

Institute for Visualization and Interactive Systems

University of Stuttgart  
Universitätsstraße 38  
D-70569 Stuttgart

Masterarbeit

# **Investigating Challenges in Generalizing Neural Radiance Fields with Learned Scene Priors**

Jonas Geiselhart

**Course of Study:** Artificial Intelligence and Data Science

**Examiner:** Prof. Dr. Dieter Schmalstieg

**Supervisor:** Dr. Shohei Mori

**Commenced:** 17. December 2024

**Completed:** 17. June 2025



---

## Abstract

This thesis investigates methods to generalize neural radiance fields across several different 3D-Scenes. Unlike prevailing approaches that emphasize more fine grained priors on ray or sample positions - often combined with classical 3D spatial (neural) processing, this work explores the use of implicit deep scene embeddings as prior to a generalized neural radiance field for scene rendering.

This work provides the theoretical groundwork for transitioning gradually from per scene retraining to a more perceiving network capable of extracting scene geometries by analyzing images and successfully building good latent representations.

In practical research the framework is implemented and analyzed. Here several key issues in the conceptualization are found and analyzed, that must be addressed in training process and model redesign.

Overall this thesis outlines a potential path toward scene-generalized NeRFs and highlights new issues that emerge through this shift in research focus.

## Kurzfassung

In dieser Arbeit werden Methoden zur Generalisierung von NeRFs (Neural Radiance fields) über verschiedene 3D-Szenen hinweg untersucht. Im Gegensatz zu vorherrschenden Ansätzen, die feinere Priors für Strahlen- oder Sample-Positionen betonen - oft in Kombination mit klassischer räumlicher 3D-Verarbeitung - erforscht diese Arbeit die Verwendung von impliziten abstrakten Encodings der Szene als Prior für ein generalisiertes Neutrales Netz für das Szenen-Rendering.

Daher liefert diese Arbeit die theoretische Grundlage für den schrittweisen Übergang von dem szenenspezifisch gelerntem zu einem wahrnehmenden Netzwerk, das in der Lage ist, Szenengeometrien zu extrahieren, indem es Bilder analysiert und eine akkurate latente Repräsentationen extrahiert.

Im praktischen Teil wird das Framework implementiert und analysiert. Hier werden mehrere Schlüsselprobleme in dem Konzept gefunden und analysiert, die im Trainingsprozess und bei der Neugestaltung des Modells angegangen werden müssen.

Insgesamt skizziert diese Arbeit einen möglichen Weg zu szenengeneralisierten NeRFs und hebt neue Probleme hervor, die durch diese Verlagerung des Forschungsschwerpunkts entstehen.



# Contents

<b>1</b>	<b>Introduction</b>	<b>11</b>
<b>2</b>	<b>Preliminaries</b>	<b>13</b>
2.1	NeRFs and Optimizations . . . . .	13
2.2	Other Preliminaries . . . . .	16
<b>3</b>	<b>Related Works</b>	<b>19</b>
3.1	Introduction of pixelwise Priors . . . . .	19
3.2	Leveraging Multiple Images - General Radiance Fields . . . . .	20
3.3	Moving from Pixels to Positions using explicit Differences . . . . .	20
3.4	Explicit Discretization of Features with 3D-CNNs . . . . .	21
3.5	Separation of Geometry and Appearance . . . . .	22
3.6	Selective Attention to specific Sampling Points . . . . .	23
3.7	Reducing Discretization Relicts with Local to Global Feature Fusion . . . . .	23
3.8	Improving Raywise Priors with Occlusion Computation . . . . .	23
3.9	Other 3D Reconstruction Methods . . . . .	24
3.10	Overview . . . . .	25
<b>4</b>	<b>Datasets</b>	<b>29</b>
4.1	Datasets . . . . .	29
4.2	Preprocessing of Datasets . . . . .	31
<b>5</b>	<b>Training &amp; Network Architecture</b>	<b>33</b>
5.1	Conceptualization . . . . .	33
5.2	Predictive Coding - Applying free energy . . . . .	35
5.3	Architectures . . . . .	35
5.4	Training . . . . .	38
<b>6</b>	<b>Experiments</b>	<b>41</b>
6.1	Improving Distinguishability of Scenes . . . . .	41
6.2	Predicting the correct geometries. . . . .	43
<b>7</b>	<b>Results and Future Work</b>	<b>51</b>
7.1	Results . . . . .	51
7.2	Future Work . . . . .	52
	<b>Bibliography</b>	<b>53</b>
<b>A</b>	<b>Appendix</b>	<b>57</b>



## List of Figures

2.1	Pipeline as illustrated in the original NeRF proposal by Mildenhall et al. [21]. In step (a) we can see how the queried points get selected, (b) illustrates how the results (in terms of color and density per point may look like), (c) demonstrates the volumetric rendering and (d) the loss generation using MSE-loss. . . . .	14
2.2	Illustration of the Cone tracing examples from MipNerf [1] . . . . .	15
3.1	The prior selection in pixelNeRF (from the original paper) . . . . .	20
3.2	The ContraNeRF generalization pipeline as illustrated in the original paper. . . .	21
3.3	Framework for generalizable NeRF-constructions . . . . .	26
4.1	Different plots showing the camera layout of the first scene from the LLFF Dataset (top), Blender Dataset (middle left), DTU Dataset (middle right), ScanNet (bottom left) and SceneNet (bottom right) for reference of real life natural scene datasets. One can see the dome-like layout of the Blender data, as well as the Forward facing camera layout of DTU and LLFF, with ScanNet and SceneNet as references, where there are more complex camera paths with different directions and less coverage from different perspectives . . . . .	30
4.2	Different examples of Blender Dataset scenes. . . . .	31
4.3	Different examples of DTU-MVS Dataset scenes. . . . .	31
4.4	Different examples of LLFF Dataset scenes. . . . .	31
5.1	Free energy Principle of Recognition . . . . .	33
5.2	The three different scene models that I have tried in my initial research. Grey instances are optional parts. If not specified otherwise the layers are fully connected. . . . .	37
5.3	The Render Model, the scene encoding is described closer in chapter 6. . . . .	38
6.1	Illustration of the new latent component . . . . .	42
6.2	Training progress similar to Tabelle 6.1, but with eight scenes to train, rather than two. . . . .	45
6.3	Training loss (left) and Validation loss (right) with respect to a different number of scenes. . . . .	48
6.4	Comparison across three combinations of validations and training data to indicate causes of overfitting. . . . .	49
6.5	Different setups to test the position of overfitting. . . . .	49
6.6	Comparison of model renderings when prompted with inconsistent latent . . . . .	49
6.7	Comparison of model renderings when prompted with no scene encoding. . . . .	50
A.1	The validation views of the lego scene from blender . . . . .	57





