellenebilir özelliklerinden faydalanarak bu modelleri parametre açısından verimli bir adaptasyon tekniği ile BEV algılama sistemlerine entegre ediyoruz. Yaptığımız deneyler, çeşitli girdi bozulmaları altında modelin dayanıklılığında artış sağlandığını ve model boyutu ile girdi çözünürlüğü arttıkça bu kazanımların daha da belirginleştiğini göstermektedir. Ayrıca yöntemimizin daha hızlı yakınsama sağladığını ve daha az öğrenilebilir parametreye ihtiyaç duyduğunu da ortaya koymaktayız. İkinci olarak, sınırlı veri kümesi kapsamı ve çoklu görüşlü sürüş sahnesi üretiminde esnek girdi kontrolünün olmaması sorununu ele almak amacıyla, orta düzeyde zorluk içeren ve veri kümesinde az temsil edilen sahne yapılarını hedefleyen yönlendirmeli bir yerleşim üretim hattı öneriyoruz. Gerçek sürüş sahnelerinin dağılımını analiz ederek, bir üretici modeli bu dağılımın belirli bölgelerinde sahneler üretmeye yönlendiriyoruz. Bu yönlendirmeli örneklerle kuş bakışı çıkaran modellerin eğitim kümesini genişletmek şu an için doğrudan bir

performans artışı sağlamasa da, üretilen yerleşimlerin büyük ölçüde gerçek dünya yapılarına uygun olduğu ve hedeflenen dağılım bölgesine denk düştüğü gözlemlenmiştir. Bu durum, önerdiğimiz yöntemin, kuş bakışı algı modellerinin genelleme yeteneğini artırmaya yönelik hedef odaklı sentetik veri üretimi açısından gelecek vadettiğini göstermektedir.

ACKNOWLEDGMENTS

I gratefully acknowledge the support of the Scientific and Technological Research Council of Turkey (TÜBİTAK) and its Department of Science Fellowships and Grant Programs (BİDEB) through the 2210-A Domestic Graduate Scholarship Program for MSc Students, which has supported my master's thesis and all related publications. I am also grateful for the additional support provided by KUIS AI and the European Union (ERC, ENSURE, 101116486).

I would like to extend my sincere appreciation to my advisor Fatma Güney, and to my thesis committee members Prof. Hazım Kemal Ekenel and Asst. Prof. Pınar Yanardağ, for kindly agreeing to become a member on my defense committee.

I am grateful to my loved ones for supporting me no matter what. Thank you to my mom and dad for always standing by me and encouraging me through every high and low.

To my one and only Furkan—thank you for being the most perfect person for me and making me feel on top of the world with your kind words and endless support every single day.

A heartfelt thanks to my girls, Rana and Büşra, my wonderful homemates, for being there for me through every academic, emotional, and life-related challenge and for the and for the countless unforgettable girls' nights filled with way too much sugar.

To all my lab friends, past and present...Particularly: Nazir, for your wisdom and kind heart that left a lasting impact; Volkan, for your endless humor and legendary Kilyos Yalısı invitations; Shadi and Ekin, for all your help and the laughter we shared; Görkay, for the many intellectual conversations— And to everyone else, thank you for sharing this academic journey and for making the lab a socially enjoyable space, even though I haven't always been the most social one.

Lastly, a special thanks to my car, for accompanying me to Sarıyer every day and for witnessing all my laughter, craziness, and loud singalongs, especially the guilty pleasure playlists I'd never dare admit to anywhere else.

A new chapter is beginning for me, and I'm deeply grateful for all the people and memories that have been with me on this road so far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BEV	Bird's Eye View
MAE	Mean Absolute Error
IoU	Intersection-over-Union
mIoU	Mean Intersection-over-Union
LDMs	Latent Diffusion Models

Chapter 1

INTRODUCTION

To navigate safely and make reliable decisions, autonomous vehicles must understand their 3D surroundings. This depends on effective perception systems that convert raw sensor data into structured 3D representations. A widely used representation is the bird's-eye view (BEV), which projects the 3D scene onto a top-down plane. BEV simplifies spatial reasoning by highlighting relevant elements like vehicles and road layout, while filtering out distractions such as the sky or distant background.

This structured representation serves as a powerful intermediate format for a variety of downstream tasks, such as *motion prediction* [Chang et al., 2019, Caesar et al., 2020], *online HD map learning* [Li et al., 2022a, Liu et al., 2023a, Liao et al., 2023, Yuan et al., 2024], and *3D object detection* [Roddick et al., 2018, Liu et al., 2023b, Li et al., 2023]. It is also becoming a popular choice in end-to-end driving systems, where BEV features are used to jointly optimize both perception and planning [Chen et al., 2019, Zhang et al., 2021, Hanselmann et al., 2022, Chen and Krähenbühl, 2022, Hu et al., 2023b].

Although LiDAR provides accurate 3D information that significantly boosts the performance of models extracting BEV representations [Liu et al., 2023b, Harley et al., 2023], it comes with increased computational costs and scalability challenges. As a result, many recent works have focused on building BEV representations using only multi-camera input [Philion and Fidler, 2020, Zhou and Krähenbühl, 2022, Li et al., 2022b, Harley et al., 2023, Chambon et al., 2024], offering a more cost-effective and scalable solution.

The performance of BEV perception models that rely solely on vision input is

(a) Driving scene

(b) Bird's eye view map

Figure 1.1: **An example of camera images from a driving scene in nuScenes paired with its Bird's Eye View (BEV) representation.** The ego vehicle is denoted by a red circle at the center of the BEV, with its front facing upward. Other vehicles are represented in blue.

significantly affected by the quality and diversity of the available camera images, as these models use visual data as their only source of environmental information. This reliance makes them particularly vulnerable to two key issues: *input corruption* after deployment and *limited data coverage* during training.

When camera images are affected by changing conditions like lighting shifts, bad weather, or sensor failures, the model's performance can drop noticeably, creating risks for real-world use. Robustness to unpredictable real-world inputs is thus a critical requirement. Another major challenge is the limited diversity in existing real-world datasets, which often varies depending on where and how the data is collected. For instance, if the data is collected in a low-traffic city, models may struggle to generalize to crowded urban scenarios. This highlights the need for targeted data generation approaches that can systematically produce underrepresented yet realistic driving scenes to improve model generalization.

Motivated by the limitations listed above, this thesis explores the robustness of camera-based BEV models under corrupted inputs and investigates targeted data generation for underrepresented driving scenes, with the goal of developing more reliable and adaptable perception systems.

1.1 Motivation

We begin by highlighting the limitations in the robustness of camera-based BEV perception models and how significant this issue is for dowstream tasks, then emphasize the potential of foundational visual representations to address these shortcomings. We then turn our attention to the insufficient coverage of real-world driving datasets and examine how current multi-view image generators, while aiming to expand data diversity, remain limited to existing scene configurations, making it difficult to systematically target specific challenging conditions. To address this, we introduce driving scene generators, which operate in BEV layout space and offer greater flexibility in designing scene content. We argue that combining layout-level scene generation with multi-view image synthesis presents a promising direction for achieving targeted dataset augmentation and improving the generalization of BEV perception models.

Insufficient Robustness: While the robustness of camera-based BEV perception models under various conditions is a critical factor for ensuring safety, recent work [Xie et al., 2023] shows that camera-based BEV perception models suffer from performance degradation when exposed to different types of corruption such as brightness changes, adversarial weather conditions, motion blur, quantization, frame loss, and camera crash, highlighting a significant challenge. The accuracy of BEV perception plays a crucial role in both motion prediction [Xu et al., 2024] and end-to-end driving. While privileged agents that have access to ground truth BEV demonstrate an impressive driving performance, their student counterparts suffer from mistakes in predicted BEV maps [Zhang et al., 2021]. Similarly, the performance of motion prediction methods drops notably while switching from ground truth BEV to the predicted BEV [Xu et al., 2024].

The availability of large-scale datasets has facilitated the emergence of visual foundation models, renowned for their generalization capabilities. Notably, DINOv2 [Oquab et al., 2023] stands out for its robust, general-purpose visual features, making it a suitable choice for various tasks such as zero-shot correspondence estimation [Zhang et al., 2023], robotics [Blumenkamp et al., 2024], object-centric

learning [Aydemir et al., 2023], point tracking [Aydemir et al., 2024], and object segmentation [Nguyen et al., 2023]. Despite the rich representation capacity of DINOv2, only a few works [Schramm et al., 2024, Sirko-Galouchenko et al., 2024] have explored its potential for BEV segmentation.

In the first part of this thesis, motivated by the potential of foundation models and the need to improve robustness in BEV perception, we explore the adaptation of DINOv2 by evaluating its robustness under various input corruptions, as well as its impact on model performance, parameter efficiency, and convergence behavior [Barın et al., 2024].

Insufficient Coverage of Datasets: Real-world autonomous driving datasets [Caesar et al., 2020, Sun et al., 2020] are expensive to collect and typically cover a limited range of driving scenarios, largely shaped by the specifics of their data collection process. Synthesizing data using controllable multi-view image generative models [Gao et al., 2024a, Wen et al., 2024] has been shown to produce realistic results and can even improve the performance of camera-based BEV perception models when used for data augmentation. These models generate realistic multi-view camera images conditioned on various scene configurations, such as annotated 3D object bounding boxes, textual scene descriptions, and camera parameters.

Although multi-view image generators are powerful tools for synthetic data generation and have been used to address dataset coverage issues, their controllability remains limited in practice. These models typically offer either scene-level or object-level control. Scene-level control involves modifying global attributes such as weather or day/night time while keeping the underlying scene configuration fixed, constrained by the distributions of the original dataset. Object-level control allows inserting or removing objects or altering their appearances [Huang et al., 2024]. However, both types of control primarily focus on visual appearance rather than structural scene variation. This limits their effectiveness for targeted data generation, where the goal is to synthesize scenes that systematically address underrepresented or challenging conditions relevant to BEV perception.

On the other hand, there are generative driving scenario generation works [Sun

et al., 2024, Chitta et al., 2025, Rowe et al., 2025] focusing on synthesizing all driving scene elements (lanes, vehicles, pedestrians, etc.) in bird's eye view from scratch. Since they do not rely on real-world driving logs, the scenarios that can be generated are unlimited. However, as long as scenarios generated by these methods do not have camera image counterparts, they cannot be used as data for BEV perception models.

To bridge this gap, in the second part of this thesis, we propose a guided layout generation pipeline that combines the structural flexibility of scenario generation with the realism of multi-view image synthesis. Our approach operates in the BEV space, where we guide a generative model to produce scene configurations that target specific underrepresented conditions—such as moderately crowded traffic scenarios. These layouts are then passed to a layout-to-multi-view image generator to synthesize realistic camera images, enabling the creation of targeted synthetic data suitable for training and evaluating BEV perception models.

1.2 Contributions

In this thesis, we address two key challenges in bird's-eye view (BEV) perception: (i) improving robustness for visual and environmental corruptions, and (ii) enhancing control over generated driving scenarios to address the insufficient coverage of existing datasets.

1.2.1 Improving Robustness

We integrate DINOv2 into a state-of-the-art BEV segmentation model, SimpleBEV [Harley et al., 2023], by utilizing an efficient adaptation technique [Hu et al., 2022]. Specifically, we replace the backbone of SimpleBEV with DINOv2 for feature extraction and then efficiently update it using Low Rank Adaptation (LoRA). We systematically analyze the effectiveness of our approach by comparing the robustness of our adaptation to the original SimpleBEV with ResNet-101 backbone in terms of accuracy, parameter efficiency, and convergence behavior. Our experiments show that our adaptation improves the robustness of BEV perception under adversarial conditions, achieves higher accuracy, and does so with significantly fewer learnable

parameters and shorter training times.

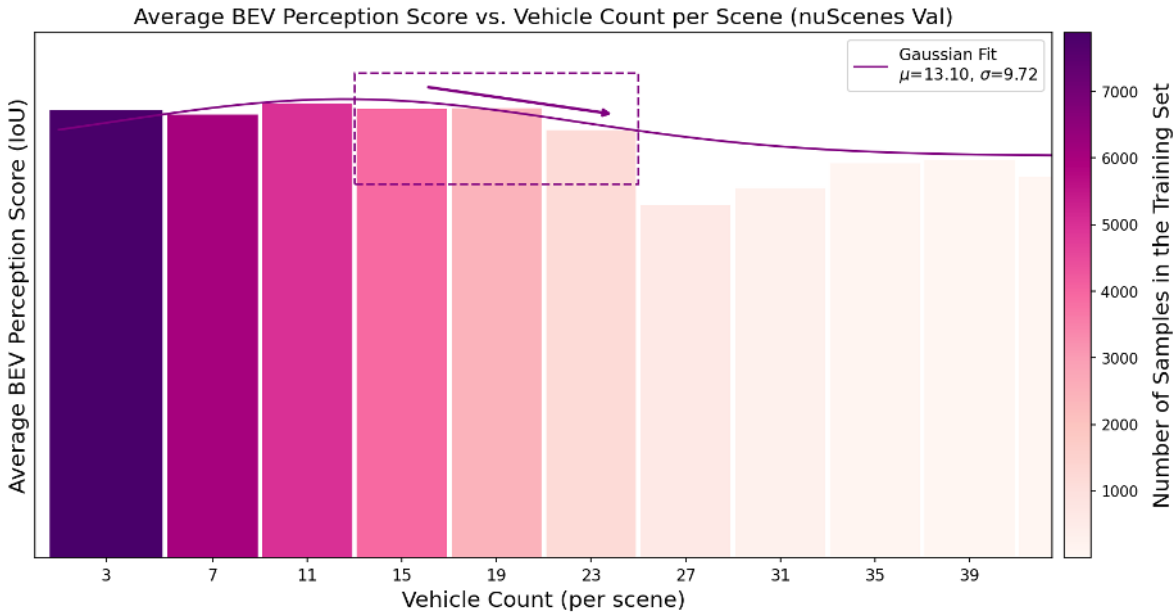


Figure 1.2: **Average IoU vs. Number of Vehicles.** Vehicle counts in scenes are grouped into bins and plotted with the average perception score on the nuScenes validation set using a pretrained BEV perception model, CVT [Pan et al., 2020]. The plot illustrates the relationship between scene crowdedness and model generalization. Each bin is color-coded based on the number of training scenes observed for that vehicle count. A Gaussian distribution (purple line) is fitted over the number of vehicles to highlight the overall trend.

1.2.2 Synthesizing Targeted Scenes with Diffusion Guidance

We select a scene parameter that is both simple and reflective of a real-world constraint—the number of vehicles in the scene—making it suitable for formulating guidance. We then identify moderately challenging scenes for BEV perception, those that fall in regions where both dataset representation and model performance begin to decline (Figure 1.2). To synthesize such scenes in a controlled manner, we propose a guided generation pipeline that specifically targets these areas in the distribution of driving layouts.

Our approach starts by learning the distribution of real-world BEV layouts using the method proposed in SLEDGE [Chitta et al., 2025]. We then introduce a custom loss function based on vehicle count to guide a diffusion-based layout generator toward producing scenes with higher traffic density. Although the generated layouts do not perfectly match the target counts, they consistently shift toward the desired density range. These layouts are then passed to the multi-view image generator MagicDrive, which produces realistic camera views for each scene.

We evaluate how well our guided scene generations reflect controllable and realistic layouts by testing them with a pretrained BEV perception model (CVT [Zhou and Krähenbühl, 2022]), observing comparable performance between our generated scenes and those based on real layouts. This means the generated layouts follow similar patterns to real data. Although augmenting the nuScenes training set with these guided scenes does not currently improve performance due to reproducibility issues, our results show that the proposed guidance mechanism can successfully push scene generation toward meaningful and underrepresented regions of the data distribution, offering a promising foundation for future work on targeted synthetic data augmentation.

1.3 Thesis Outline

This thesis is structured as follows:

Chapter 2 presents comprehensive review of camera-based BEV perception methods, vision foundation models, and multi-view image and driving scene generation approaches within the context of autonomous driving simulators. It also introduces the necessary background on diffusion models, conditioning mechanisms, and controllable generation techniques.

Chapter 3 introduces our method [Barın et al., 2024] for integrating vision foundation models into camera-based BEV perception in a parameter-efficient way to build more robust models [1]. We evaluate the effectiveness of our approach through detailed experiments, highlighting improvements in robustness, parameter efficiency,

[1]Project's GitHub repo

convergence speed, and overall performance.

Chapter 4 explores a data generation pipeline designed to synthesize moderately challenging scenes for BEV perception models. We provide an in-depth analysis of the guidance mechanism, examine its ability to lead generations toward targeted regions in the scene distribution, and discuss qualitative and quantitative results.

Lastly, Chapter 5 brings together the main results from both chapters, highlights the difficulties and limits of each method, and suggests future steps based on these points.

In summary, this thesis focuses on addressing key challenges in camera-based BEV perception by developing efficient and robust models, and introducing a targeted data generation method to enhance dataset coverage.

Chapter 2

RELATED WORK AND BACKGROUND

2.1 Camera-based BEV Segmentation

Camera-based BEV models aim to recover the 3D geometry of a scene from 2D images. Since depth information is inherently lost in monocular camera images, these models attempt to recover this missing third dimension.

