CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Diffusion Models for Novel View Synthesis in Autonomous Driving

Master's thesis in Complex Adaptive Systems

ARTUR GASPARYAN
RUIQI QIU

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Diffusion Models for Novel View Synthesis in Autonomous Driving

Artur Gasparyan
Ruiqi Qiu

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Diffusion Models for Novel View Synthesis in Autonomous Driving
ARTUR GASPARYAN
RUIQI QIU

Academic Supervisor:
Lennart Svensson, Chalmers University of Technology

Industrial Supervisors:
Adam Tonderski, Zenseact
Carl Lindström, Zenseact
Georg Hess, Zenseact

Examiner:
Lennart Svensson, Chalmers University of Technology

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Diffusion Models for Novel View Synthesis in Autonomous Driving
Artur Gasparyan
Ruiqi Qiu
Department of Electrical Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

Novel View Synthesis (NVS) generates target images from new camera poses using source images and their corresponding poses. It has gained prominence in the field of autonomous driving (AD) as a tool for generating synthetic data to improve perception systems. Current NVS implementations, such as Neural Radiance Fields (NeRFs), excel at constructing 3D scenes from sensory inputs but struggle to accurately render sparsely observed or unseen views. This thesis addresses these limitations by integrating Diffusion Models (DMs) into the NVS pipeline to enhance reconstruction quality in such cases.

We propose a pipeline inspired by ReconFusion, training NeuRAD, a NeRF-based NVS method designed for dynamic AD data, on additional poses not present in the original training set. A pretrained, open-sourced DM, Stable Diffusion, provides supervision by refining NeuRAD's outputs for these unseen views. To improve the DM's performance on AD scenes, we finetune it using Low-Rank Adaptation (LoRA), enabling efficient adaptation to small datasets. ControlNet is incorporated to extend the diffusion model with additional conditioning signals, ensuring better alignment with scene-specific characteristics.

Despite introducing these enhancements, our experiments reveal mixed results. While some metrics show improvement, others remain inconsistent, particularly in challenging scenarios. We identify weak conditional signals and limited LoRA rank as potential limitations. Future research should explore incorporating more robust conditioning signals, such as depth or temporal information, and training on diverse scenes to improve generalization and stability. These directions offer promising avenues for advancing NVS in AD applications.

Keywords: Scene Reconstruction, Novel View Synthesis, Neural Radiance Fields, Autonomous Driving, Deep Learning, Generative Models, Diffusion Models, Latent Diffusion Models, Closed Loop Simulation.

Acknowledgements

We would like to thank our industrial supervisors Adam Tonderski, Carl Lindström Georg Hess for their support and guidance throughout the project. By drawing on their great expertise, they have supported us through theoretic discussions, practical implementation, as well as much great advice. This project would not have been possible without their fantastic guidance.

Likewise, we would like to thank our academic supervisor Lennart Svensson for his continued support throughout the project. His insights during discussions and reviews has allowed us to see things from new perspectives, always giving us opportunity to learn something new. His support has been priceless, both academically, where his vast academic experience has been indispensable during all things writing-related; and practically, where giving us access to the National Supercomputing Center has made our lives significantly easier.

Finally, our gratitude goes to Zenseact AB for allowing us access to their resources, as well as the support they have provide along the way. The employees have been nothing but helpful, assisting in any way they can whenever needed. It is a company full of great people, and it has been our pleasure to work with them.

Artur Gasparyan and Ruiqi Qiu, Gothenburg, 2024-12-12

Contents

Acronyms

AD Autonomous Driving. v, 1–3, 7, 9–11, 23, 39

AV Autonomous Vehicle. 1

CNN Convolutional Neural Network. 3, 11, 32

DDIM Denoising Diffusion Implicit Model. 33

DDPM Denoising Diffusion Probabilistic Model. 15–18

DM Diffusion Model. v, 1, 4, 16, 21, 23, 33, 34

FID Frechet Inception Distance. 3, 33, 34

GAN Generative Adversarial Network. 4

KL Kullback-Leibler. 17

LDM Latent Diffusion Model. xiv, 4, 5, 15, 16, 18, 22, 23

LoRA Low-Rank Adaptation. v, xiv, xv, xvii, 2, 5, 15, 18, 23–28, 34, 38, 39

LPIPS Learned Perceptual Image Patch Similarity. xvii, 24–27, 33

MLP Multi-Layer Perceptron. 4, 5, 11

NeRF Neural Radiance Field. v, xv, 1–5, 7, 9–11, 23–27, 31–33, 37

NVS Novel View Synthesis. v, 1, 2, 4, 9

PSNR Peak Signal to Noise Ratio. xvii, 26, 27, 33

SSIM Structural Similarity Index Measure. xvii, 26, 27, 33

VAE Variational Autoencoder. 5, 15–18, 33

List of Figures

List of Tables

1

Introduction

Autonomous vehicles have become increasingly popular in recent years, powered by advances in Autonomous Driving (AD) [1]. Much of this progress has been preceded by developments in artificial intelligence, where the key focus has shifted towards data-driven deep learning solutions [2]. The deployment of physical cars requires an emphasis on robustness and safety, stressing the need for high-quality data in controlled environments to provide more accurate input. Although simulators are used today for this purpose, they have numerous shortcomings, including difficulty modeling the complexity of real-world dynamics, and a laborious development process, resulting in high costs [3]. This limits their viability for testing robust deep learning systems.

To address these challenges, an emerging solution involves leveraging Novel View Synthesis (NVS) techniques on the sensory input of Autonomous Vehicles (AVs) to generate synthetic data. Recent studies highlight the effectiveness of NVS in improving perception systems for AVs [4]–[10]. We aim to improve on the state-of-the-art in this area, particularly in the reconstruction quality of areas of the simulation not covered by the ground truth, by using Diffusion Models (DMs) to generate training data for parts of the scene that lack ground truth data.

1.1 Overview

Recent advances in scene reconstruction have enabled data-driven simulations of scenes based on sensory inputs [5], [11]. Rather than manually building a simulator, NVS is used to generate sensory inputs for an agent's new state. This process involves the use of recordings from vehicles to create a virtual environment, which gives us sensor-realistic simulation. In this digitally constructed environment, it becomes possible to explore various scenarios and paths, thus facilitating closed-loop simulation for different agents. Additionally, this technique allows for scene editing, where dynamic actors, background conditions, and other aspects of the scene can be modified to explore different scenarios. This approach is also valuable for data augmentation purposes, for example, by capturing the same scene from multiple perspectives and angles.

Recent progress in NVS is largely based on NeRFs and variants of it [12], like Instant-NGP [13], Lerf [14], NeRFPlayer [15]. In the original NeRF architecture, classical computer graphics are combined with neural networks to render sensory input from

a given camera pose. An image is rendered by emitting rays from the camera, estimating the color and density at multiple points along the ray, and aggregating this to an expected color using volume rendering. The expected color depends on the color and density of the objects it passes through, which are estimated with a feedforward neural network. However, a limitation of this method is its difficulty in generalizing to parts of the scene that are not visible in the training images. Particularly in autonomous driving scenarios, this poses a challenge in that even if we have multiple cameras in the car, there are still unseen views that the cameras cannot capture at certain angles. To overcome this, the model requires the capability to 'hallucinate' or infer these unseen areas, necessitating the integration of generative models into the system.

While there is substantial literature about applying generative models to NeRFs to improve their generalization ability [16]–[19], applying them to AD scenes imposes a set of particular challenges. One such challenge is the dynamic nature of AD scenes, with each frame representing a particular point in time. Pedestrians and vehicles move over time, making the reconstruction critically dependent on time and position.

Another challenge is that of scale. Although prominent work achieves great fidelity on inward-facing scenes where views are captured in an orbit around an object [12], AD scenes are outward-facing, with the view covering large scenes with many objects. Additionally, different weather and lighting conditions obfuscate certain visual features. The data might also lack views from certain angles since objects in AD scenes are moving, creating a need to generalize beyond the visible points in the data. Other sensors in the car, such as Lidar signals, share similar limitations for generalizing to new viewpoints since they have almost identical lines of sight.

In our work, we address these challenges by leveraging existing state-of-the-art approaches in NVS for AD, particularly NeuRAD [4]. Inspired by Reconfusion [17], we propose a pipeline to enhance the training of the NVS model to better handle unseen views. This is achieved by generating images with NeuRAD on unseen views, improving the output image using a diffusion model, and supervising NeuRAD with the enhanced image. Additionally, we extend the Reconfusion architecture by incorporating conditioning signals using ControlNet [20]. To efficiently train the diffusion model, we utilize LoRA [21], which models the change in the model parameters as a decomposed low-rank matrix instead of altering the full-rank matrix. We use LoRA to fine-tune the diffusion model for each scene before applying it to the diffusion model.

1.2 Related Work

1.2.1 Novel View Synthesis for Autonomous Driving

The application of NVS for AD has seen an increase in popularity following the introduction of neural closed-loop simulation. Neural Scene Graphs was the first to apply NeRFs for AD scenes [9]. The scene is separated between dynamic nodes and

background nodes, where the relations between objects are maintained using graphs. In the paper, there are no reported results on the reconstruction quality of unseen views, but comparisons made in [5] and [4] report significantly worse performance on a lane shift test, i.e., how well the scene is recreated when the camera laterally shifts a few meters.

The concept of splitting the scene into foreground and background is later repeated in MARS [22]. Here, the NeRFs are implemented modularly using the open-source tool Nerfstudio [23], enabling the switching and combination of different NeRF models. However, MARS has limited support for Lidar data, which is commonly included in AD datasets. Taking advantage of this modality gives potential advantages since Lidar provides geometric information in the form of 3D points. Further, MARS performance on views far from the training data was not evaluated.

A similar approach is taken in UniSim [5], where a scene is decomposed into dynamic actors, a static background, and a sky. Here, both Lidar and RGB data are used for the simulator. To improve efficiency, the NeRF generates features rather than images, which are then upscaled using a Convolutional Neural Network (CNN). However, this increases aliasing, as well as the discrepancy between RGB and Lidar, as the reconstructed rays represent RGB patches while Lidar rays are thin beams. UniSim also implements a discriminator network to improve render quality on unseen views, which is trained per scene. The ability to generalize to unseen views is compared to previous works with a lane shift test, where the Frechet Inception Distance (FID) [24] is benchmarked. UniSim significantly outperforms previous works on this metric. An ablation study is also done, where the FID is benchmarked against various components of the UniSim. Here, the GAN is demonstrated to have a significant effect in lowering the FID on the lane shift test.

NeuRAD [4] is also a simulator composed of foreground and background, which uses both Lidar and RGB data. It improves the reconstruction quality over UniSim with extensive sensor modeling for both camera and Lidar, achieving state-of-the-art performance on scene reconstruction for AD data. NeuRAD assumes actors to be rigid and does not support any deformations. Furthermore, the reconstruction quality is reduced for parts of the scene not visible in the training data, such as those obstructed in all the frames, or those not visible due to the angle of the camera. However, despite this reduction, NeuRAD is reported to outperform UniSim on the lane shift test, as reported on a set of benchmarks, where the model consistently gets lower FID scores.

DrivingGaussian [6] is another approach based on Gaussian Splatting [25], which replaces the neural networks of NeRFs with a set of distributions for the density of the points. While these point clouds are initialized from Structure from Motion in the original Gaussian Splatting [26], DrivingGaussian uses Lidar priors to improve initialization. Similar to the previous methods, DrivingGaussian manages dynamic scenes by decomposing the scene into a set of Gaussians, modeling the relation between them with a dynamic graph. The overall performance of DrivingGaussian is benchmarked against a set of comparable methods, including Neural Scene Graphs, and significantly outperforms them. The paper does not include comparisons to

NeuRAD or UniSIM and does not evaluate performance on previously unseen views. Additionally, the GPU memory requirements are reported to reach 192GB for training, which is significantly larger than the other models.

1.2.2 Improving Novel View Synthesis with Generative Models

Generative models have previously been applied to NeRFs to improve performance on sparse images and better generalize to unseen views. Several methods have extended NeRFs with Generative Adversarial Network (GAN) [27]–[29]. One drawback is their potential instability during training which can lead to mode collapse or low-quality outputs [30]. Following the increasing popularity of DMs [31], they have seen increasing use in NeRFs [16], [17]. DMs learn to synthesize images by gradually adding Gaussian noise to an image (forward process) and learning to reverse the process to maximize the likelihood of the data (backward process), typically with the use of a denoising U-Net. However, because they typically operate directly in pixel space, their computational requirements are immense, requiring prohibitively large computational resources to train.