# List of Tables

3.1	Comparison of NeRF Methods . . . . .	27
6.1	Training progress as illustrated by images during the training process, each time a training view is rendered (middle column) that corresponds to the ground truth, and a validation view is also rendered (right column). This illustrates a two scene training process. (Epochs 20 to 120). In this example the training view scene is not the same as the validation view. . . . .	44
6.2	Training progress as illustrated by images during the training process, each time a training view is rendered (middle column) that corresponds to the ground truth, and a validation view is also rendered (right column, not the same scene as GT and training scene). This illustrates a two scene training process. (Epochs 140 to 240)	46
A.1	Training progress shown every 40 epochs, using the Reduced model, 500 epochs, and 8 Scenes (only two are shown in the images). While the Training View, Depth View and Ground Truth correspond to each other, the validation view is rendered from the fern scene (see Figure 4.2 for a reference image). Interestingly here the geometries get extracted, even though the quality is rather bad and the extraction is unstable during the training process. . . . .	58



# 1 Introduction

Since its inception, computer vision has been one of the most extensively explored domains in deep learning and artificial intelligence. The visual nature of images, their interpretability, and the abundance of online training data were key factors that made 2D image-based training especially compelling. In contrast, 3D computer vision is comparatively underexplored, which is primarily due to two factors (i) the limited availability of annotated real-world 3D datasets, and (ii) the increased computational demands introduced by the additional spatial dimension. Several strategies have been developed to handle these challenges. One common approach is to represent the scene as a point cloud—discrete surface points typically obtained via LiDAR or radar. Another method is to discretize a 3D model into voxels and process them, for example, with a 3D-CNN. Other techniques include occupancy networks that predict signed distances to surfaces or classical approaches for 3D Stereo Reconstruction to generate a data mesh.

While much of the research is focused on 2D-images, we expect 3D vision to take a more central approach, as this is a more powerful way for computers to model real-life scenes directly. The approach taken in this work should be a first step to generate holistic 3D-model representation, that aims to generate context that can be efficiently processed in future applications with higher fidelity. This thesis focuses on the task of novel view synthesis by Neural Radiance Fields (NeRFs). These deep models avoid the typical volumetric discretization by directly interpreting occupancy (density) and appearance (radiance) as a continuous function of the space itself. This function can be learned from images by volumetric rendering, which is a naturally differentiable process.

Unfortunately NeRFs currently have several drawbacks that make their use in real life systems unpractical. Most notably the geometry of a scene is encoded in the weights of a prediction model, which leads to a prohibitive computational overhead for most practical uses, since the network has to be retrained for each new scene.

Additionally the level of detail is limited by the complexity of the scene as well as the number and distribution of views around the scene that should be rendered.

Despite these current restrictions NeRFs are still one of the most promising candidates to further train artificial models to understand 3D data, since they only require 2D images and relative image positions in order to train — no additional annotations. Their training process is inherently self-supervised and yields state of the art results in quality, most notably correctly learning the geometry and radiance of objects and scenes. This work is focused on mitigating some of these restrictions by introducing a second network, the Recognition Network, that is focused on generating an expressive prior from the scene that can be used for rendering a scene, therefore moving towards decoupling the tasks of scene understanding and novel view rendering. This approach also opens the door to generating meaningful latent and implicit scene representations, which could be leveraged in future tasks such as scene modification, autonomous navigation, or scene classification.

The main objective of this thesis is to find out if neural radiance fields can be developed from a single geometry memory network to a multi-scene recall network, that is potentially able to recognize new scenes and build a meaningful deep representation of the geometry from a few images without retraining. This feature representation should also be able to be rendered. While it is obvious that,

this goal is too far reaching for a master thesis, it would be interesting to construct some proof of feasibility for this idea.

More concretely I investigate the approaches that are made to generalize NeRFs in chapter 3, then I investigate how a framework to train generalizable NeRFs has to be trained in order to generate scenewise priors. This is done in chapter 5 and later these approaches are tested in chapter 6.

Although Neural Radiance Fields are still in development, it is clear, that they have enormous potential in the areas of robotics, computer vision, mediated reality and visual effects.

Within robotics, NeRFs can be used for scene reconstruction, segmentation and navigational tasks (SLAM). Wang et al. provide an extensive survey on the different applications in this area [30]. Furthermore there are considerable efforts to make NeRFs practical for mediated reality applications, such as placing virtual objects in a real life scene or rendering whole scenes in virtual reality, with real life components [16, 35]. Other interesting applications in NeRFs include scene editing or medical Imaging [18, 38].

## 2 Preliminaries

Typically the NeRF research is sorted in one of two ways. The first approach is chronological using the initially proposed NeRF as baseline and from there on sorting by time of publication into several timeframes between „milestones“. The second approach is based on the specific optimization goal - typically improving reconstruction quality, reducing training time, minimizing dataset requirements, or enhancing generative capabilities - as done for example in [10].

### 2.1 NeRFs and Optimizations

NeRFs were introduced by Mildenhall et al. [21] as a method to facilitate novel view synthesis without relying on spatial discretization. Crucially, the network is now queried with coordinates and directions and predicts densities as well as colors from it. This approach bridges the representation gap given by standard discretized machine learning representations - such as voxel grid or point clouds. It enables the network to be trained in a self-supervised manner with only images and their positions given and it generalizes very well to photometric properties and interpolate dense geometries from only images.

NeRFs typically are represented by a function representing the output of the neural network:

$$(R, G, B, \sigma) = NeRF(x, y, z, \phi, \gamma)$$

with  $R, G, B$  being the radiance values and  $\sigma$  the density of  $x, y, z$  when looked at from directions  $\phi$  and  $\gamma$ . This can now be rendered to a final color via simplified (and discretized) volume rendering:

$$\begin{aligned} RGB_i &= \sum_{ij}^N w_{ij} \cdot c_{ij} \\ &= \sum_{ij}^N T_{ij} \cdot \alpha_{ij} \cdot c_{ij} \end{aligned}$$

with  $\alpha$  being the opacity for each point:

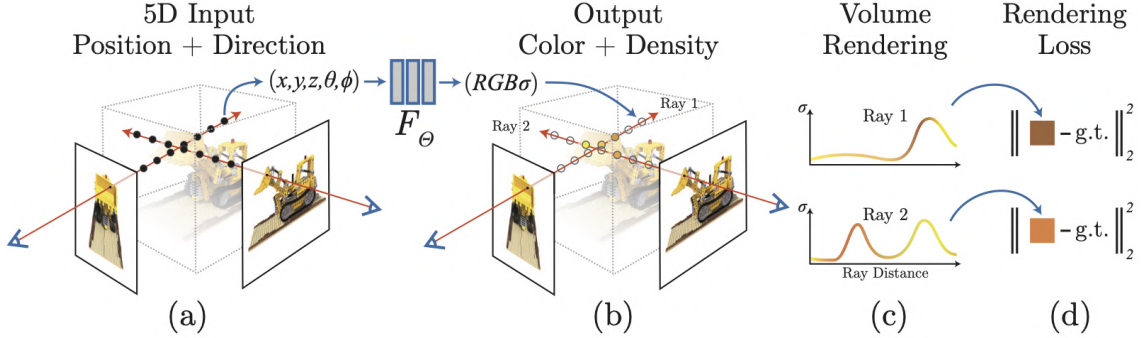
$$\alpha_{ij} = 1 - \exp(-\sigma \cdot \Delta_{ij})$$

with  $\Delta_{ij}$  being the differences between the samples) and  $T_{ij}$  being the transmittance of the point, computed as follows:

$$T_{ij} = \exp\left(-\sum_{k=1}^{j-1} \sigma_{ik} \cdot \Delta_{ik}\right)$$

As this process only contains differentiable operators we can run a full backwards pass to tune the network weights.

### 2.1.1 Vanilla-NeRF



**Figure 2.1:** Pipeline as illustrated in the original NeRF proposal by Mildenhall et al. [21]. In step (a) we can see how the queried points get selected, (b) illustrates how the results (in terms of color and density per point may look like), (c) demonstrates the volumetric rendering and (d) the loss generation using MSE-loss.

The original NeRF implementation [21] includes several key optimizations to accelerate training and improve convergence. Using these strategies Mildenhall et al. provide a good fundament to reach convergence to a detailed model within 100–300k iterations, using a batch size of 4096 rays per step. Figure 2.1 provides an illustration of the classical NeRF pipeline.

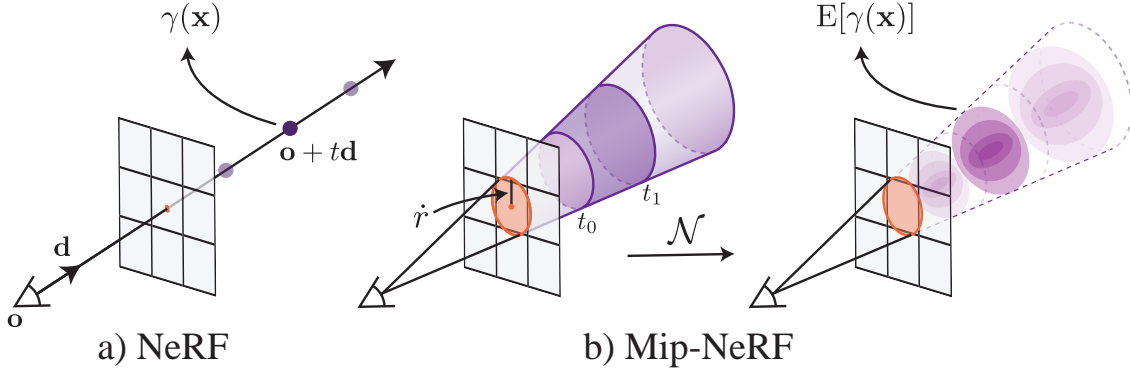
**Hierarchical Ray Sampling** is applied, to increase convergence and sampling efficiency. Typically 64 - 256 samples are drawn per ray. In the first coarse stage of computation during the volume rendering weights are computed that indicate how much each sample contributes to the final color prediction. These weights form a piecewise constant probability density function (PDF) over depth (z-values), which is then used to resample points in regions of higher importance—typically near object boundaries or high-density areas. Typically the fine and the coarse loss are used in the backwards pass (either the same weighting or a lower coarse loss weighting). The weights for this probability function can be computed as  $\sigma * \exp(-\sigma * \Delta)$ .

Also **Positional Encoding (PE)** is applied to both spatial positions as well as the camera directions (parameterized by azimuth and elevation). This encoding enables a coarse-to-fine learning strategy, where the embedding’s distance reflects the spatial distance between points in 3D space. Therefore improving robustness when learning new points. Without positional encoding, high-frequency scene details often appear blurred, as the network struggles to learn precise variations in radiance. Tancik et al. [27] demonstrated that Fourier Features as positional encoding significantly improved the Networks ability to convergence. Although alternative encodings (e.g. random gaussian encoding) improved results in the experimental case, PE via Fourier features remains the standard for NeRFs, most likely due to their training time simplicity as well as their robust performance gain.

### 2.1.2 Lifting ray samples into 3D regions

The introduction of MipNeRF by Barron et al. [1] in 2021, marked the second generation of NeRFs. This new type especially incorporates significant optimizations to improve training efficiency. This is mainly done via the two following optimizations.

**Cone Tracing** models ray-samples not only as points and their properties, but predicts regions that contribute to the pixel coloring. Instead of modeling a ray as a one dimensional line, it is more



**Figure 2.2:** Illustration of the Cone tracing examples from MipNerf [1]

useful to model it as a cone with apex at the ray origin (= focal point of the image) and has the diameter of the footprint of the pixel at the image plane, as this is a way to capture and compute the whole extend of the objects, that contribute to the pixel. The illustration from the original paper is given in Figure 2.2. For training, this involves modeling each ray sample as a 3D Gaussian distribution, with the mean at the sample point and variance corresponding to the cone's footprint. This modeling approach is augmented by the second optimization, using **Integrated Positional Encoding (IPE)**, which encodes entire regions instead of individual points. This is done via computing the expectation of all encodings that lie within the cone, i.e. numerically integrating the positional encodings of points within the traced cone the Positional Encoding  $\gamma^* = \frac{\int_x \mathbb{1}_{x \text{ in cone}} \cdot PE(x)}{\int_x PE(x)}$  (for more details see [1]).

### 2.1.3 Adaptive Scheduling of Frequencies

Yang et al. [33] aim to accelerate training and reduce training dataset requirements by introducing frequency and occlusion regularization. When using PE or IPE, NeRFs tend to overfit towards the high frequencies as they change more drastically in the embedding. This is mitigated by masking the frequencies, a process called **Frequency Scheduling** during the start of the training and only later on unmasking them. As the learning rate decreases, new frequencies are unmasked, so that the effect of new frequencies are regularized and catastrophic overfitting to the new, high frequencies is less likely. However, this approach makes convergence highly sensitive to the learning rate schedule and its exact values.