BEV segmentation methods explicitly model a feature space in BEV. After processing all camera images using a backbone, these 2D image features are transformed to a 3D voxel grid space. The voxel space is the discrete representation of the 3D world, and each voxel bin defines a specific area within the 3D environment. After constructing the 3D voxel grid, the height dimension is pooled, and the features are decoded into a 2D BEV representation of the scene. A key part of this process is mapping features of 2D images into the 3D spatial domain.

We can categorize the existing work on BEV into three based on how they extract 3D information from 2D images: Depth-based, attention-based and sampling-based methods.

2.1.1 Depth-based methods

Lift-Splat-Shoot (LSS) [Philion and Fidler, 2020] is the first learning-based BEV segmentation method that proposes a differentiable depth-based view transformation. It predicts a depth probability distribution for every pixel and uses it to project 2D image features into a 3D frustum. These depth-weighted feature bins are then projected into 2D BEV feature space through a splatting operation, which performs a differentiable pooling operation that includes sorting and cumulative sum. This part is heavy in computation as it requires to operate on all frustum features of all cameras. Although LSS is an effective method, predicted depth probability distri-

bution is not accurate enough as it is supervised through only segmentation head, also not computationally efficient for how they pool voxels.

BEVDepth [Li et al., 2023] introduces LiDAR-based depth supervision and parallel CUDA pooling to improve LSS. BEVFusion [Liu et al., 2023b] similarly speeds up pooling with precomputed 3D grids and CUDA kernels. Although both propose efficient pooling via CUDA, accuracy remains limited without additional LiDAR supervision. This is due to the complexity of accurately modeling the depth for every pixel. Furthermore, voxel grids can only accumulate sparse features, which magnifies projection errors.

2.1.2 Attention-based methods

CVT [Zhou and Krähenbühl, 2022] learns to align 2D image features with 3D voxel grids with an attention mechanism, leveraging camera-aware position embeddings to implicitly learn depth. Each point in the 3D voxel grid is represented as a learnable BEV query corresponding to a specific BEV coordinate. BEV queries attends to image features via cross-attention, they use camera extrinsic and intrinsics as a positional embedding to link the related BEV queries with the image features. Thus, in this case, the geometric projection is learned implicitly. Since each voxel grid is a query, it is not possible to increase the number of voxel grids, thus the BEV resolution is lower, as it is computationally heavy for the attention. Unlike CVT's learned embeddings, LaRA [Bartoccioni et al., 2023] differently encodes per-pixel viewing rays from camera calibration as geometric priors.

BEVFormer [Bartoccioni et al., 2023] uses deformable attention to perform projection. They select reference points from 3D BEV grids, then using camera intrinsics and extrinsics, they sample image features for these reference points.

These models learn the geometry implicitly and lack interpretable projection. Also, as the number of BEV grid points increases, attention computation becomes huge.

2.1.3 Sampling-based methods

SimpleBEV [Harley et al., 2023] address these issues by sending rays from 3D voxel grids to the images and bilinearly sampling the intersecting image features. For each grid, they average the image features falling into the grid. Other than proposing a parameter-free projection method, they find out that the batch size and input resolution are critical for the performance of BEV perception models. PointBEV [Chambon et al., 2024] introduces a sparse BEV representation by first predicting 3D reference locations, then extracting features from multi-view images at their projected positions using bilinear interpolation, similar to SimpleBEV. This approach significantly reduces memory and computation costs.

Changing the direction of sampling increases the density of features in the voxel grid, relying on the quality of the features rather than the accuracy of projection. Due to these reasons, we choose a sampling-based method, SimpleBEV [Harley et al., 2023] for our analysis.

2.2 Vision Foundational Models

Foundational models trained on large-scale data provide robust representations, improving performance in various downstream tasks [Zhang et al., 2023, Blumenkamp et al., 2024, Aydemir et al., 2023, Nguyen et al., 2023] due to their inherent semantic understanding and strong generalization capabilities. Although foundational models are trained solely on 2D data, they are shown to capture some 3D information from images, as tested on multi-view correspondence and depth estimation tasks [Zhan et al., 2023, El Banani et al., 2024].

Among the foundational models evaluated, DINOv2 [Oquab et al., 2023] performs the best for the 3D tasks considered, alongside Stable Diffusion [Rombach et al., 2022], indicating its potential for 3D scene understanding.

There is recent work building on foundational models for pre-training a BEV network for occupancy prediction [Sirko-Galouchenko et al., 2024] or sensor fusion [Schramm et al., 2024] by benefiting from semantic capabilities of DINOv2 [Oquab et al., 2023]. We directly target improving BEV prediction from camera images by

adapting DINOv2.

2.3 Autonomous Driving Simulators

Although large-scale real-world datasets such as nuScenes [Caesar et al., 2020], Waymo [Sun et al., 2020] and Argoverse2 [Wilson et al., 2023] have significantly advanced autonomous driving research by capturing the complexity of real-world scenarios, collecting such data remains expensive both in time and cost. More critically, gathering examples of rare or hazardous corner cases is often infeasible, as these situations are difficult or unsafe to stage in reality. This is where simulators play a crucial role. They offer a safe, controllable environment for training and testing, enabling repeatable experiments and the synthesis of rare scenarios that would be impractical to capture in the real world. Simulator design choices are made based on whether they are used for the perception, planning, or driving policy learning.

Early physics-based simulators like CARLA [Dosovitskiy et al., 2017], one of the most widely used open-source tool, is a virtual environment that follows realistic vehicle dynamics, traffic rules, and customizable scenarios. It is especially useful for testing driving behavior in diverse scenarios, but lacks realism due to its fully synthetic nature.

In contrast, generative simulators aim for higher realism and scalability by learning directly from real-world data. The first wave of these models, GAN-based models [Kim et al., 2021], introduced controllable scene generation for both perception and planning purposes. However, these models struggle with poor image quality. VAE-based approaches [Swerdlow et al., 2024, Zheng et al., 2024] were also proposed with similar goals, but they fall short in image generation quality and suffer from limited consistency across different views.

Diffusion-based generative models have recently advanced image synthesis by offering high visual quality, stability, and controllability through iterative refinement [Dhariwal and Nichol, 2021, Ho et al., 2020, Saharia et al., 2022]. Notably, Stable Diffusion [Rombach et al., 2022] has demonstrated the practical benefits of scaling diffusion models with large datasets, and has since been widely adopted as a

starting point for downstream tasks. In our context, it offers a strong initialization for generating diverse and realistic driving scenes.

Diffusion models are particularly effective in conditional generation settings, including text-to-image [Ramesh et al., 2022] and layout-to-image [Lee et al., 2023, Chen et al., 2023], making them highly suitable for autonomous driving simulators that require control for structured scene generation. Thus, diffusion-based driving simulators [Hu et al., 2023a, Wang et al., 2024b, Gao et al., 2024b, Yang et al., 2023, Wen et al., 2024, Gao et al., 2024a, Lu et al., 2024a, Li et al., 2024, Ma et al., 2024, Russell et al., 2025] are conditioned on a variety of structured inputs such as natural language description, object layouts, actions- critical components for a driving scene.

A subset of these models, often referred to as world models [Hu et al., 2023a, Wang et al., 2024b, Gao et al., 2024b, Russell et al., 2025], are conditioned on actions or trajectories, as their primary goal is to predict future frames given past observations. These are typically used in planning or policy learning contexts. Since world models operate on camera inputs they rely on existing driving logs. To overcome this limitation, there are works [Sun et al., 2024, Chitta et al., 2025] that synthesizes the driving scene elements (lanes, pedestrians, etc.) and model their dynamics in bird's eye view. Since BEV is an abstraction of the scene, these models can generate unlimited driving scenarios in theory.

Another line of work [Yang et al., 2023, Wen et al., 2024, Gao et al., 2024a, Ma et al., 2024, Lu et al., 2024a, Li et al., 2024] focuses on realistic multi-view scene generation rather than modeling object dynamics of the driving environment. They produce synthetic data for downstream tasks, where 360 degree coverage and multi-camera consistency are essential. Conditions are structured inputs such as 3D object annotations, BEV maps, camera configurations, and natural language descriptions. The resulting multi-view images align with real camera perspectives, enabling their use in various downstream autonomous driving tasks.

In this thesis, we design a guided BEV layout generation pipeline by utilizing existing work on driving scene synthesis in BEV. Based on the generated layouts, we then generate multi-view camera images, which requires a layout-to-multiview

image generator. For this reason, we first discuss driving scenario generators in Section 2.3.1, followed by multi-view image generators in Section 2.3.2.

2.3.1 Driving Scenario Generation

Driving scene generation in bird's eye view (BEV) is important as it removes the dependency on real-world driving logs and outputs an abstract representation that most downstream tasks utilize.

Recent works [Sun et al., 2024, Chitta et al., 2025, Rowe et al., 2025] have shifted focus toward generating driving scenarios from scratch using diffusion-based models. These models typically follow a two-stage pipeline: initial scene generation and behavior simulation. The initial scene is generated from scratch in BEV, including both static (lanes) and dynamic (vehicles, pedestrians) elements. The system then simulates driving behavior conditioned on the generated scene. Apart from the scene initialization step, these models are similar to traffic simulation works [Feng et al., 2023, Zhong et al., 2023b, Pronovost et al., 2023], which simulate agent interactions in top-down view using various behavior strategies.

DriveSceneGen [Sun et al., 2024] is the first attempt to generate both static and dynamic elements using a diffusion-based model. It diffuses a rasterized BEV to generate the initial scene and then converts this scene to its vectorized form before passing it to a simulation network. However, learning a diffusion over the image space is not efficient, as most of the pixels are empty. SLEDGE [Chitta et al., 2025] also starts with rasterized scene inputs but introduces a VAE with a transformer decoder that outputs vectorized scene representations. The model then diffuses these compact latent representations and decodes them into vectors to obtain the initial scene. They apply a rule-based policy for simulation agent behaviours. Scenario Dreamer [Rowe et al., 2025] uses vectorized representations as both input and output, arguing that it is more efficient since it is more compact.

Since we are interested in the initial scene generation part, we consider SLEDGE as a good trade-off between interpretability and speed.

Table 2.1: **Multi-view layout-conditioned generators for autonomous driving.** FID scores, which reflect the realism of the generated datasets, are reported based on the nuScenes validation set.

Model	**Conference**	**Code Availability**	**Other Cond.**	**Eval. Task**	**FID**
BEVControl	arXiv'23	✗	-	Perception	24.85
BEVGen	RA-L'24	✗	-	Perception	25.54
Panacea	CVPR'24	✓	text	Perception	16.96
MagicDrive	ICLR'24	✓	text	Perception	16.20
DrivingDiffusion	ECCV'24	✗	text, flow	Perception	–
Delphi	arXiv'24	✗	text	End2End	–
SubjectDrive	AAAI'25	✗	text	Perception	15.96

2.3.2 Multiview Image Generators

Multiview generation is essential for synthesizing data in autonomous driving since BEV perception models depend on realistic and consistent inputs from multiple camera views to understand the scene's spatial and geometric relationships.

BEVGen [Swerdlow et al., 2024] is the first attempt to explore multiview generation with 3D using an autoregressive transformer. It encodes the scene and layout while leveraging spatial embeddings learned from camera configurations. BEVControl [Yang et al., 2023] learn to diffuse multiple camera images, using a UNet-based architecture. The UNet consists of two modules: one for ensuring geometric consistency by encoding perspective maps and objects, and another for maintaining appearance consistency using cross-attention. Both of these methods are lacking in generation quality.

Later works [Wen et al., 2024, Gao et al., 2024a, Li et al., 2024] have advanced the view realism by using a pre-trained latent diffusion model, StableDiffusion [Rombach et al., 2022] as a starting point. Particularly, Panacea [Wen et al., 2024] extends

generation to the temporal dimension with decomposed 4D attention to handle time and multiview consistency. It uses ControlNet for conditioning and passes road maps, bounding boxes, depth maps, and camera poses in perspective view as conditions. [Huang et al., 2024] introduces a large pool of vehicles to control vehicle appearance. MagicDrive [Gao et al., 2024a] adds an extra cross-attention module on top of UNet layers to ensure neighbor view consistency, directly encodes continuous 3D bounding box coordinates instead of perspective view. DrivingDiffusion [Li et al., 2024] injects control information by using residual connections within the UNet, avoiding the large number of parameters required by ControlNet. Finally, Delphi [Ma et al., 2024] focuses on improving temporal consistency by using cross-attention at the scene-level and instance-aware mechanisms across time frames.

We chose to use MagicDrive [Gao et al., 2024a] because we focus on generating single timestep views and MagicDrive generates quite realistic views while providing an accessible codebase (Tab. 2.1).

2.4 Diffusion and Guidance

2.4.1 Diffusion Fundamentals

Diffusion-based methods [Ho et al., 2020, Dhariwal and Nichol, 2021] have shown impressive results and are now widely used in generative modeling, especially in tasks where high-quality and controllable image synthesis is needed. At a high level, diffusion models learn to reverse a gradual noising process. They iteratively denoise the sample starting from pure Gaussian noise, to recover a structured output such as an image or a video frame.

The learning process of diffusion models starts with the **forward process**, also known as the diffusion process. Using an actual data point $\mathbf{x}_0$, noise is gradually added to produce a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$, where $\mathbf{x}_T$ is a nearly random noise. The process of adding noise is commonly modeled as a Markov chain:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{2.1}$$

where the α_t values form a noise schedule that determines how much information is destroyed at each step. The noisy sample $\mathbf{x}_t$ can be formulated directly based on the clean input $\mathbf{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.2}$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}) \tag{2.3}$$

with $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. This allows model to sample noisy images at any t directly without going through each step.

The model generates new samples by learning how to invert the forward process. This process, referred as the **backward process**, progressively removes the added noise. It learns to estimate the reverse distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, the distribution of a less noisy sample given a noisier one. At each step t, a neural network, denoted as $\boldsymbol{\epsilon}_\theta$, is trained to predict the added noise $\boldsymbol{\epsilon}$. This is typically done by minimizing a loss function that encourages accurate denoising:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon}\|^2\right] \tag{2.4}$$

During training, this loss is minimized over randomly sampled time steps and comparing the predicted noise with the clean data sample. New samples are synthesized by initializing a Gaussian noise and progressively applying the learned denoising steps for T iterations.

Latent Diffusion Models (LDMs). While diffusion models are extremely powerful, applying them directly high-resolution images requires heavy computation. To overcome this limitation, LDMs [Rombach et al., 2022] is proposed to operate in a compact latent space. They encode the input through a pretrained autoencoder instead of operating directly on pixel-level data. The diffusion process is carried out in this compressed latent space, and the result is subsequently decoded to reconstruct the image. The training objective stays the same, but the model now predicts noise in the latent space:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{z_0,\boldsymbol{\epsilon},t}\left[\|\boldsymbol{\epsilon}_\theta(z_t, t) - \boldsymbol{\epsilon}\|^2\right] \tag{2.5}$$

This significantly reduces memory requirements and speeds up training, without lowering output quality.

2.4.2 Diffusion Control

Diffusion models allow various control over their output. They can use variety of inputs such as text [Saharia et al., 2022] or layout [Lee et al., 2023, Zheng et al., 2023, Chen et al., 2023] as conditions. It is called controllable generation, where the model learns not just $p(\mathbf{x})$, but $p(\mathbf{x}|y)$ for some condition y. Conditioning is typically implemented by providing y as an additional input to the UNet at every timestep:

$$p(\mathbf{x}_{0:T}|y) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, y) \tag{2.6}$$

Here, y can be a wide range of inputs: class labels, text embeddings, semantic maps, or even multimodal data. In practice, rather than modeling the conditional distribution at each step, models are trained to predict the noise. The clean image can be expressed as the following form:

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) \right) \tag{2.7}$$

where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y)$ is a neural network that predicts the noise added at timestep t, conditioned on y, and $\bar{\alpha}_t$ denotes the cumulative product of the forward process noise scheduler.

Conditioning. For conditioning, rather than simply concatenating or injecting y as a feature vector, modern architectures, including Stable Diffusion [Rombach et al., 2022], allow intermediate feature maps of UNet to interact with conditioning input (e.g., CLIP embeddings) via *cross-attention*. Let $\mathbf{f}$ be the feature map at a given layer in the UNet, and let $\mathbf{y}$ be the sequence of conditioning embeddings. Each attention block components are defined as:

$$Q = W_Q \mathcal{F}(\mathbf{f}) \tag{2.8}$$

$$K = W_K \mathbf{y}_{\text{emb}} \tag{2.9}$$

$$V = W_V \mathbf{y}_{\text{emb}} \tag{2.10}$$

where the queries Q are linear projections of the flattened UNet feature map $\mathcal{F}(\mathbf{f})$ at an intermediate layer, while the keys K and values V are derived from the conditioning input $\mathbf{y}_{\text{emb}}$, output of a domain-specific encoder applied to $\mathbf{y}$. Thanks to the cross-attention mechanism, each spatial location in the feature map learn to focus on relevant parts of the condition.

This conditioning is learned during training. This means that if we want to inject additional conditions, we need to train the whole model from scratch. Considering that powerful diffusion models are very large and require a lot of computation, training them is not practical. This is why conditioning methods that make use of already trained models are valuable, one of the most popular being ControlNet.