Latent Diffusion Models (LDMs) [32] build on this concept by performing the forward and backward process in latent space. By making use of a powerful pre-trained encoder-decoder, LDMs embed an image in latent space before running the diffusion process, and finally decoding the latent back to a denoised image. This significantly reduces training time, increasing the potential for application in other domains. Further, the embeddings can be combined with other sources of input, enabling the output of the model to be conditioned on additional information. In the original paper, the image embeddings are combined with encoded text captions by using cross-attention. This enables text-to-image synthesis by denoising random noise conditioned on the text caption.

The state-of-the-art in the combination of diffusion models and NeRFs is ReconFusion. It finetunes a pre-trained LDM to learn a plausible prior for NVS. The latent is conditioned on embedded outputs from PixelNeRF and CLIP [33], [34]. This prior is then distilled to a NeRF by regularizing it to render perceptually similar outputs to the diffusion model. In ReconFusion, the PixelNeRF architecture is modified to produce a useful conditioning signal rather than rendering a realistic image using volume rendering. The modified PixelNeRF encodes geometric scene information by embedding each input image using a U-Net, creating a spatial feature map. When querying for a point along the ray, the position is then projected to the view space of each input image to select a feature vector from each feature map, which is aligned with the input image. For each ray, the points are aggregated using Multi-Layer Perceptrons (MLPs), creating a set of feature vectors, which encode the pose and spatial information. These feature vectors are then further processed to produce useful features, which get concatenated with the latent. Similarly, CLIP embeddings are used to encode high-level information about the scene. CLIP learns joint embeddings for text and images by training on large captioned image datasets on the internet. The image and caption are embedded with a pre-trained image encoder and text encoder

respectively, before being projected to the same semantic space using MLPs. These MLPs are optimized using contrastive learning, where the cross-entropy loss is minimized between the cosine similarity of the embeddings and their corresponding label. The combination of the CLIP embeddings and PixelNeRF embeddings, along with the LDM, provide a strong prior which ReconFusion uses to optimize the NeRF for novel views, significantly improving the performance over previous few-view NeRF reconstruction approaches. By leveraging a pre-trained LDM to generate plausible scene images from novel viewpoints, ReconFusion uses this diffusion-based prior to guide the NeRF optimization beyond the observed training images. As a result, the NeRF is refined not only to reconstruct the given views but also to generalize effectively to previously unseen viewpoints. By incorporating the diffusion prior, ReconFusion significantly improves upon previous few-view NeRF reconstruction methods, while also enhancing performance in settings where multiple input images are available.

1.3 Scope and Outline

Our thesis focuses on improving NeuRAD's ability to handle views not included in its training data. Although NeuRAD achieves state-of-the-art performance, it struggles with views in areas of the scene that are not covered during training. By integrating the Stable Diffusion model and leveraging advanced diffusion techniques, we aim to improve NeuRADs capacity to generate accurate and visually consistent images in unseen views.

In Chapter 1, we begin with a comprehensive introduction to the challenges and objectives of this thesis. This chapter explores the role of novel view synthesis in autonomous driving, highlighting its significance and the critical problems encountered in its implementation. We then introduce the concept of diffusion models and outline how they offer a promising solution to overcome these challenges.

In Chapter 2, we talk about the foundational concepts of NeRFs. We also introduce the overview of the NeuRAD model. Then we identify its current limitations.

In Chapter 3, we give the essential theoretical background on Variational Autoencoders (VAEs), LDMs, and an introduction to the Stable Diffusion model. We also describe the process of fine-tuning the diffusion model using LoRA. Here, we explain how conditioning is added to the model with ControlNet. In the end we analyze the results of fine-tuned diffusion model.

In Chapter 4, we introduce the pipeline developed to integrate NeuRAD with Stable Diffusion. We detail the methodology, present the experimental setup, and showcase the final results. We also discuss the performance impact and limitations of our integrated models.

In Chapter 5, we summarize our work and discuss potential future developments.

2

Neural Rendering for Autonomous Driving

In this chapter, we describe the role of Neural Rendering in AD and investigate the current solutions. We begin with an overview of the theoretical foundations of NeRFs, which combine neural networks with volumetric rendering to model complex 3D scenes. NeRFs encode 3D scene representations, enabling the synthesis of novel viewpoints from sparse 2D image inputs. We provide an overview of this model and the core principles behind it. Next, we introduce the NeuRAD model, an extension of NeRF designed specifically for dynamic AD scenarios. NeuRAD builds upon NeRF's framework with extensive sensor modeling for both Lidar and RGB cameras to improve its performance. Finally, we examine the limitations of NeuRAD by evaluating the renders on poses far from the ground truth.

2.1 Overview of NeRFs

NeRFs are parameterized fields that represent volumetric information about a given scene with neural networks. A radiance field describes how light is emitted, reflected, or transmitted within a 3D scene. Learning a representation of the radiance field allows view synthesis using a neural network, enabling the rendering of images from different positions and directions in a scene. Given a set of images with the corresponding position and viewing direction of the camera (camera pose), NeRFs learn this mapping from poses to views. As given in [12], the standard approach is done by sampling along rays going into the scene and then using volume rendering to estimate the color of each ray. Volume rendering uses a neural network to predict the color and volume density of the sampled points in the ray and then aggregates this information for an estimated ray color. The NeRF then optimizes this network to better predict the color and volume density of each point, and thus, a more accurate estimate of the color of the ray. An overview of the basic NeRF-setup can be seen in Figure 2.1.

2.1.1 Fundamental NeRF theory

To optimize a neural network in the context of NeRFs, we match sampled rays to pixels in a given image to obtain ground truth colors. This involves segmenting the camera view into a grid, where each grid cell corresponds to a pixel in the projected

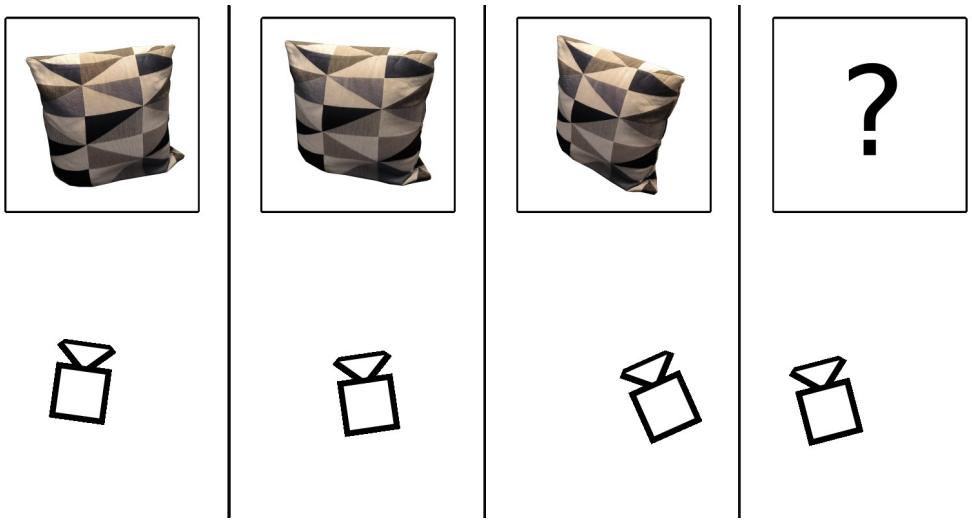


Figure 2.1: A set of image-camera pairs span a trajectory around an object, in this case, a pillow (panels 1-3). The task then becomes to infer what the image looks like from a novel camera view (panel 4).

image. To estimate the color of each cell, a ray is cast from the camera origin o in the view direction through the pixel d. The color of the pixel in the rendered image is determined by the point that intersects the ray,

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}, \tag{2.1}$$

and is visible from the camera's perspective.

In volume rendering, the expected color, C, of the ray depends on the likelihood of a ray collision at a given point combined with the expected color, c, of the point. We assume that a denser point is more likely to collide with the ray, and thus represent this likelihood as the volume density σ. From this, we get the equation

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})\, dt, \tag{2.2}$$

where $T(t)$ is defined as the accumulated transmittance

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))\, ds\right). \tag{2.3}$$

Here, t_n and t_f represent the near and far planes of the camera, $\sigma(\mathbf{r}(t))$ represents the volume density at a given point, and $c(\mathbf{r}(t), \mathbf{d})$ is the expected color of the point.

This integral is solved by sampling points along the ray, and solving a discretized version of the equation given by

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c_i}, \tag{2.4}$$

where $\delta_i = t_{i+1} - t_i$. The gap between points i and $i+1$ on the projected line of the ray is denoted by δ_i. The accumulated transmittance can be viewed as the probability of reaching point i without colliding with another particle.

$$\alpha_i = (1 - \exp(-\sigma_i \delta_i)), \tag{2.5}$$

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j). \tag{2.6}$$

Similarly, α_i can be viewed as a conditional probability of the ray colliding with point i given that the ray has reached that point. Thus, we define

$$w_i = T_i \alpha_i, \tag{2.7}$$

which is proportional to the probability of the ray colliding at a point.

To compute this discretized integral, points must be sampled along the ray. A naive implementation would be to segment the ray into even splits and to use those points for the integral. However, this would effectively lower the resolution of the scene representation since the network does not need to represent the points between the splits. Instead, the standard approach is to split the ray into evenly spaced bins, with one point uniformly sampled from each bin.

Additionally, to compute the integral, the estimated color and volume density need to be given. This is done with a multilayer perceptron, which takes point position $\mathbf{x}$ and viewing direction $\mathbf{d}$ as inputs and computes σ and c. However, neural networks struggle to learn high-frequency outputs from low-frequency inputs [35], and therefore their inputs $\mathbf{x}$ and $\mathbf{d}$ are first embedded using positional encoding, where the inputs are projected to higher dimensional space using sinusoidal functions, defined as

$$\begin{aligned} \gamma(p)_{2i} &= (\sin(2^i \pi p), \\ \gamma(p)_{2i+1} &= (\cos(2^i \pi p). \end{aligned} \tag{2.8}$$

2.1.2 Limitations in NeRFs for AD Scenes

While NeRFs prove promising for NVS, applying them to AD scenes imposes a set of particular challenges. One such challenge is the dynamic nature of AD scenes, with each frame representing a particular point in time. Pedestrians and vehicles move over time, making the reconstruction critically dependent on time in addition to position.

Another challenge in the application of NeRFs to AD scenes is that of scale. While prominent work achieves great fidelity on inward-facing scenes, where views are captured in an orbit around an object [12], AD scenes are outward-facing, with the view covering large scenes with many different objects.

Additionally, different weather and lighting conditions are obfuscating certain visual features. The data might also lack views from certain angles since objects in AD scenes are moving, creating a need to generalize beyond the points in the data. In certain environments, this is somewhat mitigated by additional sensors, such as

Lidar, but these pose an additional challenge of utilization and integration with the other sensors.

2.2 Neural Rendering for Autonomous Driving

Neural rendering techniques have been increasingly adopted in AD applications to address the challenges of novel view synthesis, dynamic object rendering, and efficient scene modeling. While traditional NeRF models excel in static environments, AD scenes require handling complex dynamics, large-scale environments, and sensor-specific data such as Lidar and RGB images. NeuRAD is a state-of-the-art approach that extends NeRF models to address these challenges, building on foundational techniques like scene graph decomposition and hash encoding. This section provides an overview of key techniques leading up to NeuRAD and its contributions to AD applications.

2.2.1 Scene Graphs

Neural Scene Graphs [9] introduced a pioneering approach for rendering dynamic scenes by decomposing them into static and dynamic components structured within a scene graph. This graph encodes object transformations and radiance, enabling efficient rendering of novel views and dynamic arrangements of objects. By leveraging video frame supervision and tracking information, the model learns to represent scene elements as augmented implicit neural radiance fields, following the methodology established in NeRF [12].

A scene S in Neural Scene Graphs is defined as:

$$S = \langle W, C, F, L, E \rangle,$$

where W is the root node of the graph, C are the leaf nodes representing the camera, F are the leaf nodes representing both static and dynamic scene objects, L are the leaf nodes assigning latent codes to object representations, and E are the edges connecting the nodes.

This approach enables inverse rendering, where 3D object detection is performed by optimizing scene representations. Moreover, it allows for novel view synthesis and manipulation of dynamic object arrangements, bolstering its utility for neural rendering in AD.