Secondly **occlusion regularization** is introduced, this alteration simply adds a small loss that penalizes predictions that are directly in front of the camera. There is often not enough overlap directly at the points in front of the camera to produce a multi view consistent prediction, that still adheres to the actual geometry of the given scene. In this case a simple occlusion loss is an easy way for the model to converge to a consistent prediction that renders a good result for the training view, but does not generalize well to novel predictions.

## 2.2 Other Preliminaries

While the previous sections covered core optimizations that directly impact NeRF training efficiency, this section introduces supporting techniques that can be integrated throughout chapter 5 and chapter 6 to further enhance model performance and generalization.

### 2.2.1 Depth-Augmentation

Currently most neural networks focus on creating consistent RGB-visualizations, for this a consistent geometry is learned. Additionally we can use the volumetric rendering to retrieve estimated depth maps from the prediction and use given or estimated depth data to compare the result to. This gives us a second way to observe the predicted instances, augment the input instances or penalize wrong results, which hopefully leads to a more straightforward training process.

In order to statistically recover the distance we use the expected value of the density:

$$d_{i,j} = \sum_{k=0}^z T_{i,j,k} * \alpha_{i,j,k} * Z_k$$

with  $i, j$  being the indexes of the ray and  $k = 0 \dots z$  being the indexes of the depth samples along the ray and  $Z_k$  being the distance of the corresponding  $z$  value.

**Depth-Supervision** is analyzed for instance by Deng et al. [8], they found that adding depth supervision can speed up training by a factor of up to 3x, as well as disambiguate the scene when only a few training instances are given. Methods with more complicated supervision [25] or specialized applications [11, 29] prove the use of this additional data, but are out of scope for this thesis.

**Depth-Augmentation** provides a second way to include spatial data into the NeRF, as especially in our case we first want to generate a prior based on the images of the scene, that already includes depth relations within the scene (for more details see chapter 5). Wang et al. [31] provide a useful way to include the depth in a prior, that allows the model generate more generalizable predictions instead of just speeding up training procedures.

The use of 2.5 dimensional data presents us with the problem that we now have to find a efficient way of integrating this into the model architecture.

### 2.2.2 Contrastive Learning

Contrastive learning, as a modern unsupervised training procedure is firstly proposed by Chen et al. [6]. The main task of this mechanism is to learn low level feature representations without the requirement of target ground truth instances. In each training step either a positive or a negative instance is given, positive instances are generated by permuting the same instance in a non ground truth data distorting way, such as data augmentations (for example cropping, blurring, or color jitter) while negative samples are just images from different distributions that were also permuted. Afterwards the representation is evaluated in order to achieve a meaningful loss by comparing the distance of the resulting latent feature representations, as positive samples contain similar information the feature representation should be the same, while negative samples contain samples from different ground truth and therefore a lot different information the distance in feature space



should be different.

The second important contrastive learning advancement was given in [13] by Grill et al.. Instead of relying on positive and negative samples, the authors solely focus on positive samples with not only working with one online network, but a second offline network that represents a moving average (to stabilize training) and show that this method brings similar good results and even better one for small batch sizes.

As outlined more detailed later in chapter 5, we assume the scene as ground truth we want to capture in features space, each image a view of the ground-truth. We aim to infer a latent representation of the scene incorporating the observed 3D-structure of the scene in this feature representation. So positive samples from the same structure are encouraged to result in the same latent representation, while negative samples from different scenes should be maximally far apart in order for the NeRF to distinguish different scenes whenever possible.

### 2.2.3 Architectures

There are several established architectures for image processing and generations, that are used in the following thesis, in this section I will explain them shortly.

#### Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a good way to distill early features from images. It is a deep neural network, that is augmented with convolutional layers, i.e. that applies the mathematical convolution operator on a block of certain size (typically from 1x1 up to 9x9), these convolute the weights with the input from the previous layer. This leads the network to better capture the local relations of the input pixels and with the cascade of such layers in the network we move slowly from local relations (low-level features) to global relations (high-level features). The advantage being that each layer only has a fraction of the parameters and is therefore faster to learn.

#### Attention & Transformers

A Transformer on the other hand is typically a more shallow model, that takes in several embeddings of picture parts as well as an positional encoding of where the embedding is in the overall input. Then the input is processed in attention blocks, that clearly state how the embeddings attend to each other (that is the key component of a transformer). A transformer contains multiple attention layers that can be used with three parts: key, query and value. The big advantage is, that attention is mostly applied on a global lever and can be leveraged in refined/abstract latent representations. Nevertheless the use must be considered carefully as the layer has a quadratic number of connections (such as fully connected layers) and therefore needs a lot of time and data to train.

There are several types of attention, that differ in processing such as multi head attention, scaled-dot product attention, linear attention, and much more products to overcome different issues. Moreover this mechanism can be utilized for different purposes most notably self and cross attention. Self-attention introduces the same value in key, query and value in order to refine the representation and allows to interact the values within each feature vector. Cross-attention on the other hand is used

with a different query than key and value in order to introduce new information to the representation. The following thesis only uses standard multi-head attention, to keep the training as simple as possible.

## 3 Related Works

While the methods presented in the last chapter are mechanisms that I have used directly simply to accelerate training and improve results, this section consists of already existent approaches in the generalization efforts put into the training of NeRF models, as well as other deep learning approaches to MVS, that can be considered similar or additive to the NeRF technology.

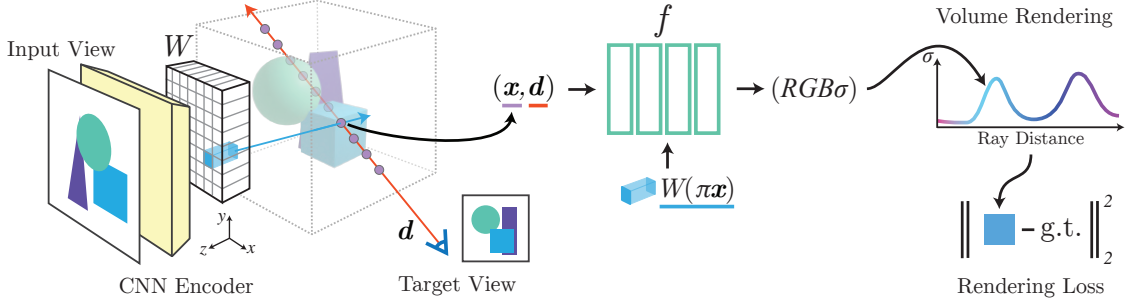
A short synopsis of the papers contributions to generalization problem: PixelNeRF was the first paper (to the best of my knowledge) to introduce ray prior, General Radiance Field refined this technique by considering point priors rather than pixel priors. Later on ContraNeRF helped refine this per point prior with contrastive per scene refinement of the prior. Now as another approach MVSNerf fused the per point priors with 3D CNNs to generate learned cost volumes in Order to enforce consistency more efficiently. GeoNeRF separated the density and appearance in cost volumes and used a more refined attention mechanism for fusion of cost volumes.