ControlNet. ControlNet extends a pretrained diffusion model by introducing a separate branch that handles extra conditioning inputs such as segmentation maps, or BEV layouts, in addition to the UNet's original prompt. The output of this branch is then fused with the main UNet's internal activations through zero-initialized convolution layers. This protects the weights of the large model from harmful noise while gradually injecting the condition c into the main network:

$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y, c) = \text{UNet}_\theta(\mathbf{x}_t, t, y) + \text{ZeroConv}(f_c(c)) \tag{2.11}$$

ControlNet is quite effective, as it is a lightweight network that makes use of the powerful weights of a large-scale diffusion model.

While training-time conditioning proves to be effective for structure-aware diffusion outputs, it requires training. Furthermore, if the conditioning signal is weak, the model could ignore it. To address these problems, *guidance* has been proposed as an alternative form of control that modifies the sampling process explicitly to enforce the desired criteria.

Guidance methods can be categorized whether they are applied during training or inference. Some approaches, such as *classifier-free guidance*, modify the training procedure itself, making the model learn conditional and unconditional behaviors by randomly dropping the conditioning input. Others, like *classifier guidance* or *loss-based guidance*, are applied during sampling by adjusting the backward diffu-

sion process to better satisfy the condition. These strategies are particularly useful when retraining the model is impractical or when new forms of control need to be introduced without modifying the pretrained model.

Classifier Guidance. One of the earliest approaches is classifier guidance [Ho et al., 2020] , where an external classifier $p(y|\mathbf{x}_t)$ is trained to predict the conditioning label from noisy data samples. During sampling, this classifier provides a gradient that encourages the denoised samples to be more consistent with the target condition:

$$\tilde{\epsilon}_\theta(x_t, t) = \epsilon_\theta(x_t, t) - s \cdot \sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t), \tag{2.12}$$

where $p_\phi(y \mid x_t)$ is the classifier output and s is the guidance scale. The second term acts as a feedback from the classifier. This approach is impractical as it requires training a robust classifier that can operate at all levels of noise, which is especially challenging for complex conditions.

Classifier-Free Guidance. With classifier-free guidance, there is no need to train an additional external classifier. This method incorporates both conditional and unconditional generation into a single diffusion training process. For learning unconditional, condition is y removed randomly during training. This teaches the model to operate in both modes. At sampling, the predicted noise from the conditional and unconditional paths are interpolated:

$$\boldsymbol{\epsilon}_{\text{guided}} = (1 + w) \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - w \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \tag{2.13}$$

Here, w is the guidance scale that adjusts the alignment with the condition. Classifier-free guidance has become a popular choice in conditional diffusion models for its ease of implementation and strong performance.

Loss-Based Guidance. Finally, *universal guidance* [Bansal et al., 2023] proposed to enforce any differentiable loss function to obtain the necessary feedback through the gradients during sampling. Loss is backpropagated to adjust the denoising process. These methods are highly flexible and can be applied without retraining the

diffusion model. One downside of this method is that the final sample quality depends heavily on the strength of the guidance and optimization parameters.

Formally, let $\mathcal{L}_{\text{task}}(\mathbf{x}_t)$ denote a differentiable task-specific loss function evaluated on the denoised prediction at timestep t. The gradient of this loss is used to modify the noise prediction as follows:

$$\boldsymbol{\epsilon}_\theta^{\text{guided}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \lambda \nabla_{\mathbf{x}_t} \mathcal{L}_{\text{task}}(\mathbf{x}_t) \tag{2.14}$$

Here, λ is a scaling factor that controls the strength of the task loss on the denoising process. This approach leads the denoising process toward regions of the sample space that optimize the desired property defined by $\mathcal{L}_{\text{task}}$.

2.5 Loss-Based Diffusion Guidance for Driving

In autonomous driving, loss-based diffusion guidance has been applied to trajectory prediction [Jiang et al., 2023] and traffic scene generation [Zhong et al., 2023b, Zhong et al., 2023a, Chang et al., 2024, Lu et al., 2024b] by incorporating objectives during sampling. These objectives aim to enforce physical realism and safety. For example, some works [Jiang et al., 2023, Zhong et al., 2023b] penalize collisions between agents, while others intentionally guide the generation toward collisions with the ego-vehicle to simulate safety-critical scenarios [Chang et al., 2024]. Traffic rules can also be encoded using Signal Temporal Logic (STL), such as enforcing speed limits or preventing off-road driving [Zhong et al., 2023b]. Guidance can also applied using high-level user intent expressed in natural language and processed by large language models (LLMs) [Zhong et al., 2023a]. [Lu et al., 2024b] further expand user control over agents by injecting guidance signals that steer actors toward specific spatial regions (e.g., intersections) or constrain their attributes (e.g., preventing heavy vehicles from taking sharp turns at high speed).

Chapter 3

ROBUSTNESS FOR CAMERA-BASED BEV SEGMENTATION

3.1 Methodology

3.1.1 Overview

Our goal is to integrate visual foundation model DINOv2 [Oquab et al., 2023] into BEV prediction model, SimpleBEV [Harley et al., 2023]. In this section, we first explain our adaptation strategy, Low Rank Adaptation (Section 3.1.2) and then SimpleBEV (Section 3.1.3) for completeness, and finally, present our approach to integrate DINOv2 into the SimpleBEV framework (Section 3.1.4).

3.1.2 Low Rank Adaptation (LoRA)

Low Rank Adaptation [Hu et al., 2022], i.e. LoRA, is widely used in Natural Language Processing to adapt pretrained large models to different tasks. LoRA is parameter efficient compared to fine-tuning, as only low-rank matrices are trained as residuals to frozen weights. Following the approach in MeLo [Zhu et al., 2023], we update only the query and value projections in all attention layers in the ViT. Formally, given a pre-trained weight matrix $\mathbf{W}_{\{Q,V\}} \in \mathbb{R}^{d\times d}$, we obtain $\mathbf{W}'$ as the result of adaptation:

$$\mathbf{W}'_{\{Q,V\}} = \mathbf{W}_{\{Q,V\}} + \mathbf{BA} \tag{3.1}$$

where $\mathbf{A} \in \mathbb{R}^{r\times d}$ and $\mathbf{B} \in \mathbb{R}^{d\times r}$ are learned matrices, r is the rank, and d is the feature dimension. We train only the $\mathbf{B}$ and $\mathbf{A}$ matrices while keeping the original $\mathbf{W}$ frozen for each attention layer.

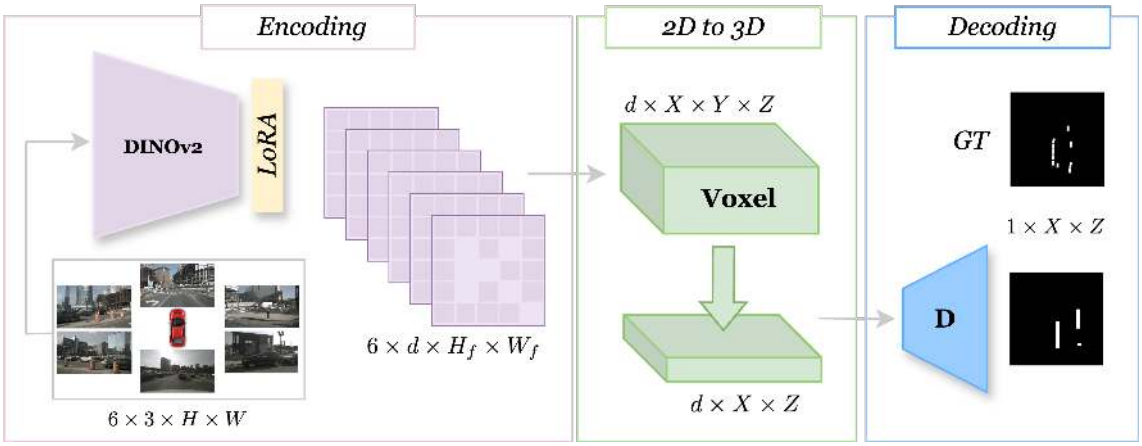


Figure 3.1: **Robustness Overview.** In this work, we propose to adapt DINOv2 to BEV segmentation using Low-Rank Adaptation (LoRA) for a robust BEV model. There are three main steps: i) We encode the camera images using DINOv2 to obtain tokens for each view, with attention weights updated through LoRA. ii) Transform image features from 2D to 3D using pull mechanism proposed by [Harley et al., 2023]. iii) Decode BEV features to 2D vehicle BEV masks.

3.1.3 SimpleBEV

Given images from N cameras, SimpleBEV [Harley et al., 2023] first extracts a feature map $\mathbf{f}_i \in \mathbb{R}^{d \times H_f \times W_f}$ for each image $i \in \{1, 2, \ldots, N\}$, where d represents the feature dimension, and $H_f \times W_f$, the size of the feature map. The method then employs a parameter-free lifting technique to transform image features into a 3D voxel grid $\mathbf{V} \in \mathbb{R}^{d \times X \times Y \times Z}$, where X, Y, and Z correspond to the width, height and depth of the grid, respectively.

For each voxel $\mathbf{V_p} \in \mathbb{R}^d$ in the grid, represented by the 3D coordinate $\mathbf{p} = [x, y, z]$, corresponding 2D pixel coordinates $\mathbf{q}_i = (u_i, v_i)$ are computed by applying the camera projection defined by the intrinsic matrix $\mathbf{K}_i$ and extrinsic matrix $\mathbf{E}_i$ to the 3D point $\mathbf{p}$

$$\mathbf{q}_i = \mathbf{K}_i \mathbf{E}_i \mathbf{p} \tag{3.2}$$

The feature values are then bilinearly sampled from the feature map $\mathbf{f}_i$ at these projected coordinates, $\mathbf{q}_i$. Then, sampled features from all N cameras are aggregated

and assigned to the corresponding voxel on the 3D grid:

$$\mathbf{V_p} = \frac{1}{N} \sum_{i=1}^{N} \text{sample}(\mathbf{f}_i, \mathbf{q}_i) \tag{3.3}$$

After aggregating features for all $\mathbf{V_p} \in \mathbf{V}$ and constructing a 3D voxel grid $\mathbf{V}$ that encapsulates the entire scene, a compressor reduces this voxel grid into a 2D BEV feature map $\mathbf{B} \in \mathbb{R}^{d \times X \times Z}$. The decoder then predicts the probability of occupancy for each grid cell. The model is trained using binary cross-entropy loss to optimize these predictions.

3.1.4 Adaptation of DINOv2 to BEV

Integrating DINOv2

We perform two main steps to integrate DINOv2 into SimpleBEV. First, we introduce additional layers in the query and key components of the ViT-based DINOv2 for adaptation, as described in Section 3.1.2. Next, we replace the feature map $\mathbf{f}_i \in \mathbb{R}^{d \times H_f \times W_f}$ with the output tokens of DINOv2 for each corresponding camera view, followed by pooling as shown in (3.3). During training, only the additional layers are updated, while the original DINOv2 model remains frozen. SimpleBEV, with its original ResNet-101 backbone, serves as the baseline for comparison.

Evaluation Aspects

We assess the quality of our adaptation by focusing on three key aspects in the evaluation:

i) First, we consider **input resolution**, both image and feature map, where the ability to achieve high performance with lower-resolution inputs is advantageous, as it demonstrates efficiency in processing.

ii) Second, we evaluate the **number of learnable parameters**, which not only indicates the resource efficiency of training but also reflects the robustness and general-purpose nature of the input features. Models with fewer parameters

that still perform well suggest that the features are inherently strong, as they require minimal transformation.

iii) Lastly, we examine **convergence speed**, favoring models that reach optimal performance quickly, as this reduces computational and time-related costs during training. Convergence is measured by the number of updates needed, with faster convergence indicating more effective use of general-purpose features.

As a result, we prioritize models that excel in using lower-resolution inputs, have fewer learnable parameters, and converge more quickly, as these traits highlight the efficiency and robustness of our adaptation.

3.2 Experiments

3.2.1 Experimental Setup

Datasets

nuScenes: We conducted our experiments on the nuScenes [Caesar et al., 2020] dataset, which is widely used for training and evaluating camera-based BEV segmentation methods. It consists of 1,000 scenes from two cities, annotated at 2Hz, with RGB images captured by six cameras. The dataset contains 28130 instances in the training set and 6019 instances in the validation set.

nuScenes-C: For robustness analysis, we conducted experiments on the nuScenes-C benchmark [Xie et al., 2023], which is designed to measure the robustness of camera-based BEV perception models across eight types of corruptions under three levels of severity. The corruptions in this dataset are grouped into 8 categories: *brightness, darkness, fog, snow, motion blur, color quantization, camera crash,* and *frame loss.* For detailed descriptions of these augmentations, please refer to RoboBEV [Xie et al., 2023].

Evaluation Metrics

Following prior work, we use the mean Intersection-over-Union (mIoU) to evaluate model performance, measuring the overlap between our predictions and the ground truth boxes.

Training Details

We train the adaptation models using ViT-B and ViT-L architectures with the AdamW optimizer, a learning rate of 1×10^{-3}, and one cycle scheduler. For fine-tuning, we use a lower learning rate, 1×10^{-5}, and set the effective batch size to 16. For the ViT-B model, we followed the default training schedule of 25K updates as suggested in [Harley et al., 2023]. For the ViT-L model, we additionally explored shorter training time, using approximately one-third of the default steps to demonstrate the effect of adaptation on convergence with a large backbone.

3.2.2 Adaptation

We compare the performance of our adaptation models to the original SimpleBEV results in Table 3.1, 3.2 and 3.3. Our comparisons focus on three key aspects:

i) Same input resolution, ensuring the models are evaluated under identical conditions

ii) Same feature map resolution to assess the efficiency of the spatial features encoded by the models

iii) Update iterations to understand the convergence speed and training efficiency of the models.

Same Input Resolution

Vision-based BEV perception models rely entirely on image input, making image resolution critically important. As demonstrated in [Harley et al., 2023], increasing the input resolution can improve performance up to a certain limit. Considering

Table 3.1: **Quantitative Results on nuScenes Validation Set wrt. Image Resolution.** This table compares the results of SimpleBEV and DINOv2 adaptations with different backbones in terms of image resolution.

Model	**Backbone**	**Image Res.**	**mIoU**	**Exp.**
SimpleBEV	ResNet-101	224 × 400	42.3	A
DINOv2	ViT-B	224 × 392	42.3	C
	ViT-L		**43.4**	D
SimpleBEV	ResNet-101	448 × 800	47.4	B
DINOv2	ViT-L	448 × 784	**47.7**	G

these, we choose to experiment on 224×400 and 448×800. We compare the models with nearly identical input resolutions[1] to assess their effectiveness when processing a similar number of pixels for a fair evaluation (Table 3.1) . For the smaller resolution, SimpleBEV and DINOv2 ViT-B reach the same level of performance, 42.3 mIoU (A vs. C), while ViT-L surpasses SimpleBEV by 1.1 points (A vs. D). For the higher resolution, ViT-L outperforms SimpleBEV by a small margin, with mIoUs of 47.4 and 47.7, respectively (B vs. G). This demonstrates that the adapted DINOv2 backbones can reach the performance of SimpleBEV using similar resolutions.

Same Feature Resolution

In addition to input resolution, we also consider feature map resolution (Table 3.2). By this, we refer to the resolution of $\mathbf{f}_i$ as introduced in Section 3.1.3, which is the output of the backbone. Different backbones operate with different strides, i.e. the downsampling ratio of the final feature map. A backbone with a lower stride is expected to be advantageous [Karaev et al., 2024], as it can better preserve the details of spatial information. Specifically, the SimpleBEV downsamples the image

[1]The input resolution for ViTs should be divisible by the patch size, which is 14 in the case of DINOv2. Therefore, we use the closest multiples of 14 to match the target resolution.

to $1/8^{th}$ of the original input resolution, while DINOv2 downsamples to $1/14^{th}$ due to patch size. Considering the same feature resolution of 28×50, the adapted ViT-L outperforms SimpleBEV by a significant margin of 5.3 IoU (A vs. F). This indicates that the adapted DINOv2 backbone can preserve spatial information more efficiently.

Table 3.2: **Quantitative Results on nuScenes Validation Set wrt. Feature Resolution.** This table compares the results of SimpleBEV and DINOv2 adaptations with different backbones in terms of feature resolution.

Model	**Backbone**	**Feature Res.**	**mIoU**	**Exp.**
SimpleBEV	ResNet-101	28×50	42.3	A
DINOv2	ViT-L	28×50	**47.6**	F

Moreover, increasing the feature resolution consistently improves performance, as expected, by providing more accurate and dense interpolation among features. For instance, raising the resolution from 16×28 to 28×50 results in a 4.2 mIoU increase (D vs. F). Additionally, a further resolution increase to 32×56 yields a slight improvement, raising the mIoU from 47.6 to 47.7 (F vs. G).