2.2.2 Hash Encoding

Neural graphics primitives, parameterized by fully connected neural networks, can be costly to train and evaluate. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding[13] reduce this cost with a versatile new input encoding that permits the use of a smaller network without sacrificing quality, thus significantly reducing the number of floating point and memory access operations.

A small neural network is augmented by a multiresolution hash table of trainable feature vectors whose values are optimized through stochastic gradient descent. The multiresolution structure allows the network to disambiguate hash collisions, making for a simple architecture that is easy to parallelize on modern GPUs. It achieves a combined speedup of several orders of magnitude, enabling the training of high-quality neural graphics primitives in a matter of seconds, and rendering in tens of milliseconds at a resolution of 1920 * 1080.

2.2.3 Overview of NeuRAD

NeuRAD [4] extends NeRF models specifically for autonomous driving scenes, addressing the challenges posed by dynamic environments and sensor-specific data. It achieves state-of-the-art performance across multiple AD datasets by incorporating actor-aware hash encoding and unified feature fields.

During training, NeuRAD learns a unified neural feature field for both the static and dynamic elements of an automotive scene. These elements are distinguished through an actor-aware hash encoding, which enables the system to differentiate between static and dynamic components. Points within actor bounding boxes are transformed into actor-local coordinates, and combined with the actor index, they are used to query a 4D hash grid. The features extracted from this grid are then decoded: volume-rendered ray-level features are converted to RGB values using an upsampling CNN, while ray drop probability and intensity are predicted using MLP.

Separating the scene into static and dynamic elements enables easier manipulation of the learned scene, since each dynamic actor can be modified or removed as needed. These capabilities make NeuRAD ideal for use in applications like sensor-realistic closed-loop simulators or advanced data augmentation systems, providing a flexible foundation for improving the performance and reliability of autonomous driving technologies.

2.3 Limitations of NeuRAD for AD Scenes

Although NeuRAD achieves state-of-the-art results on five publicly available AD datasets and demonstrates both effectiveness and robustness, it still has limitations that require improvement. Like other NeRF-based models, NeuRAD's reconstruction accuracy heavily depends on the quality of the input images. When input images are obstructed by obstacles or other objects, NeuRAD struggles to render fine details, resulting in incomplete or blurred regions in the final reconstruction. This challenge is particularly evident in scenarios with partial occlusions or interruptions in the line of sight. Additionally, when the camera or vehicle capturing the images is in motion, certain angles of the scene may not be captured at all, leading to coverage gaps that limit the model's ability to reconstruct the scene from all perspectives.

To emphasize NeuRAD's limitations, we train the model on a specific subset of a scene and evaluate its performance on previously unseen parts. Using scene 001

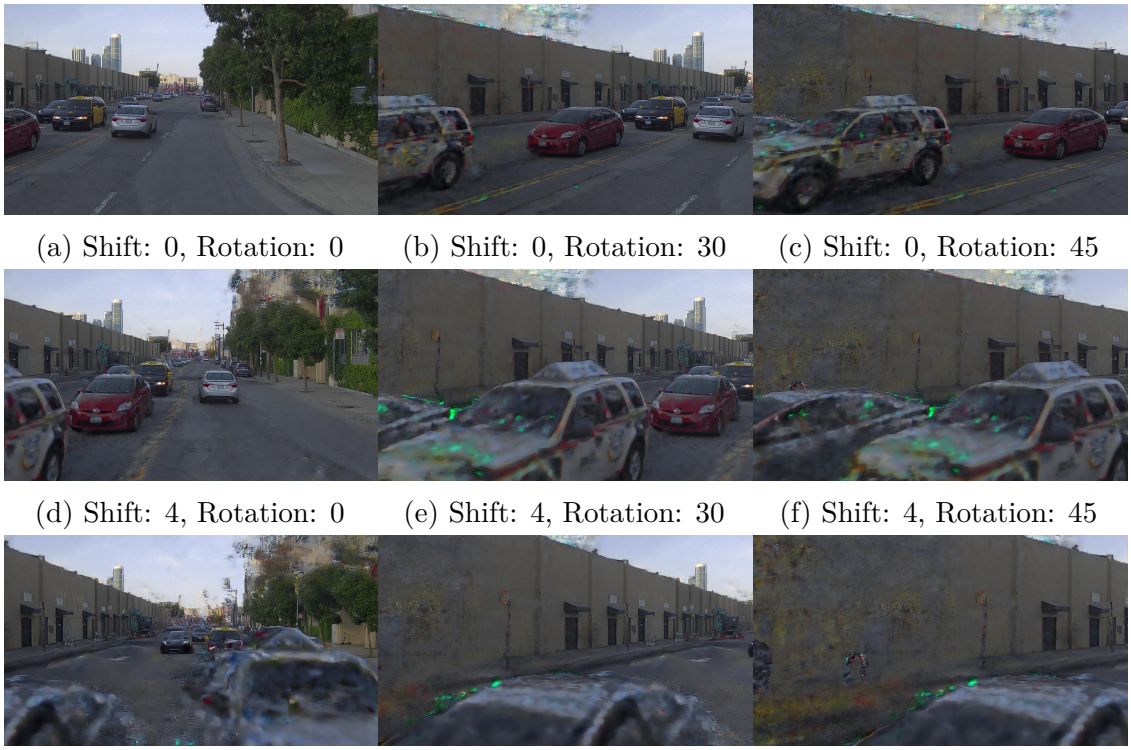
from Pandaset, the model is trained exclusively on images captured by front-facing cameras. Once training is complete, we test the model's ability to render novel views by altering the cameras position and orientation.

As shown in Figure 2.2, NeuRAD struggles to reconstruct areas outside its training perspectives. When the camera is rotated to the left, regions that were not visible during training appear incomplete or distorted. Conversely, the model performs well in rendering parts of the scene within its trained field of view, such as the right side, which aligns with its training images. This highlights NeuRAD's reliance on the specific angles and perspectives seen during training.

Further results, shown in Figure 2.3, illustrate NeuRAD's performance when rendering novel views with varying camera shifts and rotations. In Figure 2.3 (a), NeuRAD produces detailed reconstructions when the camera angle is similar to those observed during training. However, as the camera shifts several meters to the left or rotates to the right by a larger angle, as seen in Figure 2.3 (b-i), rendering quality deteriorates. For example, the bottom right region in Figure 2.3 (b) appears clear and detailed, while the upper left region shows a significant loss of detail. This degradation becomes more pronounced as the viewpoint deviates further from the original training perspectives, underscoring the model's current limitations in generalizing to novel viewpoints.

These observations reaffirm NeuRAD's difficulty in reconstructing unseen perspectives, as previously discussed. Enhancing the model's ability to handle a broader range of viewpoints is a key focus of this thesis, as such improvements would enable more robust and flexible scene generation. For instance, in autonomous driving scenarios, a vehicle often encounters occluded areas due to buildings, other vehicles, or pedestrians. NeuRAD, in its current state, struggles to accurately reconstruct these hidden or obstructed regions, which limits its applicability in complex, real-world environments.

Figure 2.2: NeuRAD struggles to accurately render sections of the scene that were not observed during training. In this instance, the model is only trained with front-facing camera images. The figure shows the NeuRAD rendered image with the camera shifted 15 degrees to the left, thus illustrating the sharp quality difference between covered and non-covered sections.



(a) Shift: 0, Rotation: 0 (b) Shift: 0, Rotation: 30 (c) Shift: 0, Rotation: 45

(d) Shift: 4, Rotation: 0 (e) Shift: 4, Rotation: 30 (f) Shift: 4, Rotation: 45

(g) Shift: 8, Rotation: 0 (h) Shift: 8, Rotation: 30 (i) Shift: 8, Rotation: 45

Figure 2.3: NeuRAD-renders across various shifts and rotations. In this setup, a NeuRAD model is trained on front-facing images. (a) shows the render from a view taken from the training dataset. (b-i) illustrate renders across various augmented views, including shifting the camera *Shift* meters along the camera-local x-axis (rows 1-3), and rotating the camera *Rotation* meters around the camera-local z-axis (columns 1-3).

3

Improving Diffusion Models on Autonomous Driving Scenes

To address the challenges outlined in Chapter 2, we propose leveraging generative models to enhance scene reconstruction in autonomous driving scenarios. Generative models are machine learning systems designed to generate new data samples that resemble the training data by learning its underlying distribution. These models have been successfully applied to generate realistic outputs such as images, text, audio, and 3D objects. In this chapter, we focus on utilizing pretrained generative models, specifically diffusion models, to address limitations in NeuRAD. Diffusion models represent the state-of-the-art in vision-based generative modeling. By incorporating their learned priors in parts of the scene where our dataset lacks information, we aim to mitigate NeuRADs current constraints and improve its performance in challenging scenarios, such as unseen perspectives or occluded regions.

This chapter begins with an overview of diffusion models, outlining their foundational principles, including Variational Autoencoders (VAEs), Denoising Diffusion Probabilistic Models (DDPMs), and Latent Diffusion Models (LDMs). This provides the necessary context for understanding their application in our approach. Next, we explore how diffusion models can enhance noisy NeRF outputs by improving image clarity and detail. We also introduce key extensions to diffusion models, including Low-Rank Adaptation (LoRA) [21] and ControlNet [20], and explain how these methods are adapted to our data. Finally, we present experiments examining the performance of the diffusion model for this task. These experiments include a parameter sweep to identify optimal noise levels, fine-tuning the model for specific scenes, and testing control signal mechanisms. Together, these steps refine the diffusion model to better complement NeuRAD and enhance scene reconstruction quality in autonomous driving datasets.

3.1 Generative Models

Recent advances in generative models are tied to advances in self-supervised learning, where models are trained to uncover hidden structures in data without relying on explicit labels. One of the foundational frameworks in this domain is the Autoencoder [36], which learns to compress data into a low-dimensional latent space and then reconstruct it from this compressed form. The U-Net architecture [37] extends this

concept by introducing residual skip connections between matching encoder-decoder blocks, making it particularly effective for tasks such as image segmentation and denoising. A variant of U-Net, the denoising U-Net [38], extends these capabilities by learning to remove noise from corrupted inputs, a key component in generative modeling.

Diffusion Models (DMs), particularly Denoising Diffusion Probabilistic Model (DDPM) [31], leverage denoising U-Nets to approximate the underlying data distribution. These models gradually add Gaussian noise to data during a forward process and learn to reverse this process during a backward step to recover the original data. The denoising U-Net serves as the backbone of this approach, iteratively refining noisy data samples toward the training data distribution.

Another significant generative framework is the Variational Autoencoder (VAE) [39], which extends the autoencoder by learning a probabilistic distribution over the latent space. Unlike traditional autoencoders, VAEs model the input data distribution explicitly, enabling both data generation and probabilistic reasoning. By combining the generative capabilities of VAEs with the iterative noise-removal process of DDPMs, Latent Diffusion Models (LDMs) [32] were introduced. LDMs operate in the latent space of pretrained encoder-decoder networks, significantly reducing the computational complexity of diffusion models while preserving their ability to generate high-quality data.

In this section, we examine the theoretical foundations of latent diffusion models, focusing on autoencoders, DDPMs, and LDMs. This theoretical discussion provides the necessary context for understanding the principles and mechanics of these generative models, which will inform their broader applications in later sections.

3.1.1 Autoencoders

Autoencoders [36] are among the foundational approaches for representing data in a latent space. As shown in Figure 3.1, an autoencoder consists of two main components: an encoder, $\mathcal{E}(\mathbf{x})$, which maps input data $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ to a latent representation $\mathbf{z} \in \mathbb{R}^{D}$, and a decoder, $\mathcal{D}(\mathbf{z})$, which reconstructs the input data from the latent space [36]. The encoder effectively compresses the input data into a low-dimensional latent space, while the decoder reconstructs samples drawn from this space to approximate the original input.