Some of the reviewed researchers [5, 15, 17, 28, 36] examine synthetic to real generalization capabilities, in summary, here they find that extensive pretraining with synthetic datasets can generate more sophisticated priors that generalize better to the given real data, then sparse real data as pretraining. This encourages data pretraining with synthetic data, as it is more flexible and the ground truth data can be adapted to the need.

### 3.1 Introduction of pixelwise Priors

Yu et al. [36] provide a way to introduce latent priors to the pixel predictions by augmenting every query of the NeRF with a corresponding latent that is created by creating pixelwise latents through a CNN that is prompted with another view of the same scene.

More concretely pixelNeRF optimizes the one and few shot learning by training a CNN (mostly on a general domain such as „car“ or „chairs“) in order to generate a *feature volume* that describes information about the pixel and their surroundings. Informations could include indications such as pixel depth, color or density distributions, but unfortunately there is no direct study over the information in this prior given, as this would be quite insightful to see what the network actually learns. Then based on the alignment information, the pixels representing the same 3D point in the prior-image are matched to the corresponding queried 3D-point, leveraging the information given by the epipolar geometry more directly. An 2D example of this can be viewed in Figure 3.1. Concretely Yu et al. found a direct improvement for only one to three images, also they give good visual results in the domain-agnostic and domain-transferred setting.



**Figure 3.1:** The prior selection in pixelNeRF (from the original paper)

### 3.2 Leveraging Multiple Images - General Radiance Fields

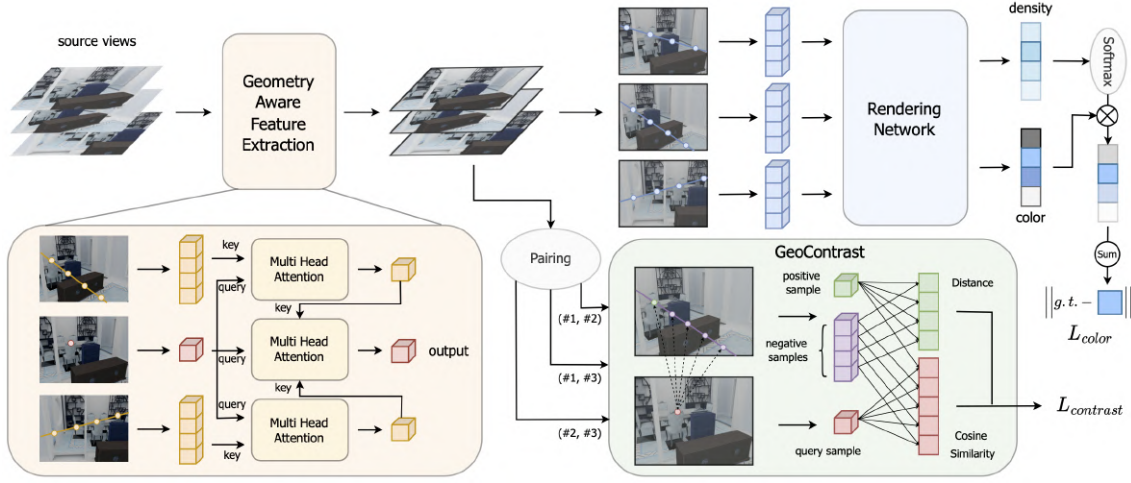
In a parallel work to pixelNeRF Trevithick et al. [28] provide conceptually an extension to pixelNeRF, as it adapts the mechanisms from Ye et al. and adds information to the prior by not using a single pixel to generate a prior but aggregating the information from several images. So again there is a per pixel feature extractor for each image, during inference the 2D features are projected to 3D space similar as in pixelNeRF and then aggregated to give a better geometric understanding of that point in order to render this scene. In step one the features are constructed with a CNN to result in one feature vector of length 128, secondly the features are projected into 3D space, here the authors overcome the ill posed question of finding the surface point in 3D space, by just assigning the feature vector to every point on the pixel ray.

Later the aggregation is done respective the queried 3D point, firstly all points are permuted by concatenating with the query point, using an MLP and later the attention mechanism to fuse the feature vectors, deciding which of the given features matters to the final 3D point feature vector, that can now be passed into a NeRF for further rendering. The different views provide a more stable generalizational capabilities, then just taking one example.

### 3.3 Moving from Pixels to Positions using explicit Differences

Yang et al. [32] aim to further improve generalization especially in the scene-transfer setting. In order to achieve a more general prior case, the authors construct geometric priors to each point, in a multi view setting, similar to GRF, but augment these priors with a contrastive learning method that should enhance the consistent geometric representation via *Geometry Aware Contrastive Learning*. Like the methods before, a CNN is trained to predict the feature vectors based on epipolar rays from a multi-view setting, then a two step multi-head attention mechanism is used to firstly attend from the points of the ray features to the concrete point query. So from the 2D features, a viewing ray is defined using the intrinsic and extrinsic parameters of each image and the rays points are sampled to lift the 2D features in the 3D space, from there on the two step attention firstly for each feature the features are enriched using cross attention by features from other views. Secondly these geometric feature maps from different views are used to fuse them all together using cross attention again. These enhanced feature maps can now be used in the Rendering Model of the NeRF.

Especially the training of the enhanced feature maps using contrastive learning supersedes previous approaches for efficient training and obtaining a meaningful prior in geometric space. This is



**Figure 3.2:** The ContraNeRF generalization pipeline as illustrated in the original paper.

achieved via using the InfoNCE loss on cosine similarity of the points sampled of a ray in one view  $i$  based on the features of these sampled points in another view  $j$ . Here positive samples represent the encoding that corresponds to the same geometric 3D location, which would be the features of the pixel from which the ray was generated in  $i$  and the features corresponding to the pixel from  $j$  that, as a ray intersects the other ray in the queried location. On the other hand every other sample in this ray is a negative sample. For better clarification an image of the pipeline is given in Figure 3.2. With this technique Yang et al. circumvent the limitation of only generating one prior per 2D pixel, by using the information given about the queried position in other images. As expected this clearly reduces the number of floaters and establishes a more consistent prior formulation across different images.

Secondly the researchers examine synthetic to real generalization capabilities, here they find that extensive pretraining with synthetic datasets can generate more sophisticated priors that generalize better to the given real data, then sparse real data as pretraining. This encourages data pretraining with synthetic data, as it is more flexible and the ground truth data can be adapted to the need.