Number of Updates

We chose the DINOv2 ViT-L model for convergence experiments due to its large scale, making it ideal for assessing convergence efficiency. The performance of DINOv2 ViT-L shows a drop of 0.2 IoU when trained only for one-third of the total iterations (D vs. E). In contrast, ViT-L not only surpasses SimpleBEV but does so even with fewer training iterations, given the same input resolution (A vs. E) and the same feature resolution (A vs. F). These results indicate that DINOv2 adaptation can converge quickly, and shorter training times have minimal impact on performance. (Table 3.3)

Table 3.3: **Convergence speed on nuScenes.** The ●●● represents the number of iterations in the full training of SimpleBEV, which corresponds to 25K steps, while ● represents one-third of it. See text for details.

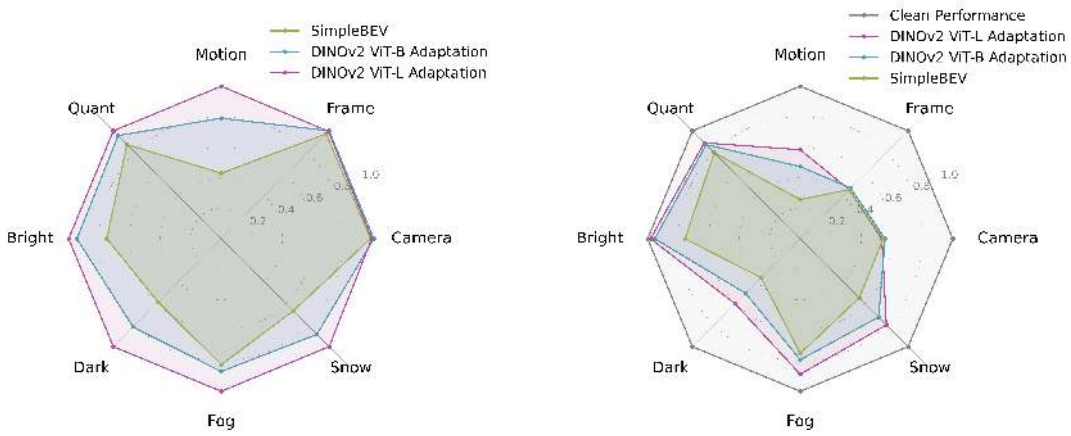
Model	**Backbone**	**Update Steps**	**mIoU**	**Exp.**
SimpleBEV	ResNet-101	●●●	42.3	A
DINOv2	ViT-L	●●●	43.4	D
		●	43.2	E

3.2.3 Robustness Evaluation

In Figure 3.2, we present a robustness analysis comparing the SimpleBEV baseline with our two adaptation variants, ViT-B and ViT-L. We evaluate the models under various corruptions from the nuScenes-C dataset and report their performance separately to assess how each model handles different types of corruptions. For a fair comparison, all models are trained at similar resolutions: 224×400 for SimpleBEV and 224×392 for the ViT backbones, as discussed in Section 3.2.2.

In Figure 3.2a, the performance of all methods is normalized based on the best-performing model for each type of corruption. For color quantization, frame loss, and camera crash, all methods perform similarly. However, the ViT-L adaptation significantly outperforms other methods in most corruption types, surpassing SimpleBEV by at least 20%, with ViT-B also showing strong results. The difference is particularly pronounced in the case of motion blur, where SimpleBEV operates at only 40% of the ViT-L adaptation's performance.

Figure 3.2b illustrate the performance drop of each model relative to its performance on clean data. This figure highlights the relative decline in performance for each model when exposed to various corruptions. As seen in the previous analysis, the ViT-B and ViT-L adaptations exhibit less performance degradation compared to SimpleBEV. Excluding the camera crash and frame loss scenarios, the ViT-L adaptation consistently maintains its performance, never dropping below 60% of its clean performance, with degradation of less than 20% for brightness, fog, and



(a) Performances under corruptions (b) Performance Drop under Corruptions

Figure 3.2: **Robustness Analysis on nuScenes-C**. We compare the models under different types of corruptions in a, where each axis is normalized over the maximum performing model, i.e. ViT-L adaptation. We show the performance drop of models relative to their performance on clean data in b, where each axis is normalized to the clean data performance of the corresponding model.

quantization. For the ViT-B adaptation, the threshold is around 45%. In contrast, SimpleBEV struggles to maintain its performance, with drops below 40% in motion blur conditions and below 30% in darkness. Overall, the DINOv2 adaptations demonstrate greater robustness than SimpleBEV in six out of eight corruption types and are comparable in the remaining two, camera crash and frame loss, highlighting the superior robustness of the adaptation approach. This finding underscores the value of exploring foundational models like DINOv2 for enhancing robustness in BEV perception tasks.

3.2.4 Ablation Study

Training Method

To highlight the impact of LoRA, we conducted experiments with the ViT-B DINOv2 adaptation across three configurations, as shown in Table **??**: i) Frozen, where

Table 3.4: **Training Method and Parameter Efficiency.** This table shows the results of different weight update strategies with the corresponding number of learnable parameters. Note that there are additional 5M parameters for the decoder.

Backbone	**Method**	**#Params**	**mIoU**
ResNet-101	-	37M	42.3
ViT-B	Frozen	0	34.3
ViT-B	Fine-tune	86M	41.5
ViT-B	LoRA	1M	42.3
ViT-L	LoRA	3M	43.4

only the decoder (with 5M parameters) is trained and no additional learnable parameters are introduced, i.e. no updates to the backbone; ii) Fine-tuning, where all 86M parameters of the ViT-B DINOv2 backbone are updated; and iii) LoRA, where a small set of parameters is learned within the attention layers of the backbone.

The experiments reveal that, while the frozen model demonstrates a decent zero-shot representation performance, it significantly lags behind SimpleBEV (34.3 vs. 42.3). Fine-tuning the entire backbone offers improved results but demands significant computational resources due to the large number of learnable parameters (86M). In contrast, the LoRA configuration, which adds just 1M parameters to the decoder, achieves results that are on par with (ViT-B; 42.3) or even superior (ViT-L; 43.4) to SimpleBEV (42.3). This is achieved by updating only 1.12% of the parameters in ViT-B and 2.70% in ViT-L. Notably, LoRA outperforms the full fine-tuning by 0.8 points, showcasing its parameter efficiency and effectiveness in enhancing the performance.

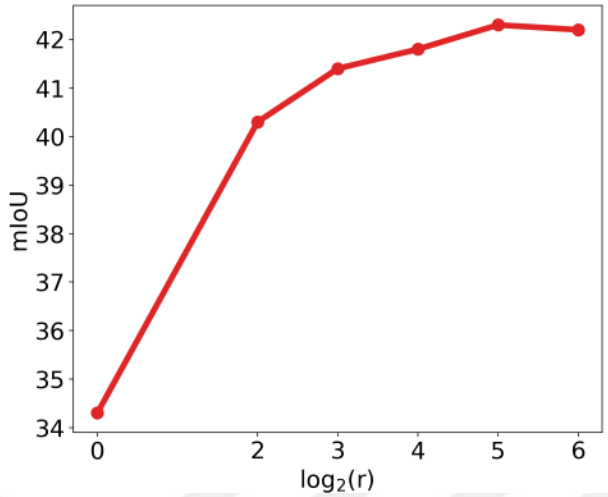


Figure 3.3: **Varying the Rank of LoRA.** This plot illustrates the effect of increasing the LoRA rank (in log scale) on the performance, with rank 0 representing a frozen backbone.

Rank of LoRA

We experimented by varying the rank of adaptation in LoRA as shown in Figure 3.3. Rank essentially controls the capacity of the adaptation, with higher ranks providing more parameters for fine-tuning. Higher ranks can capture more complex relationships and lead to better performance, while potentially losing the information from pre-training. A rank of 0 corresponds to no updates to the backbone, i.e. frozen backbone. We found that increasing the rank leads to consistent improvements up to a certain point, specifically up to rank 32. However, increasing the rank from 32 to 64 results in a performance decrease of 0.1. A potential reason for this is the loss of the inductive bias of DINOv2, due to the larger updates on the attention weights. This indicates that rank 32 strikes an optimal balance between adapting the features to the task and preserving the valuable information acquired during pre-training.

Chapter 4

TARGETED SCENE LAYOUT GENERATION WITH DIFFUSION GUIDANCE

4.1 Methodology

4.1.1 Overview

To enable targeted data generation, we first analyze the relationship between scene crowdedness, BEV perception performance, and the number of samples in the dataset (Section 4.1.2). This analysis helps us identify a region in the data distribution where model performance is suboptimal and dataset coverage is limited. To address this, we propose a guided BEV layout generation pipeline designed for layout-conditioned multi-view image generators.

Our pipeline consists of the following stages (illustrated in Fig. 4.1):

i) Following SLEDGE [Chitta et al., 2025], we train a rasterized-to-vectorized autoencoder (RVAE) on the nuScenes dataset using both rasterized scene inputs (RSI) and their vectorized counterparts. This forms a shared latent space capturing the distribution of driving scenes in bird's-eye view (BEV). We then train a latent diffusion model $\mathcal{E}_{\mathrm{RLM},\theta}$ to generate latent representations (RLMs) in this space. This allows sampling of new scenes following the training distribution. (Section 4.1.3)

ii) To control semantic properties of the generated layouts, particularly vehicle count, we introduce a guidance mechanism into the sampling process. At each denoising step, the latent is decoded, a differentiable loss is computed, and the trajectory is adjusted to push the generation towards the target condition. (Section 4.1.6)

iii) The guided latent is decoded into a structured layout containing 3D bounding boxes, semantic maps, camera poses, and scene-level text prompts, which are used as input to MagicDrive [Gao et al., 2024a]. (Section 4.1.4)

iv) MagicDrive synthesizes realistic, multi-view camera images that are geometrically consistent with the generated layout and scene attributes. (Section 4.1.4)

v) To validate the usability of the generated data, we train a BEV perception model by using the generated multi-view camera images as input and the corresponding rasterized layouts as supervision targets. This demonstrates that the synthetic data can be directly integrated into existing BEV perception pipelines. (Section 4.1.6)

This pipeline enables controlled and scalable generation of diverse driving scenes that match specific conditions and supports downstream training of BEV perception models.

4.1.2 Perception Score Analysis

Identifying a Target Region for Data Generation. To integrate guidance, following the literature discussed in Loss-Based Guidance, we aimed to select a constraint that is non-learnable, simple, and reflective of a real-world condition such as speed limits or non-collision requirements. In this context, we investigate how the number of vehicles in a driving scene influences the performance of a pretrained BEV perception model. For our analysis, we created bins of 4. As shown in Fig. 1.2, the perception score exhibits a non-monotonic trend: while it initially improves with increasing vehicle count, it starts to decline beyond a certain point. When we fit a Gaussian distribution to the perception scores, we observe that scenes with higher number of vehicles, particularly beyond the mean, are not only associated with lower performance but are also underrepresented in the dataset. This might suggests a potential link between *data scarcity* and *decreased model performance* in crowded scenarios.

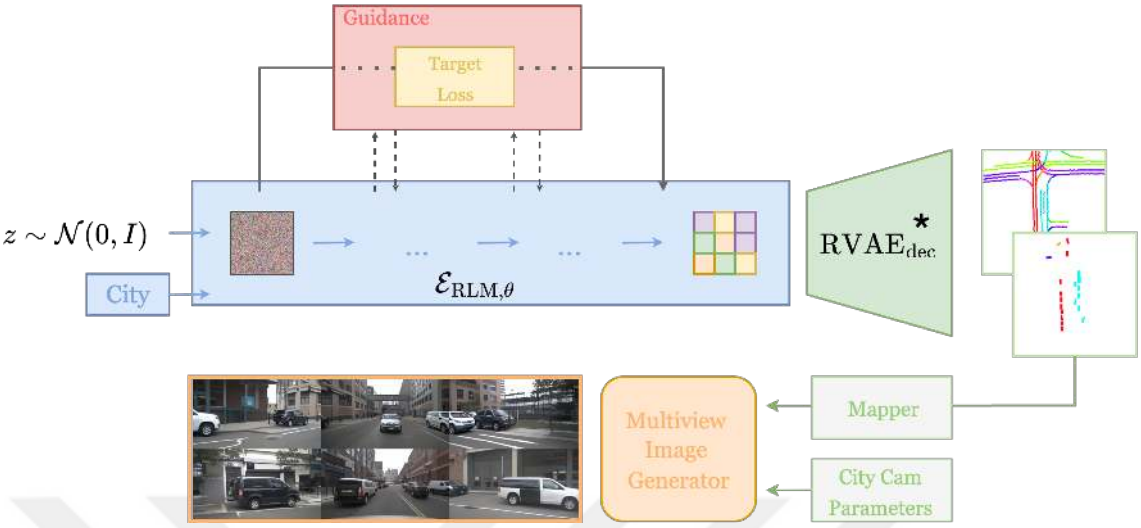
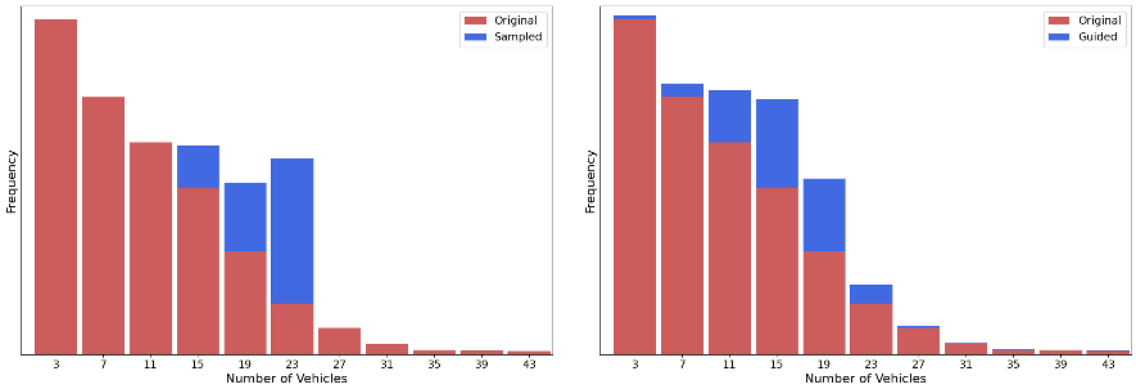


Figure 4.1: **Guidance Overview.** In this work, we propose a guided layout generation pipeline for targeted synthetic data generation. The process consists of four main steps: (i) We sample Gaussian noise and generate layouts using the latent denoiser $\epsilon_{\text{RLM},\theta}$, guiding each diffusion step with our target loss. (ii) The guided latent is decoded using RVAE_{dec} to obtain a scene configuration. (iii) The generated scene configuration is passed to a mapper to convert the layout into the input format required by the multi-view image generator. (iv) Finally, using city parameters and the generated configuration, we synthesize multi-view camera images that correspond to the generated layouts.

To better understand this underrepresented region, we focus on bins centered at vehicle counts of 15, 19 and 23 (14 to 25 vehicles) as a proof of concept. These bins lie on the declining tail of the performance distribution, where the number of training samples is lower and BEV perception score starts to decrease. We intentionally avoid targeting more extreme cases such as scenes with very high vehicle counts, which are both rarer and considerably harder to model. Instead, our goal is to target moderately challenging scenes that are currently underrepresented, with the aim of generating synthetic data that can fill this gap.



(a) Target vehicle counts overlaid on the original distribution.

(b) Actual distribution after guidance.

Figure 4.2: **Comparison of target vehicle count distributions vs actual distribution after guidance.** Left shows the sampled target counts on top of the original distribution, and right shows the result after applying guidance in the 14–25 vehicle count range.

Sampling from the Target Region. To effectively sample from target region, we assign sampling weights to each vehicle count based on the BEV perception score (IoU) and the number of existing samples in its corresponding bin. Our sampling strategy prioritizes bins that are both underrepresented and associated with lower BEV perception score. Each vehicle count is first mapped to its corresponding bin, and the sampling weight is computed as the inverse of the product of the bin's mean IoU and sample count:

$$w_c = \frac{1}{\mathrm{IoU}_{b(c)} + \epsilon} \cdot \frac{1}{N_{b(c)} + 1} \tag{4.1}$$

where $b(c)$ denotes the bin associated with count c, $\mathrm{IoU}_{b(c)}$ is the average perception score within that bin, and $N_{b(c)}$ is the number of scenes already drawn from it. The resulting weights w_c are normalized to form a probability distribution over the counts. This approach ensures more frequent sampling from regions that are both poorly represented and harder for the perception model. We visualize the effect of this strategy by comparing the original and sampled vehicle count distributions in

Figure 4.2a.

4.1.3 SLEDGE

As with all driving scene synthesis methods discussed in Section 2.3.1, SLEDGE [Chitta et al., 2025] begins by synthesizing an initial driving scene and then performs a behavior simulation based on that scene. For the scope of this thesis, we focus on the first part. The goal is to generate realistic and diverse driving scenarios in bird's eye view (BEV), using a structured scene representation extracted from the nuPlan [Caesar et al., 2021] dataset.

In SLEDGE, the driving scene S is composed of two primary types of entities: polyline-based and bounding-box-based. Polyline entities include lanes (L), red (R) and green (G) traffic lights, each encoded as a sequence of 20 BEV points in $\mathbb{R}^{20\times2}$. Bounding box entities include pedestrians (P), vehicles (V), and static objects (O), each parameterized as a 6D vector (x, y, θ, w, l, s), where (x, y) denotes the 2D center position, θ is the heading angle, (w, l) are the width and length of the box, and s is the optional speed. The ego vehicle's velocity is also included as a 2D vector $v \in \mathbb{R}^2$.