A prominent extension to autoencoders is the Variational Autoencoder (VAE) [39]. VAEs differ from traditional autoencoders by introducing a probabilistic framework that models the relationship between the input data $\mathbf{x}$ and the latent variable $\mathbf{z}$ using the distributions $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$. The encoder approximates the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$, often modeled as a Gaussian with learnable mean and variance parameters. The decoder models the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$, which measures how well the reconstructed output aligns with the input. The VAE is optimized by maximizing the Evidence Lower Bound (ELBO),

$$\text{ELBO}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})), \tag{3.1}$$

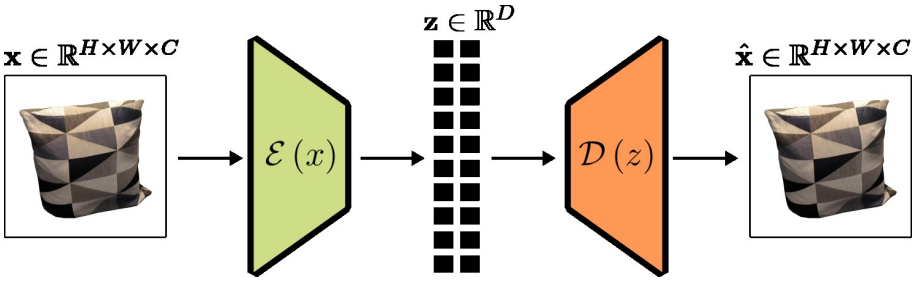


Figure 3.1: Overview of an autoencoder. It consists of two main components: the encoder and the decoder. The encoder maps input data (e.g., an image) to a lower-dimensional latent space, capturing the most essential features. The decoder then reconstructs the data from this latent representation.

where the first term evaluates the reconstruction quality, and the second term, a Kullback-Leibler (KL) divergence, ensures the encoder output aligns with the prior distribution $p(\mathbf{z})$, typically a standard Gaussian.

The encoder and decoder are implemented as neural networks. During training, the encoder predicts the mean and variance of $q_\phi(\mathbf{z}|\mathbf{x})$, from which samples are drawn. The decoder then reconstructs the input based on the sampled latent variable. This probabilistic framework enables the VAE to generate new samples by sampling $\mathbf{z} \sim p(\mathbf{z})$ and decoding it via $\mathcal{D}(\mathbf{z})$.

3.1.2 Denoising Diffusion Probabilistic Model (DDPM)

Denoising Diffusion Probabilistic Models (DDPMs) [31] are a class of generative models that iteratively transform Gaussian noise into samples resembling the input distribution. DDPMs consist of two processes: the forward diffusion process, $D(\mathbf{z}_t)$, which gradually corrupts the data by adding Gaussian noise at each timestep t, and the reverse diffusion process, $D^{-1}(\mathbf{z}_t)$, which removes noise step-by-step to reconstruct the original data. These processes are parameterized such that $\mathbf{z}_T$ at the final timestep approximates pure Gaussian noise, $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

In the forward process, the noise at each timestep t is added using:

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t|\sqrt{\alpha_t}\mathbf{z}_{t-1}, (1-\alpha_t)\mathbf{I}), \tag{3.2}$$

where α_t controls the scale of the noise. By combining these steps, the distribution $q(\mathbf{z}_t|\mathbf{z}_0)$ can be directly expressed as:

$$q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t|\sqrt{\bar{\alpha}_t}\mathbf{z}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{3.3}$$

where $\bar{\alpha}_t$ is the cumulative product of noise coefficients. The reverse process, $D^{-1}(\mathbf{z}_t)$, predicts the mean and variance of $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0)$ using a denoising U-Net [38]. The model is trained by minimizing the variational ELBO, which consists of reconstruction, prior matching, and consistency terms:

$$\text{ELBO} = \mathbb{E}_q[\log p_\theta(\mathbf{z}_0|\mathbf{z}_1)] - \sum_{t=2}^{T} \mathbb{E}_q[\|\mu_q(\mathbf{z}_t, \mathbf{z}_0) - \mu_\theta(\mathbf{z}_t)\|^2]. \tag{3.4}$$

The learned parameters θ predict the noise added at each timestep, enabling the model to denoise iteratively during inference. Training involves randomly sampling timesteps t, corrupting data according to $q(\mathbf{z}_t|\mathbf{z}_0)$, and optimizing the network to predict the original data.

3.1.3 Latent Diffusion Models (LDM)

Latent Diffusion Models (LDMs) [32] extend DDPMs by operating in a latent space instead of pixel space, significantly reducing computational requirements while retaining high-quality synthesis capabilities. LDMs utilize an autoencoder structure, based on VAE and VQVAE [40], to encode input data $\mathbf{x}$ into a latent representation $\mathbf{z} = \mathcal{E}(\mathbf{x})$. This autoencoder is trained by combination of a perceptual loss [41] and a patch-based adversarial objective [42]. The diffusion process is then applied in the latent space, $D(\mathbf{z}_t)$, rather than directly on pixel values. The forward process adds noise to the latent, producing a noisy latent $\hat{\mathbf{z}}$, and the reverse process removes the noise step-by-step using a U-Net. The denoised latent $\bar{\mathbf{z}}$ is finally decoded with $\mathcal{D}(\mathbf{z})$ to generate a recreated image, $\hat{\mathbf{x}}$.

Because the LDM operates on latent vectors, other signals can also be encoded into the same latent space. This enables conditioning on additional inputs such as text captions using cross-attention. This is typically used to create text-to-image models, by combining the latent from the images with the latent of the corresponding text captions. Figure 3.3 illustrates the diffusion process in an LDM, while Figure 3.2 details its architectural layers.

Fig 3.3 shows that the LDMs can be conditioned via concatenation or by cross-attention. It turns diffusion models into powerful and flexible generators for general conditioning inputs such as text or bounding boxes and high-resolution synthesis becomes possible in a convolutional manner. It also shows that LDMs achieve new state-of-the-art scores for image inpainting and class-conditional image synthesis and highly competitive performance on various tasks, including text-to-image synthesis, unconditional image generation and super-resolution, while significantly reducing computational requirements compared to pixel-based DMs.

3.2 Finetuning with Low-Rank Adaption

Adapting large, pretrained diffusion models to specific tasks or domains can be computationally expensive due to the substantial number of parameters involved. Low-Rank Adaptation (LoRA) [21] addresses this issue by reducing the number of trainable parameters during fine-tuning while keeping the original model weights unchanged.

LoRA introduces low-rank matrices $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ into selected layers of the model, where r is a tunable rank parameter. These matrices decompose the

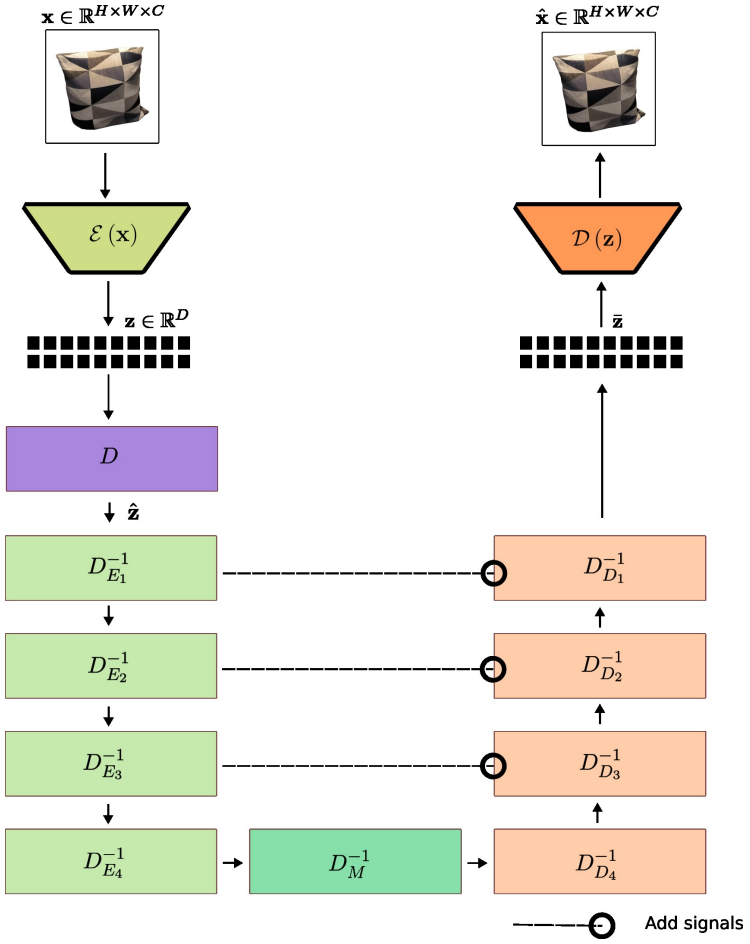


Figure 3.2: An overview of the LDM architecture. An image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is encoded to latent space $\mathbf{z} \in \mathbb{R}^{D}$ with an encoder $\mathcal{E}$. The latent passes through the forward diffusion process, D, resulting in the noisy latent $\hat{\mathbf{z}} = D(\mathbf{z})$. This then goes through the reverse diffusion process, D^{-1}, which is implemented as a denoising UNet with encoding blocks D_E^{-1}, middle block D_M^{-1}, and decoding blocks D_D^{-1}. Between each matching encoding and decoding block, $\left(D_{E_i}^{-1}, D_{D_i}^{-1}\right)$, there is a residual connection passing from the encoder to the decoder. The denoised latent $\hat{\mathbf{z}} = D^{-1}(\bar{\mathbf{z}})$ is then decoded with the pre-trained decoder $\mathcal{D}$, resulting in the recreated image $\hat{\mathbf{x}}$.

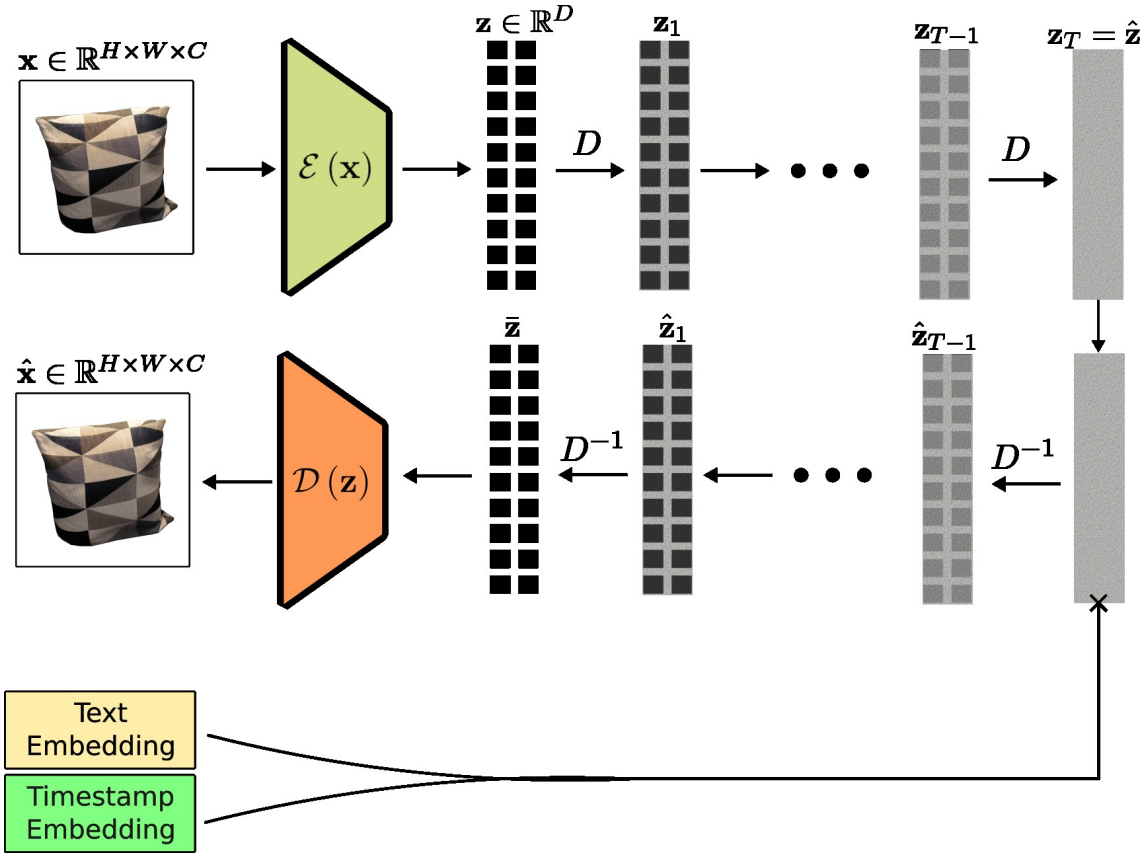


Figure 3.3: Overview of the diffusion process in an LDM. An image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is encoded to latent space $\mathbf{z} \in \mathbb{R}^{D}$ with an encoder $\mathcal{E}$. The latent goes through the forward diffusion process, D, during which, Gaussian noise is added at each timestep, t, resulting in the intermittent latent $\mathbf{z}_i$. This continues until timestep T, at which point, the noisy latent $\mathbf{z}_T = \hat{\mathbf{z}}$ is acquired. If run to completion, this noisy latent is equivalent to gaussian noise, $\hat{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{0} \in \mathbb{R}^{D}, \mathbf{I} \in \mathbb{R}^{D \times D}$. The noisy latent then goes through the reverse diffusion process, D^{-1}, where a parameterized model iteratively predicts the denoised latent $\hat{\mathbf{z}}_{t-1} = D^{-1}(\hat{\mathbf{z}}_t)$, at each timestamp t. At the finally step, $t = 0$, the denoised latent $\bar{\mathbf{z}} = \hat{\mathbf{z}}_0$ is uncovered. This is then decoded with the pre-trained decoder $\mathcal{D}$, resulting in the recreated image $\hat{\mathbf{x}}$.