### 3.4 Explicit Discretization of Features with 3D-CNNs

Chen et al. [5] aim at further generalizing the zero to few-shot prediction by fusing more traditional learning based approaches from the multi-view stereo setting with the NeRF volumetric rendering capabilities. This is done via voxelization of the world space and the generation of neural encoding volumes as priors for a sampled 3D point.

**Cost Volumes** are a prior estimation of the depth at which the ray corresponding to the pixel intersects with the first opaque object at entry point. Typically these proposals are trained in a coarse to fine manner (in resolution as well as depth prediction). This can be trained in a 2D-CNN setting, but typically is lifted to a 3D convolutional network using the image position and the resulting homography to encode the spatial information better. Then for each proposed point correspondences in other images of the same scenes are found and a cost is assigned to this point based on the similarities or dissimilarities of the found correspondences. This can then be propagated backwards as a loss value and be used to iteratively refine depth and color assignment to a 3D voxel. Yao et al.

provide a more comprehensive description on the computation of their cost volumes in [34]. The authors found that their trained MVSNet generalizes extremely good to other scenes, as well as single objects.

This generalization and similar algorithmic usage of cost volumes in a 3D setting makes cost volumes a good candidate to enhance NeRFs generalization capabilities to a new level. Chen et al. achieve this integration by firstly starting the model with a 2D feature extractor for every training view, as well as lifting these features in 3D space via warping and constructing a cost volume based on the variance of the features at the corresponding locations as enforcement of consistency, then instead of predicting depth (and therefore a 3D point) the authors use a 3D-U-Net to achieve latent representations they call „neural encoding volumes“, that should be enhanced not only with visual but also appearance features.

Finally this can now be used as a prior to augment the data in a NeRF, to query points and viewing directions and finish the training pipeline with the classical volume rendering, image loss and back-propagation of the pixel difference. More concretely a interpolation of the nearest neural encoding volumes together with the colors of the corresponding pixels of the queried location in other images are passed in the NeRF as prior.

The method clearly provides a feasible zero shot prediction for any scene, as well as faster convergence using end-to-end training. This is because the geometric consistency is enforced explicitly in the construction of neural encoding volumes.

## 3.5 Separation of Geometry and Appearance

GeoNeRF by Johari et al. [15] introduces a second cost volume approach, similar to MVSNeRF, they split the network into two functionally separate parts, one Geometry Reasoner that generates prior-tokens for each 3D query point and predicts color and density based on this prior.

Like in its predecessors, features are extracted from several views via a deep network and then warped into 3D-Space, building cascaded cost volumes. Now for each point several token are generated, the authors differ separate view-independent and view-dependent tokens from each other. These tokens are obtained in a hierarchical fashion to give latents in several features. In contrast to the previous methods the tokens are aggregated using attention heads over the different source views, this aggregation functionality can be seen as querying the understanding of a source view with an complete understanding of the whole scene (together with an occlusion mask) to generate new, more universal features, the so called „global view-independent output tokens“. These tokens are then used with an auto-encoder and MLP to predict densities, while view dependent tokens are used to predict a color via an MLP, after being re-projected to a positional embedding of the output image and augmented with an occlusion mask.

The most interesting feature about this model is the separation of prior tokens into view-dependent tokens, that predict densities and view-dependent tokens that predict color. This seems intuitive, as it allows to converge the density prediction more robustly, while the color perspective plays a secondary role, this should encourage the system to come up with less regularizers. In contrast to this architecture Johari et al. find, that texture-less areas provide a failure case, which indicates that the CNN takes meaningful depth information from the view input that is fused irregardless of perspective color changes. This is one of the first networks that demonstrates, that we can fuse density and color completely separately.

### 3.6 Selective Attention to specific Sampling Points

Reizenstein et al. [26] aim to fix the same problem as GeoNeRF, namely the geospatial reasoning of a 3D point as part of a ray rather than a single point. Contrary to Johari et al. they argue that the neural cost volumes provide a discretized space, that because of inherent resolution bound in the given grid will lead to blurry images. Since here we focus on every voxel to build a volume rather than the important parts of the neural field, that contribute to different appearances in the result. Instead of using an explicit mechanism to combine the priors of the ray or generate a prior for each queried position independently and then fusing them together, the priors are generated for each position with respect to several images and camera positions, then the fusion of the priors is done via a learned attention mechanism and then uses 3D transformer encoder to accurately predict results. Finally the results are obtained by using weighted pooling layers and a color as well as a density head for the final prediction. The results show that while the model provides no meaningful addition to single view optimization, it can benefit from scene agnostic pretraining for simple scenes, here especially simple scenes with 5 or more ground truth views are enhanced.

### 3.7 Reducing Discretization Relicts with Local to Global Feature Fusion

The NeRFusion model introduced by Zhang et al. [37] builds on the Cost Volume approaches represented earlier and aims to bridge the gap of the previously introduced methods from object centric scenes with very artificial dataset layout to „large-scale“ scenes, that consist of rooms, such as in ScanNet [7].

Currently the most generalization methods require views from very different perspectives with a big overlap to predict object positions correctly (such as [14, 20, 21], this limits the practicality to inward facing scenes rather than trajectories or rooms. (see chapter 4)

The authors augment the feature volume approach by augmenting the 2D feature lifting in 3D space with a local feature volume creation. This means instead of directly lifting the 2D image features created by a CNN encoder into a global 3D space, they generate a local volume reconstruction by considering the  $k$  neighboring images and build feature volumes from them. This step enables the fusion to focus on more finer grained details (similar to global feature generation from a dataset such as LLFF [20] before generating a global representation). In the next step the local volumes are fused together using a Global Volume Fusion, for this they use *Gated Recurrent Units*, which updates the global 3D CNN locally with beforehand generated local volumes, to generate a coherent representation of everything represented by the scene images.

### 3.8 Improving Raywise Priors with Occlusion Computation

Liu et al. [17] developed another model that uses occlusion/visibility feature maps of 3D points to adapt new scenes. This selection criteria allows for raywise priors, that are related very closely to the pixel result, without introducing unnecessary features from 3D points that do not contribute to the color or density of any points in the ray and therefore lead to poor quality results.

In this network the input views are used to construct either cost volumes or depth maps from

which visibility feature maps are generated for each 3D point, all of these sampled 3D points are aggregated together with visibility and used as a prior for a given NeRF. This is then used in the standard NeRF - volume rendering mechanism in order to generate the output color of a pixel.