SLEDGE uses a rasterized formulation to express the scene, suitable for CNN-based backbones. A function $\rho : S \to RSI$ encodes the scene into a rasterized state image $RSI \in \mathbb{R}^{W\times H\times12}$, where each entity type is assigned two image channels. For polylines, the channels store directional vectors $\Delta = [dx, dy]$ pointing to the next point in the sequence. For bounding boxes, rasterization encodes spatial extent, heading, and location, while dynamic agents include velocity and static ones encode orientation. The ego vehicle is represented at the image center. We present RSI examples in Figure 4.4.

To reconstruct structured elements from this rasterized input, SLEDGE employs a rasterized-to-vectorized autoencoder (RVAE) as illustrated in Figure 4.3. A ResNet-50 encoder extracts latent features from the RSI, yielding a rasterized latent map (RLM), i.e.,

$$\text{RVAE}_{\text{enc}} : \text{RSI} \to \text{RLM} \tag{4.2}$$

The RLM is then tokenized and passed through a transformer decoder, RVAE_{dec},

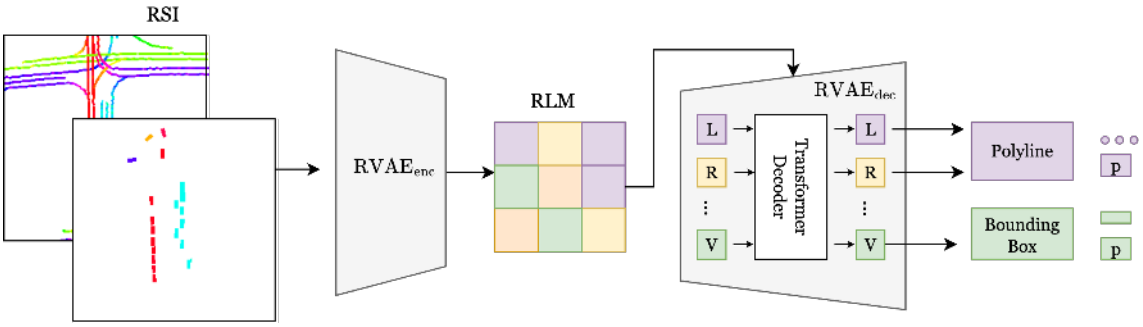


Figure 4.3: **Rasterized-to-Vectorized Autoencoder Architecture.** SLEDGE uses a rasterized-to-vectorized autoencoder (RVAE) to extract structured representations from rasterized scene inputs (RSI). The encoder RVAE_{enc} maps RSI to a latent rasterized layout map (RLM), which is decoded by RVAE_{dec} using transformer queries to predict polylines, bounding boxes, and their confidence scores.

which uses a fixed number of learnable queries per entity type to predict their attributes. Specifically, polylines are decoded as a sequence of 20×2 points and bounding boxes as a 6D vector, each accompanied by a confidence score $p \in [0, 1]$. Formally, the decoder maps the latent representation to vectorized forms as:

$$\text{RVAE}_{\text{dec}} : \text{RLM} \rightarrow (\hat{L}, \hat{R}, \hat{G}, \hat{P}, \hat{V}, \hat{O}) \tag{4.3}$$

where each $\hat{X}$ denotes the predictions for entity type $X \in \{L, R, G, P, V, O\}$. This pipeline forms an autoencoder where *rasterized* and *vectorized* forms are used jointly to train a unified latent representation.

After training the RVAE, a DiT-based generative model $\mathcal{E}_{\text{RLM},\theta}$ is trained to diffuse the latent representations (RLMs) extracted from pretrained RVAE. The model learns to generate an RLM $\mathbf{z}_0$ conditioned on a city token $\mathbf{c}$ modeled as $p(\mathbf{z}_0 \mid \mathbf{c})$, which encodes the geographic and traffic characteristics of a specific city—such as traffic flow direction or road layout orientation.

Our Modifications. While we adopt the overall framework of SLEDGE, we introduce several key modifications tailored to our goal of improving multi-view image generation from BEV layouts. First, we redefine the scene as $S = \{V, L\}$, where

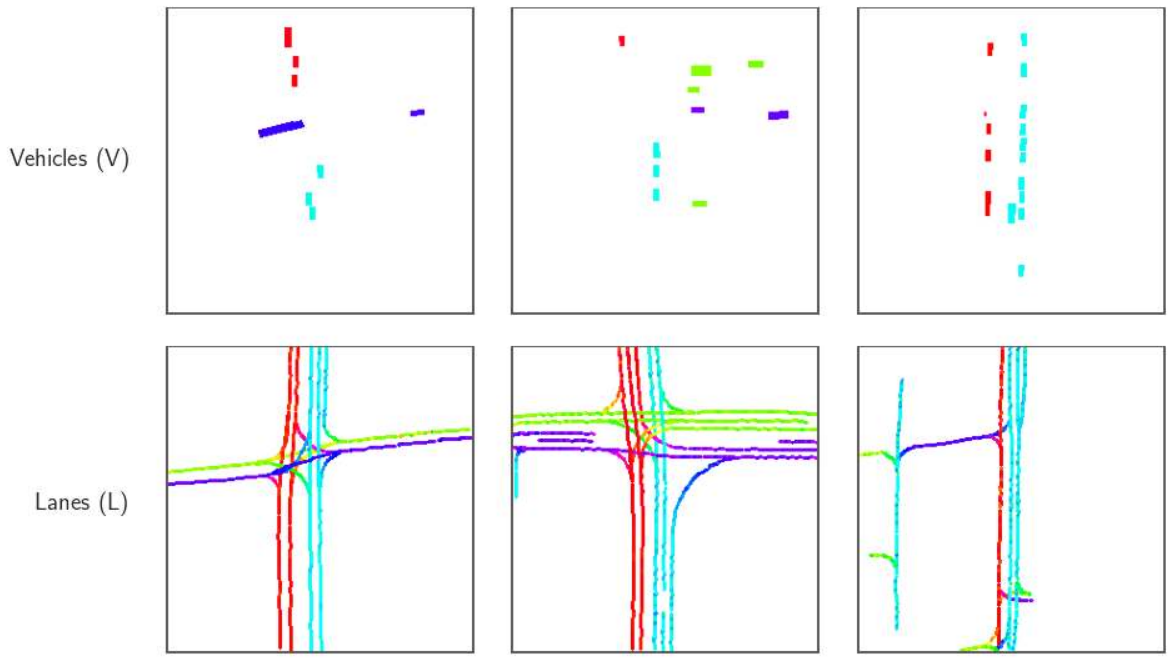


Figure 4.4: **Examples of Rasterized State Image (RSI).**

V is the set of vehicles, and L is the set of lane centerlines. We chose this specific set of entities as we focus on vehicles as they are the primary targets in BEV perception tasks. We found that excluding lane centerlines negatively impacts vehicle localization, as the spatial alignment between lanes and vehicles provides valuable contextual information. To preserve this relationship, we retain lane centerlines in our scene representation. All elements are defined in a single frame, and dynamic attributes such as velocity are removed, since we aim to generate scenarios at a single timestep. As a result, our rasterized state image I is composed of 4 channels—two per entity type (V and L). Examples of RSI are provided in Fig 4.4. During training of RVAE, we randomly drop vehicles in the scene to augment the existing layouts.

To provide 3D information to the downstream image generation model, we extend the bounding box representation of each vehicle to include height information while removing the speed. Each vehicle is represented as an 7D vector:

$$(x, y, z, \theta, h, w, l), \tag{4.4}$$

where (x, y, z) is the 3D center position of the vehicle, θ is the heading angle, (h, w, l)

are the height, width and length.

In addition, for each vehicle V, we replace the binary prediction with a multi-class classification head to predict the semantic category of the vehicle. Specifically, we classify each detected vehicle into one of eight predefined classes:

{`bicycle, bus, car, construction, emergency, motorcycle, trailer, truck`}

which are the category labels provided in the nuScenes dataset. This additional signal allows the multi-view generator to condition appearance on semantic type, further enhancing realism and controllability.

Lastly, we adopt the RLM diffusion model ($\mathcal{E}_{\text{RLM},\theta}$) as our layout generator. This model operates in the latent space learned by the RVAE and enables the sampling of diverse scene configurations that follow the training distribution. To control the semantic properties of the generated layouts, particularly the number of vehicles, we incorporate a simple guidance mechanism into the sampling process. This allows us to bias the generation toward scenes that meet specific criteria. Further details are provided in Section 4.1.6.

4.1.4 MagicDrive

Given driving scene $\mathcal{S}$, MagicDrive [Gao et al., 2024a] generates multi-view camera images $\mathcal{I} \in \mathbb{R}^{N\times C\times H\times W}$, where N is the number of camera views, C is the number of channels, and $H \times W$ is the resolution of each image. They parametrize the driving scene $\mathcal{S}$. It consists of three components: a semantic map $\mathbf{M}$, a set of 3D bounding boxes $\mathbf{B}$, and a language description $\mathbf{L}$. The map $\mathbf{M} \in \{0,1\}^{w\times h\times c}$ encodes the bird's-eye view (BEV) representation of the scene over a $w \times h$ meter grid with c semantic channels (e.g., drivable area, lanes, vehicles). The bounding boxes $\mathbf{B} = \{(c_i, b_i)\}_{i=1}^{N}$ represent N object instances, where each box is described by a class label $c_i \in \mathcal{C}$ and its 3D geometry $b_i \in \mathbb{R}^{8\times 3}$ as the coordinates of its eight corners in (x, y, z) space. The scene is further contextualized by a text prompt $\mathbf{L}$, describing high-level attributes such as location or weather. To generate images from this structured input, MagicDrive takes the camera pose $\mathbf{P} = [\mathbf{K}, \mathbf{R}, \mathbf{T}]$ as additional input, where $\mathbf{K}$ is the intrinsic matrix and $[\mathbf{R} \mid \mathbf{T}]$ defines the extrinsic

matrix. Conditioned on scene and camera pose $(\mathcal{S}, \mathbf{P})$, the model $\mathcal{G}$ generates a realistic street-view image $I \in \mathbb{R}^{H \times W \times 3}$ using a latent diffusion process guided by cross-attention and additive encoders.

Specifically, MagicDrive conditions its generative model on a combination of scene-level, object-level, and map-level features, each derived from different types of inputs. Discrete symbolic inputs such as the text prompt $\mathbf{L}$ and object class labels c_i are encoded using a pretrained CLIP encoder E_{text}, while continuous inputs such as camera parameters and 3D geometry are embedded using Fourier positional encoding and lightweight multilayer perceptrons (MLPs). 2D BEV map is encoded through a CNN-based additive encoder.

The *scene-level features* include the text prompt and the camera pose. The prompt $\mathbf{L}$, which describes high-level scene attributes such as "sunny downtown", is mapped to an embedding $\mathbf{h}^{\text{text}} = E_{\text{text}}(\mathbf{L})$. The camera pose $\mathbf{P} = [\mathbf{K}, \mathbf{R}, \mathbf{T}]$, consisting of intrinsics and extrinsics, is first Fourier-encoded and then passed through an MLP to produce $\mathbf{h}^{\text{cam}} = E_{\text{cam}}(\text{Fourier}([\mathbf{K}, \mathbf{R}, \mathbf{T}]))$.

More importantly, *object-level conditioning* incorporates both semantic and geometric details of the 3D bounding boxes in the scene. Objects are described by class labels $c \in \mathcal{C}$ and its 3D box corner coordinates $b \in \mathbb{R}^{8 \times 3}$. These are encoded as follows:

- Class labels c are embedded using CLIP: $e_c^b = E_{\text{text}}(L_c)$.
- Geometry b is first encoded using a Fourier-based positional encoder, followed by an MLP: $e_p^b = E_{\text{box}}(\text{Fourier}(b))$..
- The resulting object embedding is the concatenation:

$$\mathbf{h}^{\text{box}} = [e_c^b; e_p^b].$$

To ensure that only relevant objects influence the generation for each camera, a visibility mask f_{viz} is applied to filter out objects outside the camera views. Only the visible objects are used as conditioning inputs during generation.

The road map $\mathbf{M} \in \{0,1\}^{w \times h \times c}$ is represented as a 2D grid in the BEV frame and encoded using an additive CNN encoder to produce a map embedding $\mathbf{h}^{\text{map}} = E_{\text{map}}(\mathbf{M})$. Since perspective cues are already provided through 3D object bounding boxes and camera poses, MagicDrive avoids explicit view transformation for the map.

For conditioning, MagicDrive uses ControlNet. The text embedding $\mathbf{h}^{\text{text}}$, camera pose embedding $\mathbf{h}^{\text{cam}}$, and box embedding $\mathbf{h}^{\text{box}}$ are concatenated as $[\mathbf{h}^{\text{text}}; \mathbf{h}^{\text{cam}}; \mathbf{h}^{\text{box}}]$ and passed as a tokenized prompt to the main UNet. The map embedding $\mathbf{h}^{\text{map}}$ passed as a condition to ControlNet. These branches interact through cross-attention, enabling geometrically aligned image synthesis.

To ensure consistency across multiple camera views, MagicDrive adds a *cross-view attention module* on top of the cross-attention modules in the UNet architecture of Stable Diffusion, initialized with zero weights for stable training. See Conditioning for details of the conditioning mechanism via cross-attention. The *cross-view attention module* enables each target camera view t to attend to its adjacent neighboring views, namely the *left* l and *right* r views. For each neighboring view $i \in \{l, r\}$, attention is computed between the target view's queries Q_t and the neighboring view's keys K_i and values V_i:

$$\text{Attention}_{cv}^{i} = \text{softmax}\left(\frac{Q_t K_i^T}{\sqrt{d}}\right) \cdot V_i, \quad i \in \{l, r\} \tag{4.5}$$

$$h_t^{\text{out}} = h_t^{\text{in}} + \text{Attention}_{cv}^{l} + \text{Attention}_{cv}^{r} \tag{4.6}$$

Here, h_t^{in} denotes the output of the cross-attention module for view t at a specific UNet block. The updated feature h_t^{out} is obtained by adding attention outputs from both neighbors to h_t^{in} via a residual connection, allowing information to propagate across views while maintaining spatial alignment.

Mapping SLEDGE outputs to MagicDrive inputs. To generate a coherent scene configuration $\mathcal{S} = \{\mathbf{M}, \mathbf{B}, \mathbf{L}, \mathbf{P}\}$ for MagicDrive, we extract compatible elements from RVAE_{dec} outputs from SLEDGE.

The bounding box set $\mathbf{B}$, represented by the eight corners of each object in 3D, is directly derived from the generated vehicle set $\hat{V}$. Each vehicle is parameterized by

a 7D vector $(x, y, z, \theta, w, l, h)$, encoding its center position, heading, and dimensions. From this, we compute the 8 corners of the 3D bounding box via:

$$\mathbf{B} = \texttt{Corners3D}(x, y, z, \theta, w, l, h),$$

where `Corners3D` computes the 8 vertices based on center positions and dimension extents.

For the map component $\mathbf{M}$, we use only the rasterized form of the generated lane centerlines, denoted as $\text{RSI}_{\hat{\text{L}}}$. Generating all 8 semantic map classes supported by MagicDrive would increase the complexity, so we simplify it.

The text prompt $\mathbf{L}$ and camera pose $\mathbf{P}$ are retrieved based on the city label associated with the generated latent $\mathbf{z}_0 \sim \mathcal{E}_{\text{RLM},\theta}(\mathbf{c})$. Given the city condition $\mathbf{c}$, we sample a matching text description and camera configuration from the existing dataset subset corresponding to that city, ensuring compatibility between the scene layout and viewpoint.

4.1.5 Latent Generation with Diffusion Guidance

The core contribution of our work is the integration of diffusion guidance into the latent generation process to control a specific scene attribute that is underrepresented in the dataset. In other words, driving scenes containing 16 to 21 vehicles, as discussed in Section 4.1.2. To this end, we build on a pretrained latent diffusion model $\mathcal{E}_{\text{RLM},\theta}$ and incorporate a Loss-Based Guidance strategy into its sampling procedure.

The full sampling process is described in Algorithm 1. We follow a DDIM-based reverse diffusion process, which estimates the clean latent $\hat{\mathbf{x}}_0$ at each timestep from the noisy latent $\mathbf{z}_t$. This latent is decoded using RVAE_{dec} into vehicles $(\hat{V})$, and lanes $(\hat{L})$. We then compute a differentiable vehicle count loss based on $\hat{V}$, and add its gradient with respect to $\mathbf{z}_t$ directly to the predicted noise $\boldsymbol{\epsilon}$. We found that scaling the gradient led to worse results: if too small, the signal was ineffective; if too large, it pushed the model off-distribution. For this reason, we add the gradient without scaling.

The loss, defined in Code 4.1, acts as a direct signal to guide the generation toward the desired number of vehicles. We also retain classifier-free guidance (CFG)

to control for high-level attributes such as city labels, following the original design in SLEDGE. Overall, our approach integrates loss-based feedback into the generation loop in a simple and effective way, enabling control over structured aspects of the scene during generation.