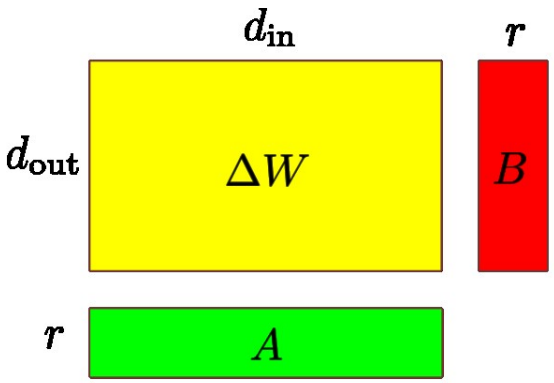


Figure 3.4: Matrix composition using LoRA. The parameter update, $\Delta\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is factorized into low-rank matrices, $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$, where r is a tuneable parameter affecting the size of the model. By only finetuning A and B, the number of parameters, represented by the area of the shapes, is significantly reduced during training.

parameter update $\Delta\mathbf{W}$ such that $\Delta\mathbf{W} = BA$. To ensure that the adaptation process does not initially alter the behavior of the pretrained model, B is zero-instantiated at the start of training. This guarantees that $\Delta\mathbf{W}$ starts as a zero matrix, leaving the output of the model identical to that of the original pretrained model.

During fine-tuning, the original model weights remain frozen, and only the low-rank matrices A and B are updated. The output of the adapted model for a given input $\mathbf{x}$ can then be expressed as:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \Delta\mathbf{W}\mathbf{x}, \tag{3.5}$$

where $\mathbf{W}\mathbf{x}$ represents the output of the original pretrained model, and $\Delta\mathbf{W}\mathbf{x} = (BA)\mathbf{x}$ introduces task-specific adjustments learned during fine-tuning. This formulation allows the model to combine the knowledge of the original pretrained weights with the new information introduced by A and B.

3.3 Conditioning on Additional Signals with ControlNet

DMs can be conditioned on additional inputs, such as edge maps, human pose skeletons, segmentation maps, or depth images, to guide the generation process. These conditioning signals provide explicit control, enabling task-specific or personalized image generation. Learning conditional controls for large pre-trained text-to-image DMs presents significant challenges. These include limited training data for specific conditions and the risk of overfitting or catastrophic forgetting when fine-tuning large models. ControlNet [20] addresses these issues by introducing an efficient end-to-end architecture for incorporating conditional signals without compromising the

performance of the pre-trained model.

As illustrated in Figure 3.5, ControlNet extends the architecture of an LDM by freezing the original encoder weights and adding a trainable copy of the encoder and middle block. The conditioning signal $\mathbf{c}$, provided as an image-shaped input (e.g., edge maps or depth images), is encoded using a pre-trained encoder $\mathcal{E}_c$ to produce a feature vector $\mathbf{c}_f$. This feature vector is processed through zero-convolution layers, which are initialized to zero, before being passed to the trainable copy of the model. The outputs of the trainable layers are further processed with zero-convolutions and then added to the corresponding layers of the original frozen model. This integration ensures that the trainable components augment the pretrained model's capabilities without degrading its original performance.

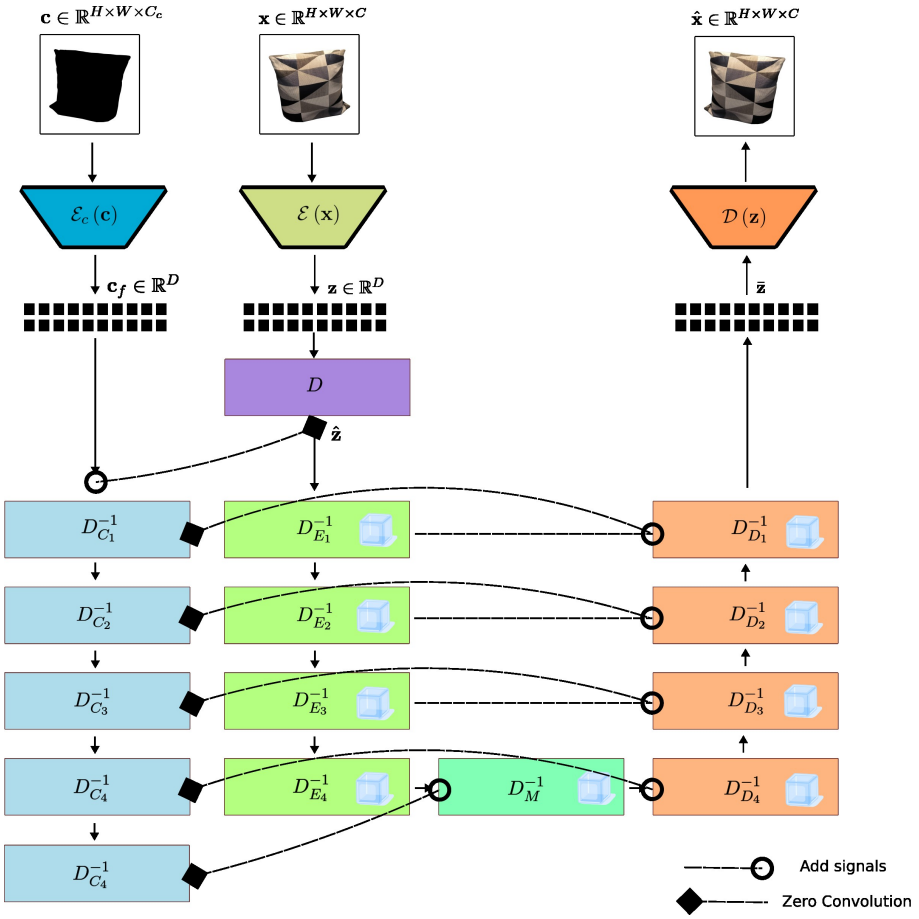


Figure 3.5: An overview of the ControlNet architecture. An LDM backbone, seen in Figure 3.2, is extended by freezing each layer and adding a trainable copy of the encoder blocks and middle block. An image-shaped conditioning signal $\mathbf{c}$ is encoded with another pre-trained encoder $\mathcal{E}_c$, resulting in a conditioning feature vector $\mathbf{c}_f$. Zero-convolutions are applied to this feature vector before it is passed to the trainable copy. Another zero-convolution is applied to the output of each trainable block before the final output is added to the corresponding mid- and decoder block.

3.4 Adapting Stable Diffusion to improve NeRF renders in AD Scenes

To integrate diffusion models into NeuRAD for AD, we adapt Stable Diffusion v2, an open-source LDM provided by HuggingFace [43]. Stable Diffusion is a state-of-the-art text-to-image model trained to generate high-quality, photo-realistic images based on text prompts. For this application, we leverage its pre-trained weights and fine-tune the model to suit the specific requirements of AD scenes. In this section, we describe how the previously mentioned techniques are implemented, and the evaluations we make to select a good configuration for integrating Stable Diffusion with NeuRAD.

While Stable Diffusion has been trained on diverse datasets, we assume that most objects within the AD scene are represented in its training data. However, the model lacks knowledge of the specific spatial layout of our target scenes, which increases the complexity of the latent space and reduces the likelihood of effective denoising. To address this, we fine-tune Stable Diffusion on images from each specific AD scene, tailoring a diffusion model to complement each NeRF model. Due to the computational demands of fine-tuning large diffusion models, we employ LoRA to efficiently update the model parameters while maintaining computational feasibility.

3.4.1 Tuning Noise Strength

A critical parameter in this adaptation process is the noise strength, which controls the level of noise added to an image before the sampling steps in the denoising process. In image-to-image tasks, the noise strength ranges from 0 (no noise) to 1 (maximum noise), and it determines the extent to which the diffusion model modifies the input. Figure 3.8 illustrates the impact of noise strength on diffusion outputs for two augmented reference views.

As seen in the figure, increasing the noise strength introduces more randomness to the input image, which can enable the model to address noise artifacts introduced by NeRF. However, excessive noise strength complicates the denoising process, often resulting in oversmoothing or loss of detail. Conversely, low noise strength limits the models capacity to correct artifacts, constraining improvements in image quality. Through experiments, we aim to identify an optimal noise strength that balances effective denoising with the preservation of image detail, ensuring that the diffusion model enhances the visual quality of NeRF renders without introducing additional distortions.

3.4.2 Finetuning on Autonomous Driving Dataset

AD scenes exhibit significant variation in road conditions, weather, time of day, and traffic. While Stable Diffusion is trained on a broad dataset, it lacks the specificity needed to accurately represent individual AD environments. To address this, we finetune the DM for each unique scene in NeuRAD, ensuring it captures scene-specific characteristics and improves the denoising of NeRF outputs.

We apply LoRA to the U-Net architecture, specifically targeting the cross-attention and linear layers. This approach allows efficient fine-tuning of the diffusion model for each scene while minimizing computational costs.

3.4.3 NeRF-Compatible Conditioning Signals

The original diffusion model employs text-to-image conditioning during the denoising process. However, AD scenes lack text annotations, necessitating the use of visual information extracted directly from the scene. To incorporate these signals, we utilize ControlNet, as outlined in Section 3.3, enabling the model to condition on visual inputs during training.

In the original ControlNet architecture, a trainable copy of the diffusion model is used. However, this approach introduces significant computational overhead, which conflicts with the efficiency of our LoRA strategy. To mitigate this, we replace the trainable copy with additional LoRA weights, reducing computational requirements. These weights are trained jointly with the low-rank weights of the original encoder, forming two complementary encoders that operate together during training.

When choosing which signals to condition on, some are more eective than others. Effective signals provide relevant scene information, such as *Spatial information*, i.e., depth and pose, which can encode object positions and geometry, and *temporal information*, where the video timestep can help capture motion dynamics. In practice, Conditioning signals must be available during NeRF training and conform to the image-shaped input format required by ControlNet.

For simplicity, we condition on pose information. During NeRF training, an image segment is sampled along with the corresponding camera pose. Rays are projected through the camera for each pixel, forming a ray bundle that describes the origin and direction of each ray in global coordinates. This ray bundle serves as the conditioning signal, guiding the diffusion model to align with the spatial structure of the scene.

3.4.4 Rendering Consistency of Diffusion Models

To distill the knowledge from a diffusion model into NeuRAD, it is critical to ensure that the diffusion process produces consistent outputs across frames and, more importantly, for repeated applications on the same input. High variability in outputs for the same frame can negatively impact NeRF training, as the radiance field relies on multiple views of the same regions to reconstruct the scene.

To evaluate consistency, we apply the diffusion model to the same input image multiple times, using different random seeds for each iteration. Figure 3.6 illustrates the resulting visual discrepancies. These variations were quantified using the Learned Perceptual Image Patch Similarity (LPIPS) metric, which measures perceptual differences closely aligned with human perception. LPIPS provides an objective measure of the variability introduced by the diffusion process.

(a) NeuRAD reference image (b) Diffusion model with Seed 0

(c) Diffusion model with Seed 1 (d) Diffusion model with Seed 2

Figure 3.6: The diffusion outputs on a NeuRAD-rendered image from an area with poor coverage in the training set. (a) displays the original NeuRAD render. (b-d) show the outputs of the diffusion process with seeds $0, 1, 2$ respectively. Notably, the diffusion model morphs small details, such as the noise on the ground, between each render. This comparison illustrates the visual inconsistency between subsequent diffusion renderings, even when the input image is completely unchanged. A qualitative comparison can be seen in Table 3.1.