The occlusion is determined by the consistency of depth/cost volumes in the input views, i.e. if the depth at which all the rays are estimated to end intersect in the same point consistency is assumed. This introduces a sparsity in the visible features that can be beneficial in prompting the NeRF as now a lot less features per ray can be considered. Effectively a selection method for features and the importance based on consistency is given, without the usage of discrete cost volumes, but adaptive sampling points. Unfortunately this network fails in generalization to points that are not visible from any views in current work. This indicates that the features the model learns are no geometric prior (like e.g. pixelNeRF or MVSNerF), but more about consistency fusion of different feature (e.g. ContraNeRF).

## 3.9 Other 3D Reconstruction Methods

In this subsections two methods to reconstruct objects and novel views based on depth are presented.

### 3.9.1 Occupancy Networks as supervised NeRF Alternative

Another method, that generates an implicit and contiguous representation of a given 3D space. Proposed by Mescheder et al. [19], with one important follow up work by Peng et al. [23]. The major differences to NeRFs are 1) only occupation, not density or color are predicted; 2) a 3D supervision (Points, Meshes) is needed and 3) some form of 3D input (such as points) has to be given.

This method is interesting for my work especially because the generated scene prior from 3D data is given as a feature vector containing a latent representation of either 2D or 3D plane feature interpolation of nearby points interpolated. This method. It is especially good with single & smooth objects in one domain, but generalizes to scenes.

Functionally the occupancy network is prompted with an arbitrary 3D input, that is encoded with a correspondingly trained encoder, now the feature vector is projected onto multiple planes (either some 2D planes or one 3D plane) from which a decoder decodes the feature contribution towards a given point  $x$ , this feature vector is then further processed to predict an occupancy probability.

The most interesting part is that the occupational network (first version) can only be prompted with a low number of 512 parameters to encode a object such as chair, if trained on all chairs from the ShapeNet dataset [4].

### 3.9.2 Scene priors with explicit Multi View Stereo Computation

This work by Cao et al. [3] is a follow up to the MVSFormer [2] architecture previously, and while it is not directly a NeRF, it is the current state of the art network in general multi view stereo and therefore definitely contains interesting components for our novel view synthesis problem, even though it is considered a multi view stereo problem, that only reconstructs depth based on 3D points. The transformer structure contains a scene-agnostic windows with global attention, feature encoders,



cost volumes and positional encoding, so there is a significant methodological overlap to the previously reviewed methods.

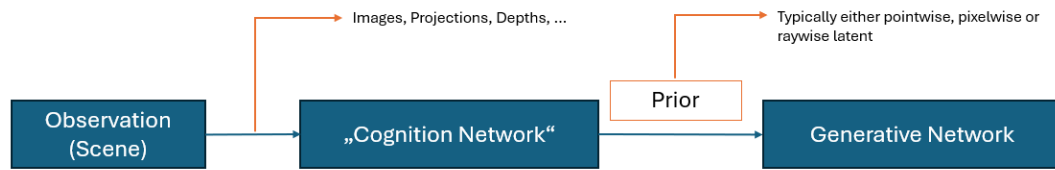
The first part of the network is a handcrafted feature encoder that includes a feature pyramid network to sample the instance down, processed at the lowest level and then up again, with the theory being that the coarse latent representation is for incorporating reference views, while the second pyramid is here to add features from the first and fuse them together with the newly learned information from the reference views. The attention mechanism to incorporate reference view is called „Side View Attention“(SVA). The SVA is applied in several consecutive modules, in each module a latent representation (using a DINOv2 backbone) of the source view (being the view we query for distance), that is firstly used as query with reference views being used as key and value pairs and secondly self attention applied to the latent source view. This alternating attention application allows the features to interact with each other and also with the reference views without being distorted too much. A good illustration for such a module is given in Figure 3 of [3]. Secondly the classical cost volume formulation is used in order to fuse the source view with the reference view. More comprehensive a four stage coarse to fine strategy using warping, fusing features into global space, frustoconical positional encoding, cost volume transformer or another 3D convolutional network is used to obtain the final result.

Especially interesting is the first feature encoder, since the separated feature encoder can be seen as a 3D capture of the scene. The following observations are notable 1) the cross attention mechanism is not prompted with any camera position, relations on the images are based on the similarities in the images, still most of the performance enhancement is attributed to this attention mechanism in the ablation study. It effectively captures the long range dependencies 2) adaptive attention scaling helps cost volume regularization to be more flexible, and 3) a feature pyramid network with fine to coarse is helpful for increasing resolution.

## 3.10 Overview

In this chapter several methods of designing generalizable NeRFs were presented, a short overview is given in Tabelle 3.1. The current approaches could be summarized in a framework like shown in Table 3.3, namely two major components one Cognition Network, that is prompted with a observation (such as an image (reference view), a depth map or a 3D estimation) and from this generates a prior that describes a fixed part of each scene and is used in the prediction of the respective point by the generative network.

In general there are two main problems that all of the authors of new generalizing methods try to solve: 1) how do we get a prior that describes the 3D-point we want to predict most accurately, using only 2D (or 2.5D) data and 2) how can we enforce consistency along the priors in respect to different observations.



**Figure 3.3:** Framework for generalizable NeRF-constructions

**Table 3.1:** Comparison of NeRF Methods

	pixelNeRF	GRF	ContraNeRF	MVSNeRF
Input Views	Single	Multiple	Multiple	Multiple
Learned Examples	Categories	Categories	Scenes	Scenes/Objects
Prior	Raywise	Raywise	Pointwise	Pointwise
3D Lifting			Epipolar Geometry	Homography
Additional Learning Mechanism	2D CNN	2D CNN, Attention	Attention	2D+3D CNN
Consistency Enforcer			Contrastive Learning	Neural Cost Volumes
Scene Capabilities	–	–	+	+
Zero / Few Shot Capabilities	o	o	++	++
Problems	Complicated Scenes	Complicated Scenes	Blurred images	Light effects and low overlap
	GeoNeRF	NeRFusion	Neural Rays	
Input Views	Multiple	Multiple	Multiple	
Learned Examples	Scenes	Objects	Objects	
Prior	Point + Raywise	Pointwise	Point + Raywise	
3D Lifting	Homography	Homography+GRU		
Additional Learning Mechanism	2D+3D CNN, Attention	2D+3D CNN	(Depth Estimator)	
Consistency Enforcer	Cost Volumes	Cost Volumes	(Occlusion Indicator)	
Scene Capabilities	++	++	?	
Zero / Few Shot Capabilities	++	++	+	
Problems	Textureless areas	Background/Foreground-Diff	Occluded pixels	



## 4 Datasets

The performance of NeRFs is significantly dependent on the the layout of the dataset. In order to extract a good geometry representation the overlap of images from different directions is essential. Generally datasets can be distinguished by several categories, either RGB or RGB-D, virtual or real, but for our case the most useful categorization is by Camera Layout.