Code Snippet 4.1: Vehicle count loss

```
def vehicle_count_loss(vehicle_probs, target_count):
    probs = torch.sigmoid(vehicle_mask)
    actual_count = probs.sum(dim=1)
    loss = torch.mean((actual_count - target_count) ** 2)
    return loss
```

Algorithm 1 Guided Diffusion Sampling with Decoder Supervision

Require: Pre-trained atent denoiser $\mathcal{E}_{\text{RLM},\theta}$, pre-trained decoder RVAE_{dec}, scheduler $\mathcal{N}$

Require: City label y, null label $\varnothing$, loss function $\mathcal{L}$

Require: Guidance scale w, timesteps T, target vehicle count k

$\texttt{labels} \leftarrow [y, \varnothing]$ `# conditional and null CFG labels`

$\mathbf{z}_0 \sim \mathcal{N}(0, I)$ `# sample initial latent`

$\mathcal{N}.\texttt{set\_timesteps}(T)$ `# set diffusion steps`

for each $t \in \{T, \ldots, 1\}$ **do**

  $\mathbf{z}_t \leftarrow \mathcal{N}.\texttt{scale\_input}(\mathbf{z}_t, t)$ `# scale latent for step`

  $\boldsymbol{\epsilon} \leftarrow \mathcal{E}_\theta(\mathbf{z}_t, \texttt{labels}, t)$ `# noise prediction`

  $\hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{z}_t - \sqrt{1-\bar{\alpha}_t} \cdot \boldsymbol{\epsilon}\right)$ `# predict clean latent`

  $\mathbf{v}, \mathbf{l}, \leftarrow \text{RVAE}_{\text{dec}}(\hat{\mathbf{x}}_0)$ `# decode scene into vehicles & lanes`

  $\mathcal{L} \leftarrow \texttt{vehicle\_count\_loss}(\mathbf{v}, k)$ `# count loss`

  $\boldsymbol{\epsilon}_{\text{guided}} \leftarrow \boldsymbol{\epsilon} + \nabla_{\mathbf{z}_t} \mathcal{L}$ `# apply loss-based guidance`

  $\boldsymbol{\epsilon}_{\text{guided}} \leftarrow \texttt{CFG}(\boldsymbol{\epsilon}_{\text{guided}}, w)$ `# apply classifier-free guidance`

  $\mathbf{z}_{t-1} \leftarrow \mathcal{N}.\texttt{step}(\boldsymbol{\epsilon}, t, \mathbf{z}_t)$ `# update latent`

end for

return $\text{RVAE}_{\text{dec}}(\mathbf{z}_{t=0})$ `# decode final latent`

4.1.6 Bird's Eye view Model Training

To demonstrate that our generated dataset can be integrated into any BEV perception model, we conduct a supervised training setup using the synthetic data. Given the generated multi-view camera images $\mathbf{I} \in \mathbb{R}^{N \times C \times H \times W}$—where N is the number of views, and $C \times H \times W$ are the image channels and resolution—and the corresponding generated layouts converted to rasterized form RSI $\in \mathbb{R}^{X \times Y}$, where $X \times Y$ denotes the spatial resolution of the BEV layout, we train a BEV perception model by using the camera images as input and the rasterized layouts as supervision targets. This setup validates that the generated data is compatible with existing BEV perception pipelines and can be used for downstream tasks.

4.2 Experiments

4.2.1 Experimental Setup

Dataset

We utilized nuScenes [Caesar et al., 2020] for our experiments. See Section 3.2.1 for the dataset details.

Evaluation Metrics

RVAE Performance To evaluate the performance of our Raster-to-Vector Autoencoder, we report the mean Intersection over Union *(mIoU)* between the original rasterized input and the rasterized version of the decoded latents. While mIoU is a strict metric—where even small pixel-level deviations can lead to a significant drop in score—it remains a strong indicator of how well the spatial extents of objects are preserved through the latent representation and reconstruction process. Importantly, our primary task is bird's-eye view (BEV) segmentation. Therefore, our evaluation is aligned with the goal of assessing how well the reconstructed layout matches the original in the BEV space.

Guidance Effectiveness To evaluate the effectiveness of the guidance for generating the layout, we compute the *absolute mean error* between the target and the number of vehicles generated. While this is a strict metric that penalizes all values other than the target count, it does not capture the directionality of the error. In practice, the magnitude and direction of deviation can be informative. For example, if an unconditioned generation yields 2 vehicles and the guided generation produces 10 vehicles toward a target of 16, the result though not exact reflects a meaningful shift in the intended direction.

To better capture such cases, we also consider a softer criterion, call it *Guidance Direction Score (GDS)*, which measures the relative improvement over unconditional generation. It is defined as:

$$r = \frac{|c_{\text{normal}} - c_{\text{target}}| - |c_{\text{guided}} - c_{\text{target}}|}{|c_{\text{normal}} - c_{\text{target}}| + 1e\text{-}6} \tag{4.7}$$

where c_{normal} is the number of vehicles in the generation without guidance, c_{guided} is the count in the guided sample and c_{target} is the target vehicle count.

To obtain a bounded score, we apply a sigmoid transformation to the result:

$$\text{GDS} = \frac{1}{1 + e^{-k \cdot r}} \tag{4.8}$$

where k is a scaling factor. This ensures that GDS remains in the range $(0, 1)$, with values above 0.5 indicating improvement due to guidance, values below 0.5 indicating degradation, and 0.5 denoting no change.

Image Generation Quality and Control Accuracy Finally, to evaluate the image quality and control accuracy of the generated camera images, we utilize a *pretrained BEV perception model* (CVT [Pan et al., 2020]). This evaluates whether the generated images are consistent with the original dataset distribution that the multiview image generator (MagicDrive) was trained on. Performance close to that on the original validation set indicates that the generated images are distributionally similar. In addition, we incorporate our generated dataset as a training augmentation to assess its effectiveness on the BEV perception task.

Training Details

We trained the RVAE variants on 4 RTX A6000 GPUs using a batch size of 256 and a learning rate of 0.0001 with the AdamW optimizer for 40k steps. We used 50 queries for both vehicles and lanes. RVAE is trained with both a reconstruction loss and a KL loss. In addition to the original formulation, we introduced an angle loss that is aware of the periodicity of angles, ensuring that values such as -2π and 2π are treated as equivalent and are not penalized. This replaces the standard reconstruction loss applied to angle predictions. We assign three times more weight to bounding box reconstruction compared to lane center reconstruction.

The latent diffusion model is trained using the DiT-B backbone on 2× RTX A6000 GPUs. We train our model using AdamW with a learning rate 1e-4. We used batch size of 64 and trained the model for a total of 40,000 steps. We apply classifier-free guidance with a scale of 4.0 over 50 denoising steps, using the DDIM scheduler. We perform guidance on top of this with 50 inference timesteps.

For training the BEV perception model CVT [Pan et al., 2020], we follow the setup described in the original paper and its official codebase. Specifically, we train the model for 50,000 iterations using a batch size of 16 and a learning rate of 4×10^{-3}.

4.2.2 BEV Layout Generation

RVAE

	mIoU (%)	
Model	**Vehicles**	**Lanes**
RVAE 2D	40.0 ± 15.2	24.8 ± 8.2
RVAE 3D	26.0 ± 19.6	20.3 ± 8.1
RVAE 3D+	32.5 ± 16.6	21.9 ± 7.5

Table 4.1: **RVAE mIoU scores on the nuScenes validation set.**

We present per-entity mIoU scores for different RVAE variants (RVAE2D, RVAE3D, and RVAE3D+) on the nuScenes validation set in Table 4.1. All models are trained to reconstruct vehicles (V) and lanes (L), with a higher loss weight assigned to vehicles, as they are our primary focus. RVAE2D is trained using 2D bounding box coordinates, parameterized by a 5D vector and a binary class label. RVAE3D extends this to 3D bounding box coordinates, parameterized by a 7D vector and a binary class label. RVAE3D+ further incorporates a multiclass classification objective for vehicles. For further details, refer to Section 4.1.3.

We report mIoUs based on the following procedure: since RVAE is not a deterministic model due to its KL loss, we perform the forward pass three times per sample and average the resulting IoUs to obtain a stable per-sample estimate. The final mean IoU is computed by averaging these per-sample IoUs across all validation samples. In addition to the mean, we also report the standard deviation across samples to capture the variability in reconstruction quality.

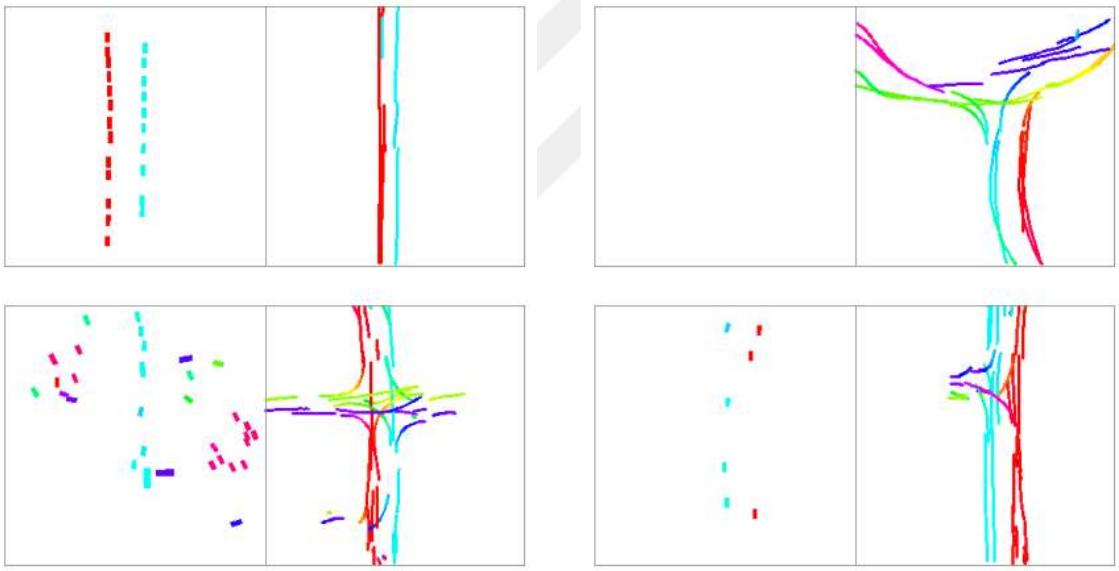
Unsurprisingly, all models learn to reconstruct vehicles more accurately than lane centers, which aligns with the higher loss weight assigned to vehicles.

The best-performing model for both vehicles and lanes is RVAE2D, achieving a 40.0% mIoU score for vehicles. This is expected; moreover, RVAE2D can be considered an upper bound among these models, as it is evaluated in the same 2D space in which it was trained. Its objective is to predict the pixel locations of vehicles in the rasterized BEV representation, and it involves fewer parameters to learn, which also benefits lane reconstruction.

RVAE3D performs significantly worse than RVAE2D, with a 14.0-point drop in vehicle mIoU—likely due to the added complexity of predicting the z-dimension. RVAE3D+, on the other hand, performs relatively better: it trails RVAE2D by 7.0 points but outperforms RVAE3D by 6.5 points in vehicle mIoU. We observe that incorporating class signals for vehicles improves both vehicle and lane mIoUs compared to RVAE3D. This suggests that reducing the complexity of regressing vehicle extents allows the model to better focus on accurately predicting their centers. Based on this observation, we adopt RVAE3D+ as our default 3D model for the remainder of our experiments.

It is also worth noting that all models exhibit relatively high standard deviations across samples, which we attribute to the stochastic nature of the RVAE, introduced by the KL divergence term.

We also provide two qualitative reconstruction examples from RVAE2D in Fig. 4.9. In the left column, we show the ground truth rasterized scene image (RSI), the middle column shows the reconstruction produced by our model, and the right column overlays the ground truth and reconstruction. Consistent with the quantitative results, the model reconstructs vehicles more accurately than lanes in these examples. For vehicles, the most challenging cases occur when objects are dense, such as adjacent vehicles in crowded scenes. For lanes, the most difficult scenarios are turns and intersections, where geometry is more complex.

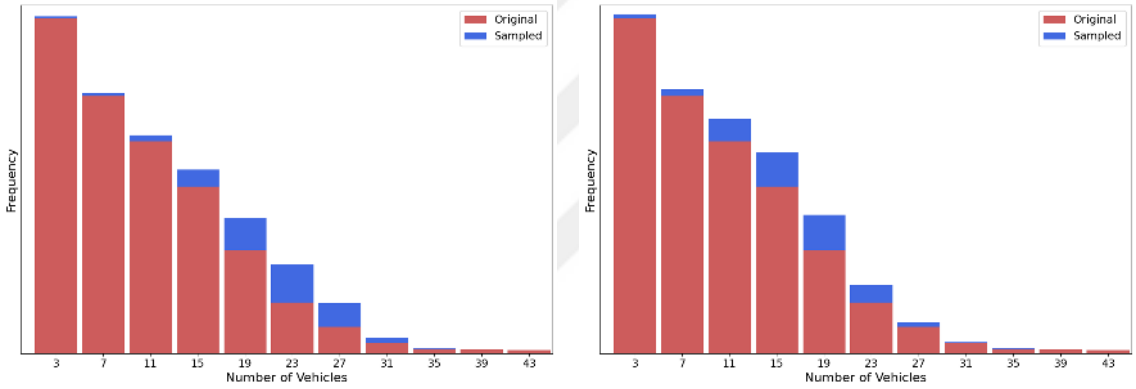


(a) Boston (b) Singapore

Figure 4.5: **Comparison of layout generations (without guidance) in Boston and Singapore scenes.** Generated scenes are visualized by concatenating vehicles on the left and lanes on the right. Figure 4.7b is conditioned on Boston, where the traffic flow is upwards and follows right-hand traffic rules. Figure 4.5b is conditioned on Singapore, where the traffic flows downward and follows left-hand traffic rules.

Latent Diffuser ($\mathcal{E}_{RLM,\theta}$)

We provide qualitative results showing generations with different seeds from the Latent Diffuser ($\mathcal{E}_{\text{RLM},\theta}$) in Fig. 4.5, scenes from two cities, Boston and Singapore. Depending on the seed, the quality of generations varies, with some samples appearing more coherent than others. As observed, the distribution of generated layouts reflects city-specific differences in both density and traffic flow. Boston generations tend to include more vehicles, suggesting higher traffic density, while Singapore generations often contain fewer or even no vehicles.



(a) Actual distribution after guidance for Boston scenes.

(b) Actual distribution after guidance for Singapore scenes.

Figure 4.6: **Comparison of vehicle count distributions after guidance for Boston (left) and Singapore scenes (right).**

4.2.3 Diffusion Guidance

Quantitative Results

In Fig. 4.2b, we present the vehicle count distribution of the dataset after applying guidance. While the blue bars in Fig. 4.2a represent the target distribution we aim to achieve, the blue bars in Fig. 4.2b indicate the actual number of vehicles in

$\mathbf{MAE}_{boston}$	$\mathbf{MAE}_{singapore}$	$\mathbf{MAE}_{all}$	$\mathbf{GDS}_{all}$
3.53	4.85	4.18	0.84

Table 4.2: **Guidance Metrics.**

the generated scenes after applying our loss function (Eq. 4.1) for guidance (using RVAE2D).

We observe that vehicle counts in the 15 and 19 bins are successfully generated; however, counts in the 23 bin are not captured. The guided distribution appears shifted by one bin to the left compared to the target distribution. This shift can be considered an inherent effect of applying guidance over latents that were originally trained on an imbalanced dataset. As a result, the generator tends to favor more frequent, lower-count regions.

This phenomenon becomes even more apparent when we analyze the guided distribution by city, shown in Fig. 4.6. Layouts conditioned on Boston (left column) tend to align more closely with the target distribution and do not exhibit a strong shift toward lower vehicle counts. In contrast, Singapore-conditioned scenes clearly shift toward generating fewer vehicles. This behavior is closely related to city-specific vehicle density: Boston typically features heavier traffic and wider roads, whereas Singapore generally includes fewer vehicles and narrower roads. While increasing the guidance strength could improve alignment with the target counts overall, it also introduces a risk of generating out-of-distribution layouts that may appear less coherent or realistic.

We quantitatively evaluate the effect of guidance in Table 4.2. First, we compute the mean absolute error (MAE) between the target and actual vehicle counts after guidance. On average, the guided generations deviate by 4.18 vehicles, which corresponds to approximately one bin size (each bin size is 4 vehicle counts). This provides a numerical explanation for the observed shift. When analyzed per city, the MAE for Boston is 3.53, significantly closer to the target vehicle counts compared to the MAE of 4.85 for Singapore. This aligns with our earlier observation of

city-dependent distributional shifts.

Additionally, we compute the Guidance Direction Score (GDS), as described in Guidance Effectiveness. The GDS, which approaches its upper bound of 1, indicates that the guidance mechanism successfully leads generations in the correct direction toward the intended target counts even if the exact values are not always achieved.