Table 3.1 reports the LPIPS scores between diffusion outputs generated with different seeds and compares them to the original NeuRAD render. Notably, the LPIPS scores are higher between outputs generated with different seeds than between the NeuRAD output and the diffusion-processed image. This highlights the significant role of inherent randomness in the diffusion process, which can overshadow differences between diffusion models.

To investigate whether fine-tuning improves consistency, we repeated the experiments using diffusion models fine-tuned with LoRA at ranks 4 and 128, as well as models conditioned on ray information using ControlNet. Table 3.2 demonstrates that fine-tuning and conditioning significantly reduce LPIPS variability across different seeds. The additional context provided by the fine-tuned weights and control signals stabilizes the denoising process, resulting in more consistent outputs. This improved reliability is crucial for handling noisy NeRF renders and ensures better alignment across views, enhancing NeRF training.

Table 3.1: LPIPS between diffusion outputs from different seeds on a NeuRAD render. The distance metric LPIPS $(\mathbf{x}_i, \mathbf{x}_j)$ is reported for each $i \in \{0, 1, 2\}, j \in \{1, 2, 3\}$, where $\mathbf{x}_0$ is the reference NeuRAD output, while $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are the diffusion outputs with seeds $0, 1, 2$ respectively. A visual comparison can be seen in Figure 3.6.

	Seed 0	Seed 1	Seed 2
Reference	0.1507	0.1511	0.1526
Seed 0	-	0.1644	0.1638
Seed 1	-	-	0.1621

Table 3.2: LPIPS between diffusion outputs from different seeds on a NeuRAD output, with the diffusion being finetuned with various configurations, similar to Table 3.1. In (a) and (c), a finetuned diffusion model is used with LoRA-ranks 4 and 128 respectively. In (b) and (d), the models use a ControlNet model to condition on ray-information, as described in Section 3.4.3, once again with LoRA-ranks 4 and 128.

	Seed 0	Seed 1	Seed 2
Reference	0.1464	0.1473	0.1486
Seed 0	-	0.1221	0.1225
Seed 1	-	-	0.1215

(a) Diffusion UN4

	Seed 0	Seed 1	Seed 2
Reference	0.1467	0.1474	0.1462
Seed 0	-	0.1057	0.1045
Seed 1	-	-	0.1047

(b) Diffusion UN4CN4

	Seed 0	Seed 1	Seed 2
Reference	0.1928	0.1904	0.1920
Seed 0	-	0.0889	0.0878
Seed 1	-	-	0.0872

(c) Diffusion UN128

	Seed 0	Seed 1	Seed 2
Reference	0.1751	0.1723	0.1746
Seed 0	-	0.0895	0.0882
Seed 1	-	-	0.0876

(d) Diffusion UN128CN128

3.5 Results and Discussion

To evaluate the effectiveness of finetuning diffusion models in improving noisy NeRF outputs, we finetune Stable Diffusion using LoRA as described in Section 3.4.2. The model is trained on images from a single scene, focusing on front-facing camera images. For evaluation, two approaches are employed: first, the model is validated on a reserved subset of front-facing images not used during training. Second, the model's performance on NeRF outputs is tested using a pretrained NeuRAD model applied to front-left camera images from the scene. These images, not included in the NeRF models training data, exhibit poor reconstruction quality, allowing us to assess how effectively the diffusion model improves these noisy outputs. Metrics such as LPIPS, PSNR, and SSIM are used to evaluate the denoised outputs against the ground truth.

Figure 3.7 highlights qualitative differences across LoRA configurations. While varying LoRA ranks does not produce substantial qualitative differences, all configurations significantly reduce NeRF noise. However, a noticeable smoothing effect is observed in the outputs, particularly in background elements. This smoothing could hinder NeRF training, as volume rendering relies on maintaining consistent

and distinct pixel colors to ensure accurate reconstruction.

Quantitative results are presented in Table 3.3. Finetuned diffusion models consistently outperform the base model on general image-quality metrics across all measures. For noisy NeRF images, however, the base model achieves competitive LPIPS scores, indicating that perceptual similarity alone may not fully capture the improvements introduced by finetuning. Finetuned models with higher LoRA ranks consistently achieve better SSIM scores, suggesting improved structural similarity and robustness to noise. We also observe that conditioning the diffusion model on pose information via ControlNet, as described in Section 3.4.3, improves consistency across outputs. Despite this, pose conditioning does not mitigate the performance degradation caused by increased noise strength. These findings emphasize the importance of balancing noise strength to achieve effective denoising without sacrificing image clarity.

Figure 3.8 illustrates the effect of noise strength on the diffusion models outputs. At low noise strength (e.g., 0.1), the model effectively reduces NeRF-induced noise while preserving high image quality. However, as noise strength increases, the outputs become increasingly blurred, and image quality deteriorates, eventually surpassing the original noisy input in degradation. This degradation suggests that excessive noise during diffusion negatively impacts reconstruction quality, even with finetuning.

Table 3.3: An evaluation of finetuning the diffusion model across various configurations. Stable Diffusion 2.1 is fine-tuned and evaluated on Scene 001 from Pandaset. Specifically, we compare the base diffusion model (base), diffusion with LoRA weights (UN4, UN128) as described in Section 3.4.2, and diffusion with LoRA weights combined with ControlNet (UN4CN4, UN128CN128) as described in Section 3.4.3. For each configuration, LoRA-weights are tested at both rank 4 and rank 128, with each run representing a complete fine-tuning of the diffusion model weights. Performance is assessed using standard image-based metrics - LPIPS, PSNR, and SSIM - on the dataset. To evaluate generalization, the models are also tested on a reference dataset, as described in Section 3.5. Each metric is reported for both the ground truth image in the validation set and for the reference dataset.

Model	Noise Strength	LPIPS↓	PSNR↑	SSIM ↑	Ref LPIPS↓	Ref PSNR↑	Ref SSIM ↑
Base	0.1	0.18	23.43	0.69	**0.49**	20.45	0.66
Base	0.2	0.25	22.00	0.63	0.51	20.46	0.67
Base	0.3	0.31	21.00	0.59	0.52	20.49	0.67
UN4	0.1	0.14	24.40	0.72	**0.49**	20.70	0.67
UN4	0.2	0.16	23.66	0.69	0.51	20.71	0.67
UN4	0.3	0.18	23.09	0.66	0.53	20.60	0.67
UN128	0.1	0.11	25.41	**0.77**	0.52	20.66	**0.68**
UN128	0.2	0.13	24.91	0.74	0.58	20.47	**0.68**
UN128	0.3	0.14	24.55	0.72	0.64	19.96	**0.68**
UN4CN4	0.1	0.11	25.00	0.75	0.50	20.78	0.67
UN4CN4	0.2	0.13	24.45	0.72	0.56	20.63	0.67
UN4CN4	0.3	0.15	23.94	0.70	0.59	20.37	0.65
UN128CN128	0.1	**0.10**	**25.45**	**0.77**	0.52	**20.82**	**0.68**
UN128CN128	0.2	0.11	25.05	0.75	0.58	20.59	**0.68**
UN128CN128	0.3	0.12	24.93	0.75	0.66	19.52	**0.68**

(a) NeuRAD reference image

(b) Diffusion with UN4

(c) Diffusion with UN128

(d) Diffusion with UN4CN4

(e) Diffusion with UN128CN128

Figure 3.7: NeuRAD outputs which have been refined with Diffusion models finetuned with different configurations. (a) shows a poor NeuRAD render from a view in an area not covered in the training set. (b-c) show the image refined by a diffusion model which has been finetuned with LoRA rank 4 and 8 respectively, as mentioned in Section 3.4.2. (d-e) show the result of adding the ray as a conditioning signal, as outlined in Section 3.4.3.

(a) NeuRAD with shift:0, rotation: 45

(e) NeuRAD with shift:4, rotation: 0

(b) Noise Strength: 0.1

(f) Noise Strength: 0.1

(c) Noise Strength: 0.3

(g) Noise Strength: 0.3

(d) Noise Strength: 0.5

(h) Noise Strength: 0.5

Figure 3.8: Visual examination of the impact that noise strength has on diffusion the diffusion output. For comparison, two augmentations are applied to a selected reference view from the dataset. The first augmentation, shown in (a), is a 45 degrees rotation along the camera-local z-axis. The second augmentation, shown in (e), is a 4 meter shift along the camera-local x-axis. Images (b-d) and (f-h) show the diffusion output across the noise levels $\{0.1, 0.3, 0.5\}$, for pictures (a) and (e) respectively.

4

Augmenting Neural Rendering with Diffusion Models

This chapter outlines the integration of diffusion models with NeuRAD to enhance scene rendering quality in autonomous driving scenarios. The discussion begins with an overview of the pipeline design, detailing the steps required to combine the capabilities of diffusion models with NeuRADs neural rendering framework. Key implementation details, including modifications and optimizations introduced during integration, are presented to provide a comprehensive understanding of the process. We then evaluate the effectiveness of this integrated approach, showcasing the results achieved through experimental validation. Finally, the outcomes are analyzed to assess how well diffusion models address the limitations of NeuRAD, particularly in handling noisy renders and improving reconstruction quality.

4.1 ReconFusion

Our approach draws inspiration from the ReconFusion framework introduced by Wu et al. [17], which optimizes a NeRF model by minimizing two key losses. The first is the reconstruction loss, which measures the difference between NeRF-rendered images and a limited set of input images. This loss ensures that the model closely matches the original views and is consistent with the traditional NeRF training pipeline. The second is the sample loss, which compares renderings from randomly generated poses with predictions from a modified diffusion model conditioned on those poses.

In ReconFusion, the modified diffusion model incorporates conditioning signals that differ from the original NeRF implementation. Instead of text embeddings, it uses CLIP embeddings of the input images to encode high-level semantic information. Additionally, it employs PixelNeRF to extract deep feature maps from the input images. These feature maps are generated from all images in the dataset, and ray intersections are used to select relevant features from the views that overlap with the sampled rays. The selected features are combined to form a deep feature map that, along with the CLIP embeddings, serves as a conditioning signal for the diffusion model via cross-attention. The diffusion model then outputs refined samples, which are compared to NeRF renderings using an image-space loss, further enhancing the quality and consistency of NeRF-generated scenes.

However, the dynamic nature of our scenes poses challenges for directly applying the PixelNeRF component of ReconFusion. In the ReconFusion framework, scenes are assumed to be static, with objects maintaining fixed positions across frames. In contrast, our scenes contain dynamic objects, making it infeasible to use ray intersections at the current time to extract features from a different frame. This limitation prevents us from leveraging PixelNeRF as a feature extraction mechanism.

Additionally, our conditioning framework relies on ControlNet, which requires image-shaped conditioning signals. While CLIP embeddings provide high-level semantic information, they are not inherently image-shaped and are therefore unsuitable as direct inputs to ControlNet. Although we use CLIP embeddings via cross-attention, the limited parameter capacity of our low-rank fine-tuning approach may not adequately capture the required weight shifts to represent such embeddings effectively. These constraints necessitate alternative methods for integrating conditioning signals into the diffusion-NeRF pipeline.

4.2 Adapted Training Pipeline

Inspired by ReconFusion, we set up a training pipeline to use diffusion-improved NeRF renders as labels. To adapt it for dynamic scenes, we remove the PixelNeRF conditioning and CLIP embeddings, instead relying on the steps mentioned in chapter 3. The pipeline can be seen in Figure 4.1

Our training process is based on the original NeRF-training, as implemented in the open-source framework Nerfstudio [23]. The model is trained on a sequence of image-pose pairs from Pandaset, showing the trajectory of a car passing through a scene. In a given training step, a random $S \times S$ patch P is taken from the image, along with a ray bundle,

$$\mathbf{r}_{ij} = \begin{pmatrix} \mathbf{o}_{ij} \\ \mathbf{d}_{ij} \end{pmatrix}, (i,j) \in P, \tag{4.1}$$

consisting of the origin and direction of the ray passing through each pixel.

The original NeuRAD model is trained to estimate the color along a given ray,

$$C_{ij}(t) = F_\Theta(\mathbf{r}_{ij}, t). \tag{4.2}$$

In our case, we want to train the model in unseen parts of the scene. To accomplish this, we introduce an *augmentation* to the rays in the ray bundle. The augmentation is defined as a transformation α to the pose, resulting in the augmented ray bundle:

$$hat\mathbf{r}_{ij} = \alpha(\mathbf{r}_{ij}). \tag{4.3}$$

As in the original pipeline, this ray bundle is passed to a NeRF model which then estimates the augmented color values,

$$\hat{C}_{ij}(t) = F_\Theta(\hat{\mathbf{r}}_{ij}, t). \tag{4.4}$$

In NeuRAD, the estimated image $\mathbf{C}(t)$ is upsampled using a CNN to give it the same size as the target image.