These results demonstrate that the guidance mechanism is effective. Achieving exact vehicle counts in a generative setting is inherently challenging, especially when aiming to preserve scene coherence and stay within the learned distribution, which tends to favor lower vehicle counts in this case. Nonetheless, our approach consistently moves the generations in the intended direction, validating the effectiveness of the proposed loss formulation.

Qualitative Results

We illustrate the effect of guidance on BEV layouts in Figure 4.7. The upper row shows a generated scene with the Boston configuration, while the bottom row shows a scene with the Singapore layout. In the generations without guidance (left column), we report the number of vehicles in the scene. In the guided generations, we provide the target vehicle count and also report the actual number of vehicles generated. As can be seen, in both cases, the number of vehicles increases toward the target count. In the Boston case, the model successfully reaches the exact number. The differing density characteristics between the two cities can also be observed: the Singapore generation contains fewer vehicles and responds less effectively to the guidance compared to the Boston case. Another observation is that lane generation is of lower quality compared to vehicle generation and tends to become distorted under vehicle count guidance.

4.2.4 Image Generations

Qualitative Results

Using guided layouts generations, we generate six camera images per scene using MagicDrive [Gao et al., 2024a]. Examples of Boston-conditioned scenes are shown in Fig. 4.10, while Singapore-conditioned scenes are shown in Fig. 4.11. In each figure, the top row displays the left, front, and right camera views, while the bottom row shows the back-right, back, and back-left views.

Both Boston and Singapore scenes depict crowded traffic conditions, consistent with our guidance mechanism. In general, MagicDrive struggles to accurately generate far-away vehicles, likely due to limited resolution and visibility constraints in those regions.

Furthermore, in Fig. 4.12, we compare image generations without and with guidance. The BEV layouts are shown on the left, and the corresponding generated scenes are shown on the right. We also visualize vehicle bounding boxes in the camera images, aligned with their locations in the BEV layouts. The effect of guidance is evident in both the Boston and Singapore cases, with a noticeable increase in vehicle density in the guided scenes.

Table 4.3: **BEV Perception Model (CVT) evaluation results on real and synthetic datasets.** We use a pretrained BEV perception model, CVT, trained on the nuScenes training set, and evaluate it on both real and generated datasets to assess the effectiveness of scene-level control and the quality of the generated camera images.

Dataset	**Gen./Real**	**mIoU**$_{\text{target\_bins}}$	**mIoU**
nuScenes valset	Real	31.97	32.20
nuScenes valset	Gen	23.94	24.93
Ours2D	Gen	21.25	21.25
Ours3D+	Gen	22.34	22.34

Table 4.4: **BEV perception model (CVT) performance on nuScenes val with different synthetic data augmentation experiments.** We augment the nuScenes training data using synthetic dataset generated with nuScenes training layouts, as well as dataset generated by our proposed method.

Dataset	**Guided**	**# Synt. Data**	**mIoU$_{\text{target_bins}}$**	**mIoU**
Real	-	-	31.97	32.20
w Trainset Gen.	No	28k	31.31	31.72
w Ours2D	Yes	6k	31.00	31.65
w Ours3D+	Yes	6k	30.74	31.38

Quantitative Results

To evaluate both control effectiveness and generation quality, we perform a quantitative comparison of BEV perception scores across different datasets using a pretrained BEV perception model (Tab. 4.3). For our experiments, we adopt CVT [Pan et al., 2020], which is trained on the nuScenes training set.

CVT uses visibility masks to ignore vehicles that are not visible to the ego-vehicle during evaluation. However, our generated datasets do not contain such visibility masks. To ensure a fair comparison, we evaluate all datasets by considering all vehicles as visible. While this results in lower mIoU scores overall, it provides a consistent evaluation setting across real and generated datasets.

We report two types of mIoU metrics: mIoU$_{\text{target_bins}}$, which considers only the scenes where the vehicle count falls into our target bins, and the overall mIoU$_{\text{target_bins}}$ across the entire dataset.

The datasets evaluated are as follows: the original nuScenes validation set (real), MagicDrive-generated images conditioned on nuScenes validation layouts (gen nuscenes), and our two datasets generated using guided latents: Ours2D from RVAE2D and Ours3D+ from RVAE3D+.

The real nuScenes validation set achieves an mIoU of 32.20 without filtering based on visibility. Its $\text{mIoU}_{\text{target\_bins}}$ is slightly lower at 31.97, indicating that the scenes we target (those within the 14–25 vehicle range) are generally more challenging.

The MagicDrive generated dataset conditioned on nuScenes layouts achieves an mIoU of 24.93 overall and 23.94 for the target bins, showing a noticeable performance drop and highlighting the synthetic to real gap of approximately 8 points.

For our generated datasets, since all samples are from the target bin region, the mIoU and $\text{mIoU}_{\text{target\_bins}}$ values are the same. Ours2D lags behind the gen-nuscenes baseline by 2.69 points, while Ours3D+ lags by only 1.36 points, slightly outperforms the 2D variant, highlighting the benefit of incorporating height prediction and vehicle class signals into the generation process. These results suggest that our generated layouts deviate minimally from the original data distribution, and our generated configurations mostly reflect the characteristics of the real world dataset configurations.

4.2.5 BEV Segmentation Model Training

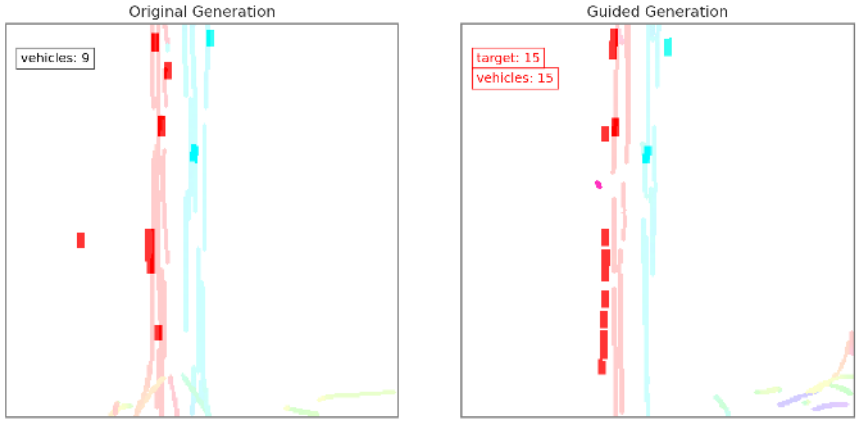
To evaluate the effectiveness of synthetic datasets on the BEV perception task, we augment the real nuScenes dataset with different generated datasets and present the results in Table 4.4. We train the CVT [Pan et al., 2020] model from scratch mixing real and synthetic datasets. The synthetic augmentation datasets we include in our experiments are as follows: (i) data generated by using training set layouts (denoted as Trainset Gen), (ii) our guided generations using RVAE2D latents (denoted as Ours2D) and (iii) using RVAE3D+ latents (denoted as Ours3D+).

Neither the training set generations nor our guided generations lead to an improvement in the generalization performance of CVT compared to using only real data. Interestingly, despite the Trainset Gen dataset being significantly larger (28k scenes vs. 6k in Ours2D, Ours3D+), both methods result in nearly the same performance.

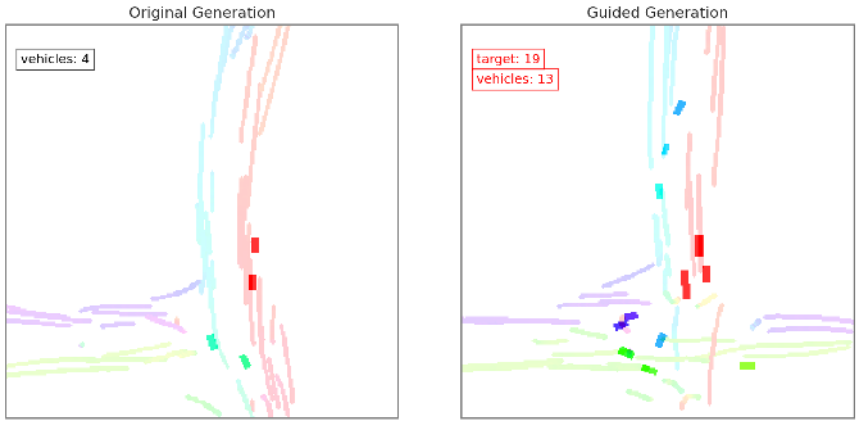
In Fig. 4.8, we further analyze the distribution of IoU scores across different vehicle count bins. The purple bars represent the model trained on the original

nuScenes dataset, while the green bars correspond to the model trained with additional samples from our guided generations from RVAE2D. Although we observe a darker shade of green in certain bins, indicating a higher sample count, this does not translate into improved IoU scores. In fact, there is a slight performance drop in some bins.

Reproducibility Note We were unable to reproduce the results reported in the original MagicDrive paper, which claimed that augmenting with training set generations improves CVT performance from 36.00 to 40.36 mIoU (when visibility masks are included). When we included visibility masks in our evaluation, we obtained a score of 35.64, slightly lower than the reported baseline. Due to this discrepancy, it remains inconclusive whether our proposed guidance method is ineffective or if the performance gains reported by MagicDrive depend on specific experimental conditions or implementation details that were not replicated in our setup.



(a) Boston



(b) Singapore

Figure 4.7: **Comparison of layout generations without (left column) and with guidance (right column) for Boston and Singapore.** Guidance effectively pushes the number of vehicles toward the target count.

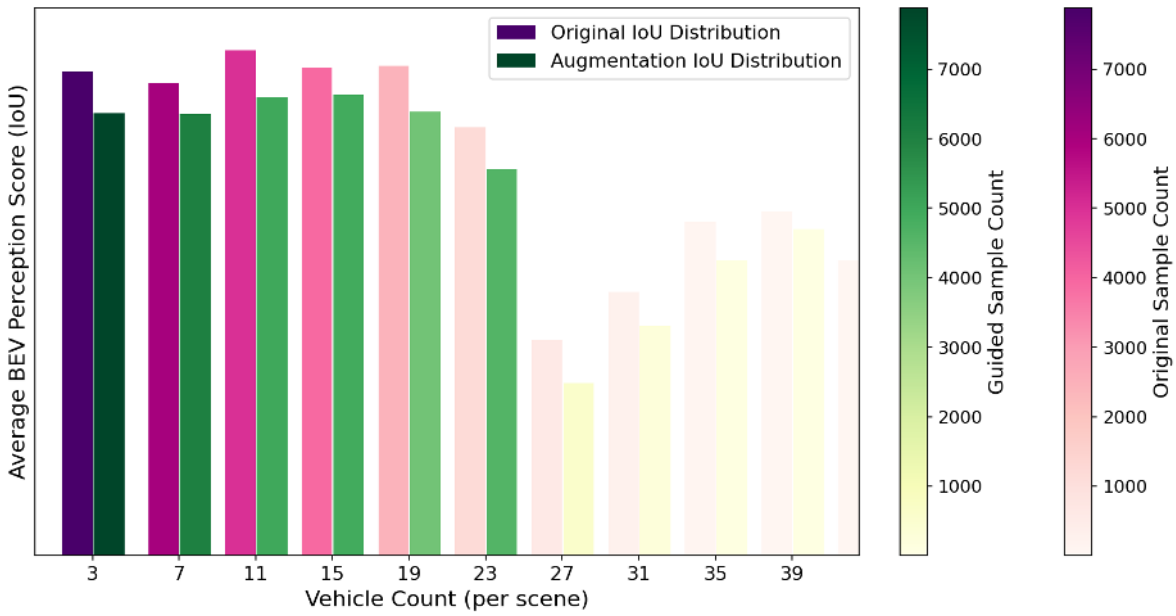


Figure 4.8: **Average IoU Score Distribution by Vehicle Count across Original and Augmented Training Sets.** The purple bars represent scores from a BEV model trained on the original nuScenes training set, while the green bars correspond to training on the original set combined with our guided augmented data.

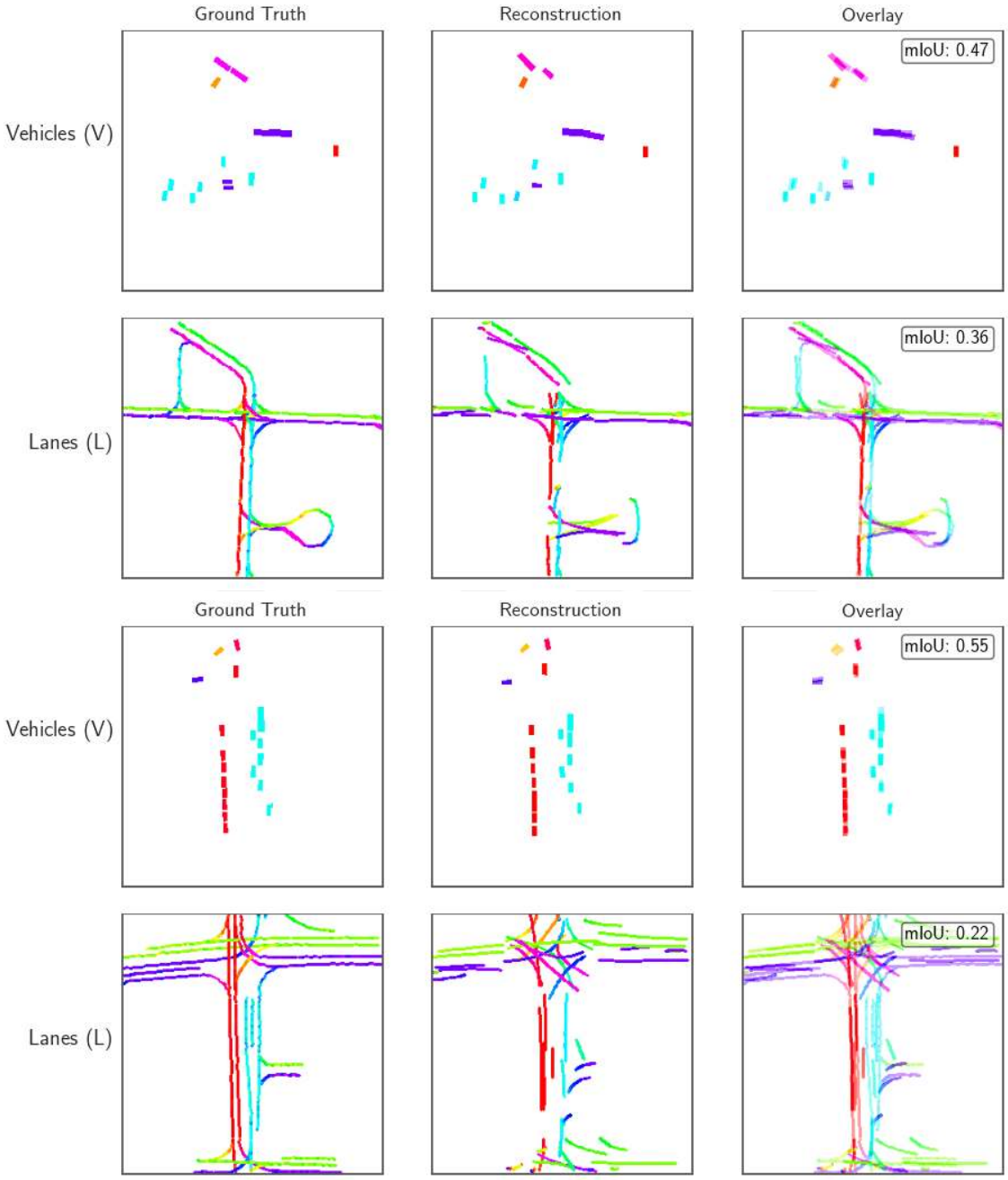


Figure 4.9: **RVAE reconstruction examples from the nuScenes validation set.** Each subfigure shows the ground truth RSI (left), the model reconstruction (middle), and their overlay (right).

Figure 4.10: **Qualitative results from MagicDrive using our guided latents conditioned on Boston scenes.** The generated multi-view camera images exhibit dense vehicle clusters and right-hand traffic flow, reflecting Boston-specific traffic patterns under guided generation.

Figure 4.11: **Qualitative results from MagicDrive using our guided latents conditioned on Singapore scenes.** The generated multi-view camera images exhibit dense vehicle clusters and left-hand traffic flow, reflecting Singapore-specific traffic patterns under guided generation.

(a) Boston

(b) Singapore

Figure 4.12: **Comparison of generated scenes without and with guided BEV layouts.** BEV layouts are shown on the left, and the corresponding generated scenes are shown on the right. As guidance is applied, vehicle density increases accordingly.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Discussion

In this thesis, we addressed two distinct yet related challenges in camera-only BEV perception: (i) the insufficient robustness of current BEV perception models when exposed to corrupted inputs, and (ii) the limited coverage of real-world driving datasets, which existing multi-view image generation pipelines fail to address effectively due to their lack of input condition diversity and controllability.

In Chapter 3, we investigated the effectiveness of using DINOv2 with Low-Rank Adaptation (LoRA) to improve robustness in BEV segmentation on the nuScenes dataset. We first demonstrated that our approach achieves performance comparable to SimpleBEV—a state-of-the-art BEV segmentation method—while using a smaller variant of the DINOv2 backbone in the clean setting. Further scaling of the backbone or increasing the input and feature resolutions led to significant improvements over the baseline. We also compared our adaptation strategy with both frozen and fully fine-tuned versions, and found that our method is considerably more parameter-efficient and converges faster during training.

When evaluated under corruption scenarios using nuScenes-C, our approach showed increased robustness across a wide range of visual corruptions, including extreme weather and lighting conditions. Additionally, we observed that robustness improves further as the DINOv2 backbone is scaled up. These findings support our motivation to leverage general-purpose visual representations from foundation models to enhance BEV segmentation. Importantly, our method achieves these gains while requiring significantly fewer parameter updates than full fine-tuning or supervised baselines, making it both efficient and scalable.

In Chapter4, we proposed a guided layout generation method to create control-

lable input configurations for multi-view image generators. Our goal was to generate moderately challenging scenes—those that are still within the nuScenes dataset distribution and therefore easier to synthesize, yet difficult enough that BEV perception performance begins to decline in this region (e.g., with CVT). After identifying this target region, we first learned the distribution of scene configurations using the method proposed in SLEDGE [Chitta et al., 2025], and then integrated our guidance mechanism to lead generation toward the desired scene properties. While the generated scenes did not perfectly match the intended vehicle counts, we were able to consistently shift the outputs closer to the target range. We also observed that generating crowded scenes was more difficult for Singapore than for Boston, likely due to differences in traffic density and road layout between the two cities.

In conclusion, to quantitatively evaluate the realism of our generated data, we used a pretrained BEV perception model (CVT) as a sanity check. The model's performance on our guided layouts was comparable to that on scenes generated from real-world configurations, suggesting that our guided generations closely reflect real scene configurations. Finally, we assessed the impact of our data on BEV model training by augmenting the nuScenes dataset with our guided scenes. However, we did not observe any performance improvement. Despite this outcome, we argued that it should not be interpreted as a failure of the proposed guidance method to improve BEV perception generalizability, since we encountered reproducibility issues with MagicDrive—since even augmenting with real-layout-based generations failed to improve performance in our setup, as opposed to reported results in the original paper. These inconsistencies suggest that experimental conditions may have affected the observed results.

Nonetheless, our method successfully demonstrated the ability to guide layout generation toward meaningful and underrepresented regions of the data distribution. This capability offers a promising direction for future research on targeted synthetic data generation to improve BEV perception performance.

5.2 Limitations and Future Work

A key limitation of our method proposed in Chapter3 is that the analysis is currently restricted to DINOv2 and SimpleBEV, without exploring the performance of other promising 3D-aware foundation models, such as Stable Diffusion or more recent models trained directly on 3D data like Dust3r [Wang et al., 2024a]. Additionally, other BEV perception models like CVT[Pan et al., 2020] and BEVFormer [Liu et al., 2023b] are not considered, which may limit the generalizability of our findings across different architectures. Comparing a broader range of foundation models for BEV perception is a compelling direction for future research and could provide deeper insights into how various pretrained models can be integrated into BEV frameworks.

The main limitations of our guided generation method proposed in Chapter 4 are as follows. First, we were not able to conclusively determine the effectiveness of our guidance in improving BEV perception generalization. Second, our approach is bounded by the training dataset distribution. Although we did not use extreme guidance strengths or highly uncommon driving scenarios, we still struggled to achieve precise control over the generated outputs. Even when increasing the guidance strength, the generations occasionally became incoherent likely due to the limited diversity and imbalance in the training dataset. Third, our current pipeline is designed around a specific multi-view image generator and requires engineering effort to adapt to other generators.

Despite these limitations, our method successfully demonstrated the ability to guide layout generation toward meaningful and underrepresented regions of the data distribution. As future work, integrating alternative, preferably higher-resolution, layout-conditioned multi-view image generators into our pipeline may offer more conclusive and improved results in BEV perception tasks. Since our guidance method is flexible, it can be reused across different systems for scalable, targeted dataset generation. Additionally, because layouts can be converted to 3D bounding boxes, the targeted generation pipeline could also be evaluated on 3D object detection or other autonomous driving downstream tasks to explore its potential for improving generalizability beyond BEV segmentation.

BIBLIOGRAPHY

[Aydemir et al., 2023] Aydemir, G., Xie, W., and Guney, F. (2023). Self-supervised object-centric learning for videos.

[Aydemir et al., 2024] Aydemir, G., Xie, W., and Güney, F. (2024). Can visual foundation models achieve long-term point tracking? *arXiv preprint arXiv:2408.13575*.

[Bansal et al., 2023] Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., and Goldstein, T. (2023). Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852.

[Barın et al., 2024] Barın, M. R., Aydemir, G., and Güney, F. (2024). Robust bird's eye view segmentation by adapting dinov2. *arXiv preprint arXiv:2409.10228*.

[Bartoccioni et al., 2023] Bartoccioni, F., Zablocki, É., Bursuc, A., Pérez, P., Cord, M., and Alahari, K. (2023). Lara: Latents and rays for multi-camera bird's-eye-view semantic segmentation.

[Blumenkamp et al., 2024] Blumenkamp, J., Morad, S., Gielis, J., and Prorok, A. (2024). Covis-net: A cooperative visual spatial foundation model for multi-robot applications.

[Caesar et al., 2020] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving.

[Caesar et al., 2021] Caesar, H., Kabzan, J., Tan, K. S., Fong, W. K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., and Omari, S. (2021). nuplan: A closed-

loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*.

[Chambon et al., 2024] Chambon, L., Zablocki, E., Chen, M., Bartoccioni, F., Pérez, P., and Cord, M. (2024). Pointbev: A sparse approach for bev predictions.

[Chang et al., 2019] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). Argoverse: 3d tracking and forecasting with rich maps.

[Chang et al., 2024] Chang, W.-J., Pittaluga, F., Tomizuka, M., Zhan, W., and Chandraker, M. (2024). Safe-sim: Safety-critical closed-loop traffic simulation with diffusion-controllable adversaries. In *European Conference on Computer Vision*, pages 242–258. Springer.

[Chen and Krähenbühl, 2022] Chen, D. and Krähenbühl, P. (2022). Learning from all vehicles.

[Chen et al., 2019] Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2019). Learning by cheating.

[Chen et al., 2023] Chen, Q., Wu, Z., Liu, L., Hu, X., Kalantidis, Y., Wang, Z., Darrell, T., and Zhu, J.-Y. (2023). Layoutdm: Discrete diffusion model for layout-to-image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14759–14769.

[Chitta et al., 2025] Chitta, K., Dauner, D., and Geiger, A. (2025). Sledge: Synthesizing driving environments with generative models and rule-based traffic. In *European Conference on Computer Vision*, pages 57–74. Springer.

[Dhariwal and Nichol, 2021] Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

[El Banani et al., 2024] El Banani, M., Raj, A., Maninis, K.-K., Kar, A., Li, Y., Rubinstein, M., Sun, D., Guibas, L., Johnson, J., and Jampani, V. (2024). Probing the 3d awareness of visual foundation models.

[Feng et al., 2023] Feng, L., Li, Q., Peng, Z., Tan, S., and Zhou, B. (2023). Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 3567–3575. IEEE.

[Gao et al., 2024a] Gao, R., Chen, K., Xie, E., Hong, L., Li, Z., Yeung, D.-Y., and Xu, Q. (2024a). MagicDrive: Street view generation with diverse 3d geometry control. In *International Conference on Learning Representations*.

[Gao et al., 2024b] Gao, S., Yang, J., Chen, L., Chitta, K., Qiu, Y., Geiger, A., Zhang, J., and Li, H. (2024b). Vista: A generalizable driving world model with high fidelity and versatile controllability. *arXiv preprint arXiv:2405.17398*.

[Hanselmann et al., 2022] Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., and Geiger, A. (2022). KING: generating safety-critical driving scenarios for robust imitation via kinematics gradients.

[Harley et al., 2023] Harley, A. W., Fang, Z., Li, J., Ambrus, R., and Fragkiadaki, K. (2023). Simple-bev: What really matters for multi-sensor bev perception?

[Ho et al., 2020] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

[Hu et al., 2023a] Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall,

A., Shotton, J., and Corrado, G. (2023a). Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*.

[Hu et al., 2022] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models.

[Hu et al., 2023b] Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., Lu, L., Jia, X., Liu, Q., Dai, J., Qiao, Y., and Li, H. (2023b). Planning-oriented autonomous driving.

[Huang et al., 2024] Huang, B., Wen, Y., Zhao, Y., Hu, Y., Liu, Y., Jia, F., Mao, W., Wang, T., Zhang, C., Chen, C. W., et al. (2024). Subjectdrive: Scaling generative data in autonomous driving via subject control. *arXiv preprint arXiv:2403.19438*.

[Jiang et al., 2023] Jiang, C., Cornman, A., Park, C., Sapp, B., Zhou, Y., Anguelov, D., et al. (2023). Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9644–9653.

[Karaev et al., 2024] Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., and Rupprecht, C. (2024). CoTracker: It is better to track together.

[Kim et al., 2021] Kim, S. W., Philion, J., Torralba, A., and Fidler, S. (2021). Drivegan: Towards a controllable high-quality neural simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5820–5829.

[Lee et al., 2023] Lee, J., Kim, W., Kang, B., Kim, M., and Park, J. (2023). Diverse layout-to-image generation via conditional denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18788–18797.

[Li et al., 2022a] Li, Q., Wang, Y., Wang, Y., and Zhao, H. (2022a). Hdmapnet: An online hd map construction and evaluation framework.

[Li et al., 2024] Li, X., Zhang, Y., and Ye, X. (2024). Drivingdiffusion: Layout-guided multi-view driving scenarios video generation with latent diffusion model. In *European Conference on Computer Vision*, pages 469–485.

[Li et al., 2023] Li, Y., Ge, Z., Yu, G., Yang, J., Wang, Z., Shi, Y., Sun, J., and Li, Z. (2023). Bevdepth: Acquisition of reliable depth for multi-view 3d object detection.

[Li et al., 2022b] Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., and Dai, J. (2022b). Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers.

[Liao et al., 2023] Liao, B., Chen, S., Wang, X., Cheng, T., Zhang, Q., Liu, W., and Huang, C. (2023). MapTR: Structured modeling and learning for online vectorized hd map construction.

[Liu et al., 2023a] Liu, Y., Yuan, T., Wang, Y., Wang, Y., and Zhao, H. (2023a). Vectormapnet: End-to-end vectorized hd map learning.

[Liu et al., 2023b] Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D., and Han, S. (2023b). Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[Lu et al., 2024a] Lu, J., Huang, Z., Yang, Z., Zhang, J., and Zhang, L. (2024a). Wovogen: World volume-aware diffusion for controllable multi-camera driving scene generation. In *European Conference on Computer Vision*, pages 329–345. Springer.

[Lu et al., 2024b] Lu, J., Wong, K., Zhang, C., Suo, S., and Urtasun, R. (2024b). Scenecontrol: Diffusion for controllable traffic scene generation. In *2024 IEEE In-*

ternational Conference on Robotics and Automation (ICRA), pages 16908–16914. IEEE.

[Ma et al., 2024] Ma, E., Zhou, L., Tang, T., Zhang, Z., Han, D., Jiang, J., Zhan, K., Jia, P., Lang, X., Sun, H., et al. (2024). Unleashing generalization of end-to-end autonomous driving with controllable long video generation. *arXiv preprint arXiv:2406.01349*.

[Nguyen et al., 2023] Nguyen, V. N., Groueix, T., Ponimatkin, G., Lepetit, V., and Hodan, T. (2023). Cnos: A strong baseline for cad-based novel object segmentation.

[Oquab et al., 2023] Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.-Y., Xu, H., Sharma, V., Li, S.-W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. (2023). DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

[Pan et al., 2020] Pan, B., Sun, J., Leung, H. Y. T., Andonian, A., and Zhou, B. (2020). Cross-view semantic segmentation for sensing surroundings. *IEEE Robotics and Automation Letters*, 5(3):4867–4873.

[Philion and Fidler, 2020] Philion, J. and Fidler, S. (2020). Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d.

[Pronovost et al., 2023] Pronovost, E., Ganesina, M. R., Hendy, N., Wang, Z., Morales, A., Wang, K., and Roy, N. (2023). Scenario diffusion: Controllable driving scenario generation with diffusion. *Advances in Neural Information Processing Systems*, 36:68873–68894.

[Ramesh et al., 2022] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

[Roddick et al., 2018] Roddick, T., Kendall, A., and Cipolla, R. (2018). Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*.

[Rombach et al., 2022] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

[Rowe et al., 2025] Rowe, L., Girgis, R., Gosselin, A., Paull, L., Pal, C., and Heide, F. (2025). Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17207–17218.

[Russell et al., 2025] Russell, L., Hu, A., Bertoni, L., Fedoseev, G., Shotton, J., Arani, E., and Corrado, G. (2025). Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*.

[Saharia et al., 2022] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494.

[Schramm et al., 2024] Schramm, J., Vödisch, N., Petek, K., Kiran, R. B., Yogamani, S., Burgard, W., and Valada, A. (2024). Bevcar: Camera-radar fusion for bev map and object segmentation.

[Sirko-Galouchenko et al., 2024] Sirko-Galouchenko, S., Boulch, A., Gidaris, S., Bursuc, A., Vobecky, A., Pérez, P., and Marlet, R. (2024). Occfeat: Self-supervised occupancy feature prediction for pretraining bev segmentation networks.

[Sun et al., 2020] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Sun et al., 2024] Sun, S., Gu, Z., Sun, T., Sun, J., Yuan, C., Han, Y., Li, D., and Ang, M. H. (2024). Drivescenegen: Generating diverse and realistic driving scenarios from scratch. *IEEE Robotics and Automation Letters*.

[Swerdlow et al., 2024] Swerdlow, A., Xu, R., and Zhou, B. (2024). Street-view image generation from a bird's-eye view layout. *IEEE Robotics and Automation Letters*.

[Wang et al., 2024a] Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., and Revaud, J. (2024a). Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709.

[Wang et al., 2024b] Wang, X., Zhu, Z., Huang, G., Chen, X., Zhu, J., and Lu, J. (2024b). Drivedreamer: Towards real-world-drive world models for autonomous driving. In *European Conference on Computer Vision*, pages 55–72. Springer.

[Wen et al., 2024] Wen, Y., Zhao, Y., Liu, Y., Jia, F., Wang, Y., Luo, C., Zhang, C., Wang, T., Sun, X., and Zhang, X. (2024). Panacea: Panoramic and controllable video generation for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6902–6912.

[Wilson et al., 2023] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., et al. (2023). Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*.

[Xie et al., 2023] Xie, S., Kong, L., Zhang, W., Ren, J., Pan, L., Chen, K., and Liu, Z. (2023). Robobev: Towards robust bird's eye view perception under corruptions. *arXiv preprint arXiv:2304.06719*.

[Xu et al., 2024] Xu, Y., Chambon, L., Zablocki, É., Chen, M., Alahi, A., Cord, M., and Pérez, P. (2024). Towards motion forecasting with real-world perception inputs: Are end-to-end approaches competitive?

[Yang et al., 2023] Yang, K., Ma, E., Peng, J., Guo, Q., Lin, D., and Yu, K. (2023). Bevcontrol: Accurately controlling street-view elements with multi-perspective consistency via bev sketch layout. *arXiv preprint arXiv:2308.01661*.

[Yuan et al., 2024] Yuan, T., Liu, Y., Wang, Y., Wang, Y., and Zhao, H. (2024). Streammapnet: Streaming mapping network for vectorized online hd map construction.

[Zhan et al., 2023] Zhan, G., Zheng, C., Xie, W., and Zisserman, A. (2023). What does stable diffusion know about the 3D scene? *arXiv preprint arXiv:2310.06836*.

[Zhang et al., 2023] Zhang, J., Herrmann, C., Hur, J., Cabrera, L. P., Jampani, V., Sun, D., and Yang, M.-H. (2023). A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence.

[Zhang et al., 2021] Zhang, Z., Liniger, A., Dai, D., Yu, F., and Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach.

[Zheng et al., 2023] Zheng, G., Zhou, X., Li, X., Qi, Z., Shan, Y., and Li, X. (2023). Layoutdiffusion: Controllable diffusion model for layout-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22490–22499.

[Zheng et al., 2024] Zheng, W., Song, R., Guo, X., Zhang, C., and Chen, L. (2024). Genad: Generative end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 87–104. Springer.

[Zhong et al., 2023a] Zhong, Z., Rempe, D., Chen, Y., Ivanovic, B., Cao, Y., Xu, D., Pavone, M., and Ray, B. (2023a). Language-guided traffic simulation via scene-level diffusion. In *Conference on Robot Learning*, pages 144–177. PMLR.

[Zhong et al., 2023b] Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., and Pavone, M. (2023b). Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 3560–3566. IEEE.

[Zhou and Krähenbühl, 2022] Zhou, B. and Krähenbühl, P. (2022). Cross-view transformers for real-time map-view semantic segmentation.

[Zhu et al., 2023] Zhu, Y., Shen, Z., Zhao, Z., Wang, S., Wang, X., Zhao, X., Shen, D., and Wang, Q. (2023). MeLo: Low-rank adaptation is better than fine-tuning for medical image diagnosis. *arXiv preprint arXiv:2311.08236*.