Since we do not have a target for the augmented image, we use a pretrained DM, introduced in chapter 3, to improve the estimate estimation using the inherent generative capabilities as guidance for the NeuRAD. More concretely, we use a pretrained VAE to encode the estimated image patch into a latent space, resulting in a latent embedding,

$$z_P(t) = \mathcal{E}\left(\hat{C}_P(t)\right). \tag{4.5}$$

To this latent variable, Gaussian noise is then added according to a forward diffusion process, D, resulting a noisy latent variable,

$$\tilde{z}_P = D(z_P). \tag{4.6}$$

In our case, we use Denoising Diffusion Implicit Model (DDIM) [44], as implemented in the HuggingFace Diffusers library [45], as our scheduling algorithm.

At this stage, cross-attention is used to combine the noisy latent with additional information. Since the original Stable Diffusion model is trained in a text-to-image context, the original conditioning consists of CLIP embeddings of textual input. In our case, we provide a constant CLIP embedding of an empty prompt, effectively relying solely on the noisy latent.

Then, the noise is gradually removed from the latent variable using the diffusion model, gradually transforming it according to the conditional training distribution of the diffusion model. This results in the denoised latent,

$$\bar{z}_P = D^{-1}(\tilde{z}_p), \tag{4.7}$$

where D^{-1} is implemented as a timeconditioned denoising UNet. When using the Controlnet-based model, we use the ray bundle itself as a conditioning signal, embedding it using Controlnet before adding it to the decoder layers of the UNet.

Once the latent is denoised, it is transformed back to the image space using the decoder of the VAE. This is then used as a target for the NeRF-model by comparing their outputs with a mean-square error loss, resulting in an *augmentation loss*. To prevent catastrophic forgetting, we include the original losses from NeRF pipeline, and simply add the augmentation loss weighted by a loss multiplier [46].

4.3 Results and Discussion

In this section, we evaluate the integration of diffusion models with NeuRAD and analyze their performance across various configurations. The models are assessed using four primary metrics: Learned Perceptual Image Patch Similarity (LPIPS), Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Frechet Inception Distance (FID). These metrics enable a comprehensive evaluation of image quality, comparing the outputs of different models against ground truth data and assessing generalization to unseen views. LPIPS measures perceptual similarity, PSNR evaluates pixel-level accuracy, SSIM assesses structural similarity, and FID quantifies the distributional distance between two sets of images. Unlike

other metrics, FID does not require a ground truth image, making it particularly useful for evaluating shifted perspectives without matching ground truth.

To further assess generalization, we evaluate FID at three levels of lateral shifts: 0 meters (no shift), 4 meters, and 8 meters to the left. This shifted FID provides insights into the model's ability to render novel views and maintain spatial coherence in unseen regions. Shorthand Notation

To simplify references to the models used in the experiments, we adopt the following shorthand:

- BN: Just NeuRAD, without diffusion.
- BD: NeuRAD, trained with a non-finetuned DM.
- UN4 and UN128: NeuRAD, trained with a DM which have been fine-tuned using LoRA at ranks 4 and 128, respectively.
- UN4CN4 and UN128CN128: NeuRAD, trained with a DM which has been finetuned with LoRA at ranks 4 and 128, respectively, and which employs pose-conditioning.

These variations allow for direct comparisons between the baseline NeuRAD model and models augmented with diffusion and conditioning techniques.

To determine a good set of parameters for further investigation, we run a parameter sweep on scene 001. Table 4.1 presents the results of experiments investigating the effects of three key parameters on the performance of the UN128CN128 model:

- Augment Loss Multiplication: Determines the relative weight of the diffusion loss in the optimization process. Higher values increase the influence of the diffusion model but risk overwhelming NeuRADs training.
- Augment Phase Step: Specifies the training step at which the diffusion model is introduced. Introducing diffusion after NeuRAD achieves stable reconstructions prevents suboptimal early-stage training.
- Noise Strength: Controls the level of noise added during diffusion. Effective noise levels must balance addressing NeuRAD artifacts with preserving image details.

Results indicate that excessive noise strength (e.g., 0.3) degrades performance, overwhelming the diffusion models capacity to denoise effectively. The optimal noise strength appears to be 0.1, striking a balance between denoising capacity and preserving NeuRADs outputs.

Next, we evaluate the performance with different diffusion models. In Table 4.2, we report model performance across three scenes: 001, 016, and 053. Results reveal inconsistencies, with no single model consistently outperforming others across all scenes. While UN128CN128 achieves the best performance on Scene 001, it does not generalize as effectively to the other scenes. Interestingly, the base NeuRAD model scores the best FID at 8 meters in one scenario, suggesting that in certain cases, integrating diffusion models may degrade performance.

Table 4.1: Evaluation results using the adapted pipeline with various parameter settings on Scene 001. Includes metrics for models evaluated with augmentation loss multipliers of 20, 40, and 60, augmentation introduced at step 0 or step 20000, and noise strengths of 0.1, 0.2, and 0.3. Each experiment reports LPIPS, PSNR, SSIM, and FID metrics (FID_0, FID_4, FID_8). For LPIPS and FID, lower values indicate better performance, while for PSNR and SSIM, higher values indicate better performance.

Aug Loss Mult	Aug Phase Step	Noise Strength	LPIPS↓	PSNR↑	SSIM↑	FID_0↓	FID_4↓	FID_8↓
20	0	0.10	0.19	26.34	0.77	**27.10**	136.06	**166.22**
20	20000	0.10	0.19	**26.52**	0.77	27.83	**134.18**	169.09
40	0	0.10	0.19	26.50	0.77	29.04	139.17	174.15
40	20000	0.10	0.19	26.42	0.77	28.99	137.33	167.27
60	0	0.10	0.19	26.40	0.77	28.44	137.92	170.18
60	20000	0.10	0.19	26.38	0.77	29.05	137.55	169.45
20	0	0.20	0.19	26.37	0.77	29.64	137.23	170.52
20	20000	0.20	0.19	26.49	0.77	30.89	140.49	174.56
40	0	0.20	**0.18**	26.43	0.77	28.54	134.64	168.20
40	20000	0.20	0.19	26.48	0.77	28.69	136.08	166.55
60	0	0.20	0.19	26.39	0.77	29.84	135.36	168.69
60	20000	0.20	0.19	26.41	0.77	27.92	135.65	171.78
20	0	0.30	0.19	26.48	0.77	29.23	139.62	170.96
20	20000	0.30	0.19	26.44	0.77	29.53	141.30	175.65
40	0	0.30	0.19	26.36	0.77	29.09	136.49	171.69
40	20000	0.30	0.19	26.43	0.77	27.64	136.41	169.20
60	0	0.30	0.19	26.51	0.77	28.03	134.87	173.72
60	20000	0.30	0.19	26.47	0.77	29.18	136.64	166.75

Figure 4.2 illustrates the rendered outputs for various models using the front-left camera pose. Subfigure (a) shows the ground truth image, while subfigure (b) highlights the limitations of the base NeuRAD model, with artifacts and poor reconstruction in most regions except the right-hand side, which corresponds to the training view. Subsequent subfigures (c-g) depict the results of integrating diffusion models with NeuRAD. While fine-tuned diffusion models slightly improve the rendered outputs, the changes are minimal and fail to significantly enhance overall image quality. These qualitative observations align with the quantitative metrics reported in Table 4.2, indicating that diffusion models contribute limited benefits in this context.

The results suggest that while diffusion models, particularly when finetuned or using pose-conditioning, offer some improvements in handling noisy NeRF renders, these benefits are not consistent or substantial. High variability in performance across scenes and limited improvements in qualitative results highlight the challenges of effectively integrating diffusion models into NeuRAD. Further work is required to refine this approach, particularly in addressing the smoothing effects observed and improving generalization to unseen views.

Table 4.2: Evaluation results using the adapted pipeline with various parameters. Includes metrics for NeuRAD finetuned with: No diffusion, non-finetuned diffusion, diffusion finetuned with rank 4, diffusion finetuned with rank 128, diffusion with ControlNet finetuned with rank 4, diffusion with ControlNet finetuned with rank 128. Each experiment is done with noise strength 0.1, augmentation introduced at step 20000, and 40 as augmentation multiplier. FID values are reported at a lateral shift of 0, 4, and 8 meters respectively.

Model	LPIPS↓	PSNR↑	SSIM↑	FID_0↓	FID_4↓	FID_8↓
BN	0.19	26.47	0.77	29.10	138.51	168.96
BD	0.19	26.45	0.77	29.65	139.83	169.84
UN4	0.19	**26.52**	0.77	27.93	138.24	173.21
UN128	0.19	26.39	0.77	29.89	141.20	171.43
UN4CN4	0.19	26.45	0.77	28.56	136.32	172.82
UN128CN128	0.19	26.46	0.77	**27.27**	**135.33**	**167.93**

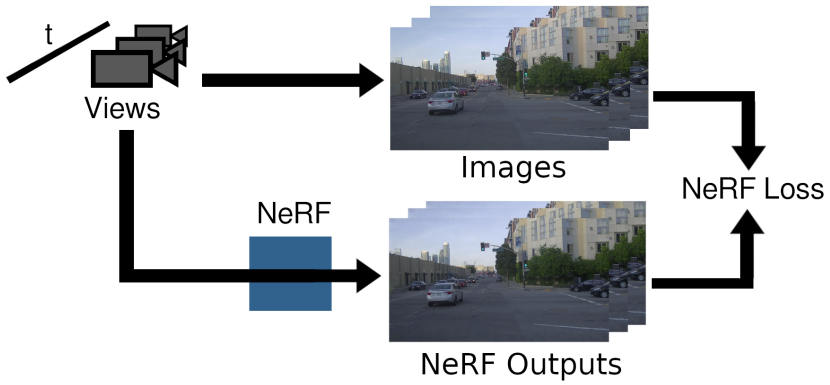
(a) Evalution on scene 001

Model	LPIPS↓	PSNR↑	SSIM↑	FID_0↓	FID_4↓	FID_8↓
BN	0.15	26.89	0.86	25.17	125.25	**210.98**
BD	0.15	26.75	0.85	24.64	125.70	217.76
UN4	0.20	26.04	0.83	38.40	134.54	216.99
UN128	**0.14**	**26.99**	**0.86**	**21.99**	121.03	216.13
UN4CN4	0.17	26.43	0.85	28.54	**120.68**	212.35
UN128CN128	**0.14**	26.85	**0.86**	24.57	126.70	212.92

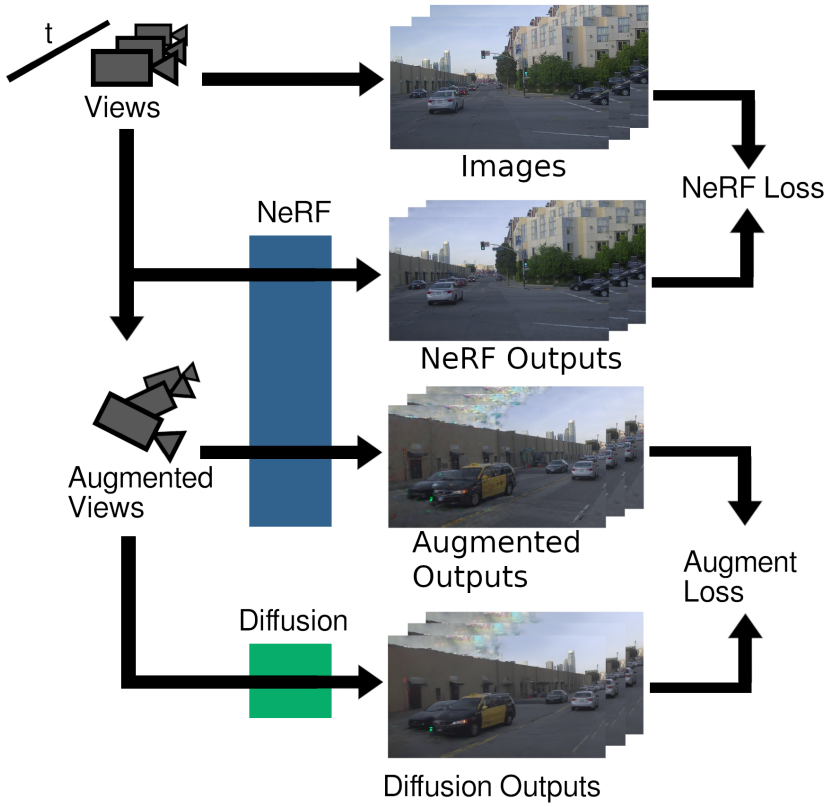
(b) Evalution on scene 016

Model	LPIPS↓	PSNR↑	SSIM↑	FID_0↓	FID_4↓	FID_8↓
BN	0.16	**27.01**	0.81	26.42	126.78	**302.29**
BD	0.16	26.88	0.81	25.55	126.70	316.85
UN4	0.16	26.95	0.81	26.37	127.57	315.08
UN128	0.16	26.79	0.81	25.70	**125.26**	306.55
UN4CN4	0.16	26.85	0.81	25.85	126.68	313.16
UN128CN128	0.16	26.91	0.81	**25.07**	127.68	313.63

(c) Evalution on scene 053

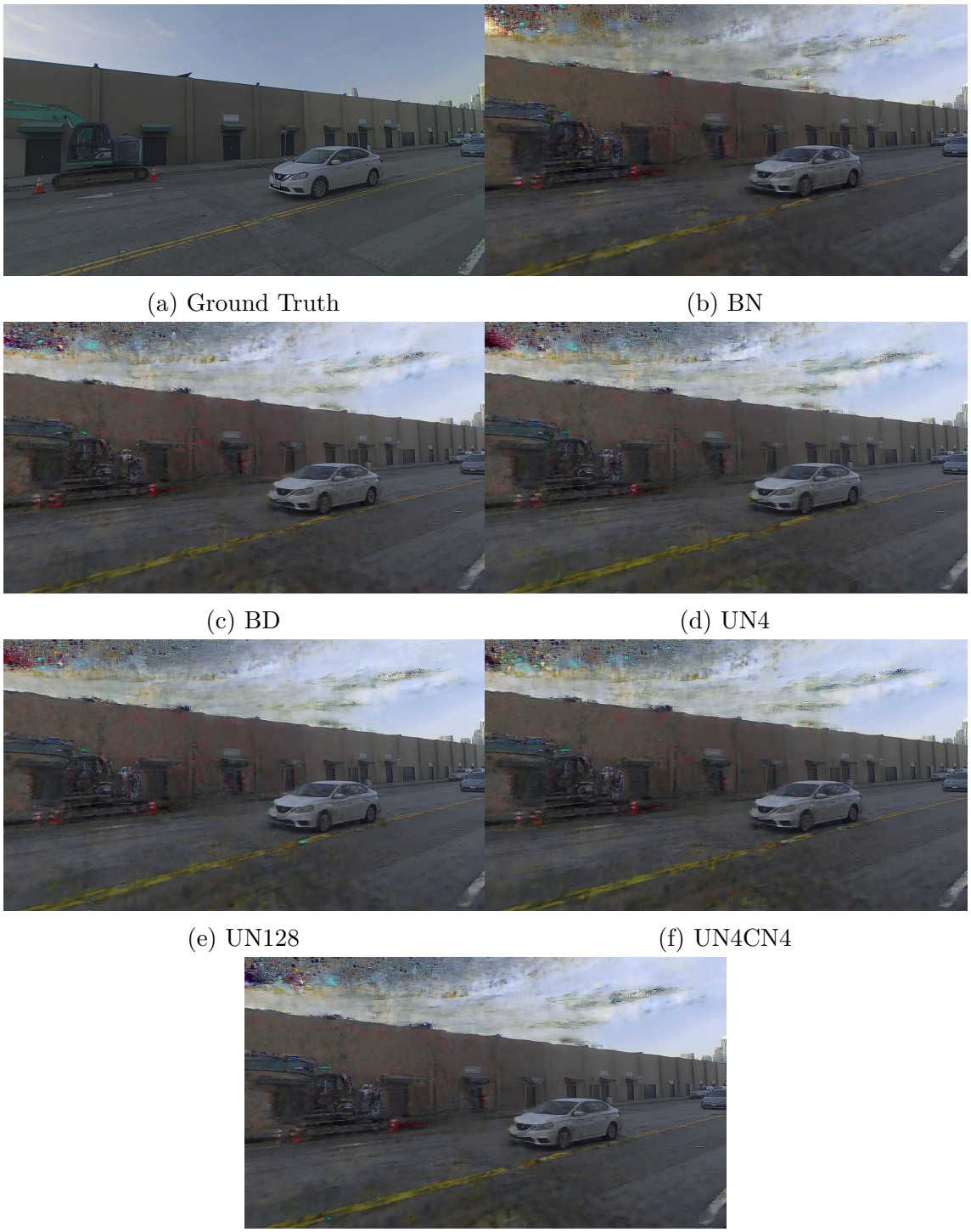


(a) The basic NeRF pipeline in an autonomous driving setting. Each sequence is a set of view-image pairs unfolding over time as the ego-vehicle moves through a scene. The images are fed into a NeRF which predicts the color values of each pixel. This NeRF is then compared to the output as a recreation loss, resulting in a NeRF loss. Note that certain models may add additional loss functions, such as the ray drop loss in NeuRAD.



(b) Our pipeline, which extends the original NeRF pipeline by augmenting the views in original dataset with a random shift or rotation. Since these augmented views do not have a corresponding image in the dataset, we use a diffusion model to enhance the noisy output from the NeRF. This enhanced output is then used as a target for the NeRF model, using the difference between the two outputs as an additional loss.

Figure 4.1: Our pipeline compared to the original NeRF pipeline.



(a) Ground Truth (b) BN

(c) BD (d) UN4

(e) UN128 (f) UN4CN4

(g) UN128CN128

Figure 4.2: NeuRAD renders compared across different model configurations. (a) shows the ground truth image captured by the car's front-left camera. (b) depicts the NeuRAD output without any diffusion model finetuning. (c) illustrates the result of applying a diffusion model not finetuned to the task. (d-e) show renders refined using diffusion models finetuned with LoRA rank 4 and 128, respectively, as described in Section 3.4.2. (f-g) present results from diffusion models incorporating ray conditioning via ControlNet, with LoRA ranks 4 and 128, as outlined in Section 3.4.3.

5

Conclusion

This thesis investigates enhancing NeuRADs performance by training it to handle new poses not included during its original training. The objective is to improve the models ability to render unseen parts of the scene, addressing the challenge of generalizing to novel viewpoints. Our contributions include fine-tuning Stable Diffusion for AD scenes using recent techniques and attempting to adapt the ReconFusion pipeline for improved neural rendering of rare views.

To adapt the diffusion model for specific scenes, we fine-tune it to incorporate domain-specific knowledge. This approach aims to generate images more aligned with the scenes characteristics, enhancing coherence and accuracy. To manage computational costs, we employ LoRA, significantly reducing memory and training requirements. Conditional signals are also introduced to guide the diffusion process.

Despite these efforts, the experiments did not yield significant improvements to NeuRAD. This may be due to insufficient LoRA rank to capture complex scene information or the limited conditioning signals, with ray bundles alone proving inadequate. Future work could explore incorporating additional signals, such as depth or temporal information, and training base models across multiple AD scenes to improve generalization.

The limited scope of the current experiments restricts broader conclusions. Future work should replicate these studies across more scenes and parameters to account for variability and better evaluate the approach's effectiveness.

Adapting ReconFusion for AD scenarios remains a promising direction. Extending PixelNeRF to handle temporal data could improve rendering in dynamic environments by capturing changes in objects and scenes over time. Additionally, masking dynamic elements using dataset annotations, such as those in Pandaset, could allow ReconFusion to focus on reconstructing static components. However, this would require substantial computational resources for fine-tuning to ensure accurate reconstruction of static elements while minimizing the impact of dynamic ones.

Bibliography

[1] J. Baruch, "Steer driverless cars towards full automation," *Nature*, vol. 536, no. 7615, pp. 127–127, 2016.

[2] M. R. Bachute and J. M. Subhedar, "Autonomous driving architectures: Insights of machine learning and deep learning algorithms," *Machine Learning with Applications*, vol. 6, p. 100 164, 2021.

[3] H. Choi, C. Crump, C. Duriez, *et al.*, "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," en, *Proc. Natl. Acad. Sci. U. S. A.*, vol. 118, no. 1, e1907856118, Jan. 2021.

[4] A. Tonderski, C. Lindström, G. Hess, W. Ljungbergh, L. Svensson, and C. Petersson, "Neurad: Neural rendering for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 14 895–14 904.

[5] Z. Yang, Y. Chen, J. Wang, *et al.*, "Unisim: A neural closed-loop sensor simulator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1389–1399.

[6] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang, "Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 634–21 643.

[7] H. Turki, J. Y. Zhang, F. Ferroni, and D. Ramanan, "Suds: Scalable urban dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 375–12 385.

[8] Y. Yan, H. Lin, C. Zhou, *et al.*, "Street gaussians for modeling dynamic urban scenes," in *ECCV*, 2024.

[9] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide, "Neural scene graphs for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2856–2865.

[10] T. Fischer, L. Porzi, S. R. Bulo, M. Pollefeys, and P. Kontschieder, "Multi-level neural scene graphs for dynamic urban environments," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 125–21 135.

[11] W. Ljungbergh, A. Tonderski, J. Johnander, *et al.*, "Neuroncap: Photorealistic closed-loop safety testing for autonomous driving," in *European Conference on Computer Vision*, Springer, 2025, pp. 161–177.

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[13] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM transactions on graphics (TOG)*, vol. 41, no. 4, pp. 1–15, 2022.

[14] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 729–19 739.

[15] L. Song, A. Chen, Z. Li, *et al.*, "Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 5, pp. 2732–2742, 2023.

[16] Z. Zhou and S. Tulsiani, "Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 588–12 597.

[17] R. Wu, B. Mildenhall, P. Henzler, *et al.*, "Reconfusion: 3d reconstruction with diffusion priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 551–21 561.

[18] C. Deng, C. Jiang, C. R. Qi, *et al.*, "Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 637–20 647.

[19] D. Watson, W. Chan, R. M. Brualla, J. Ho, A. Tagliasacchi, and M. Norouzi, "Novel view synthesis with diffusion models," in *The Eleventh International Conference on Learning Representations*, 2023.

[20] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3836–3847.

[21] E. J. Hu, yelong shen, P. Wallis, *et al.*, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[22] Z. Wu, T. Liu, L. Luo, *et al.*, "Mars: An instance-aware, modular and realistic simulator for autonomous driving," in *CAAI International Conference on Artificial Intelligence*, Springer, 2023, pp. 3–15.

[23] M. Tancik, E. Weber, E. Ng, *et al.*, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–12.

[24] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[25] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering.," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[26] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06, Boston, Massachusetts: Association for Computing Machinery, 2006, pp. 835–

846, ISBN: 1595933646. DOI: 10.1145/1179352.1141964. [Online]. Available: https://doi.org/10.1145/1179352.1141964.

[27] H. Kim, G. Lee, Y. Choi, J.-H. Kim, and J.-Y. Zhu, "3d-aware blending with generative nerfs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 906–22 918.

[28] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 453–11 464.

[29] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 154–20 166, 2020.

[30] T. Salimans, I. Goodfellow, W. Zaremba, *et al.*, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016.

[31] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[32] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[33] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "Pixelnerf: Neural radiance fields from one or few images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4578–4587.

[34] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[35] H. Lyu, N. Sha, S. Qin, M. Yan, Y. Xie, and R. Wang, "Advances in neural information processing systems," *Advances in neural information processing systems*, vol. 32, 2019.

[36] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 415–419. DOI: 10.1109/SMC.2018.00080.

[37] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[38] B. Park, S. Yu, and J. Jeong, "Densely connected hierarchical network for image denoising," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2104–2113. DOI: 10.1109/CVPRW.2019.00263.

[39] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *The 2nd International Conference on Learning Representations*, ICLR), 2013.

[40] A. van den Oord, O. Vinyals, and k. kavukcuoglu koray, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[41] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[42] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 873–12 883.

[43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 10 684–10 695.

[44] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

[45] P. von Platen, S. Patil, A. Lozhkov, *et al.*, *Diffusers: State-of-the-art diffusion models*, version 0.12.1, 2023.

[46] Z. Wang, E. Yang, L. Shen, and H. Huang, "A comprehensive survey of forgetting in deep learning beyond continual learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.