### 4.1 Datasets

Initially I implemented five different datasets with four real life and one synthetic datasets, two front facing datasets, and four depth datasets - after considerations only three datasets are used for training, these three are explained further in this section. Besides the LLFF, Blender, and the DTU Dataset, I implemented SceneNet and ScanNet, but did not use both of them, because the camera layout does not allow for diverse overlaps, that generally help NeRFs achive good consistency. Figure 4.1 gives an overview over the camera layout of all implemented and used datasets.

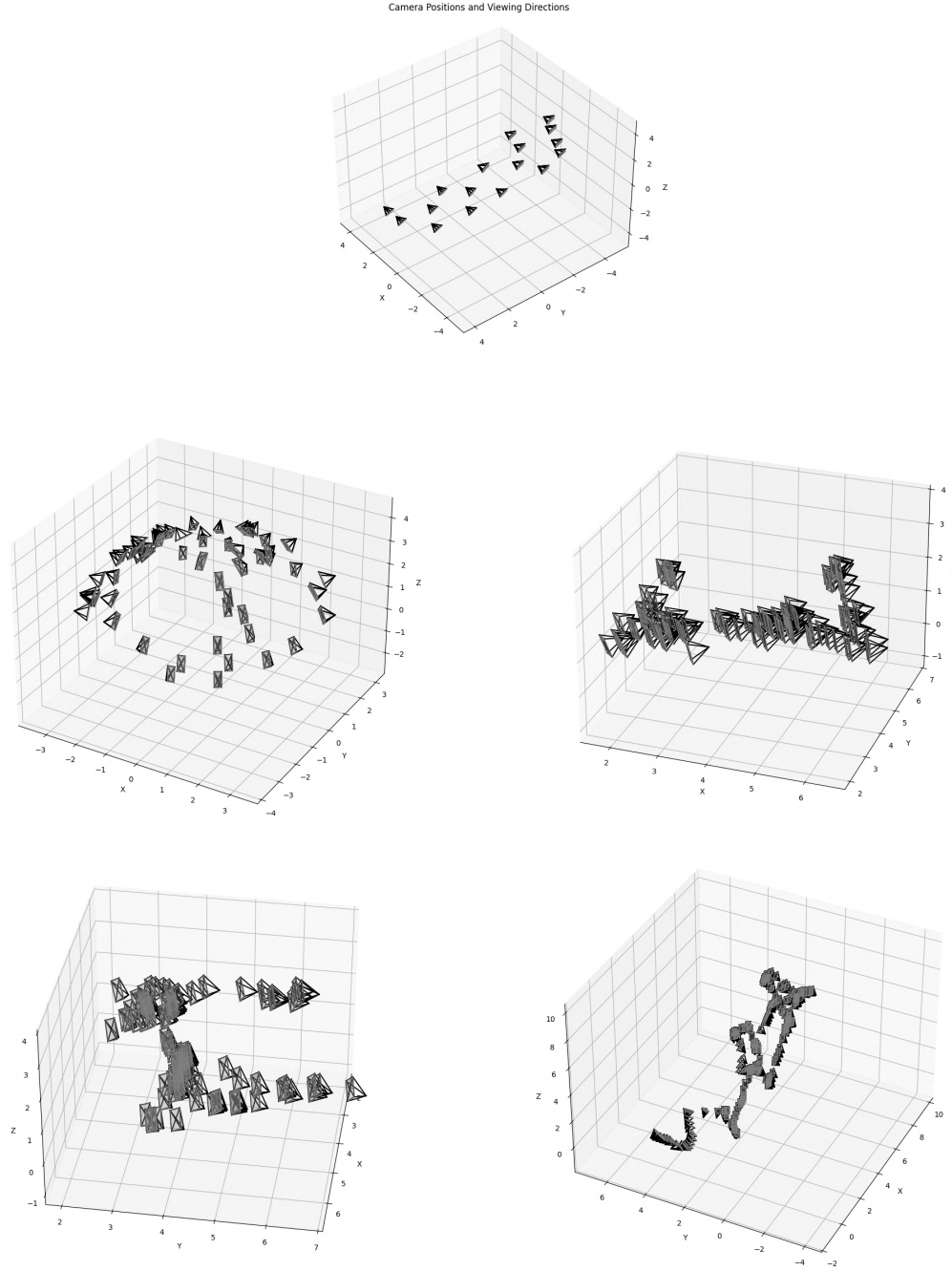
#### 4.1.1 Blender Dataset

The Blender dataset introduced in the first NeRF Paper by Middenhall et al. [21]. It is by far the most used dataset for NeRF evaluation, this is due to the good camera setup, depending on the scene they are either irregularly sampled from a half dome or a whole dome, also they have very distinctive lighting, differ in back-/foreground and are in general very detailed (every image has the resolution of 800x800). The only problems with this dataset is, that it only contains 8 scenes with each 100 images and no depth ground truth data to compare against. Figure 4.2 shows several scene examples from ground truth image data (using white background).

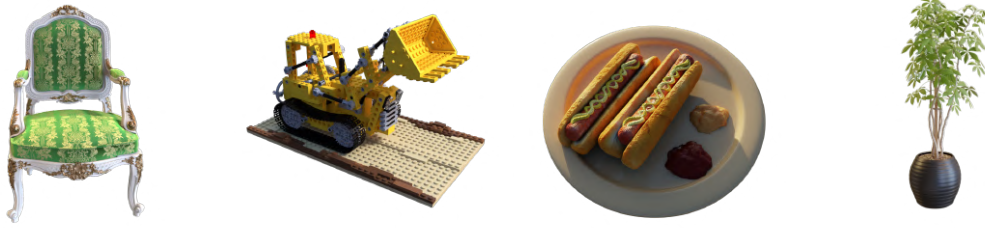
#### 4.1.2 DTU-MVS

The DTU Multiple View Stereo dataset contains 124 scenes with each image being taken by a robot with either 49 or 64 images taken, each with a resolution of 1600 x 1200. This ensures that the network cannot infer the scene during training from the position, since every scene has the same positions. Additionally Jensen et al. [14] provide different lighting scenarios - from which I only use the one with the most lighting - as well as depth information, which can be leveraged either as training target or additional input data.

This dataset is especially interesting as it is more challenging, than the Blender dataset, it is harder to separate the scenes, so models that work on Blender do not necessarily work on DTU. Nevertheless the real life dataset still has a somewhat favorable camera layout and good quality in comparison to the real life room datasets. Images from the dataset are shown in Figure 4.3.



**Figure 4.1:** Different plots showing the camera layout of the first scene from the LLFF Dataset (top), Blender Dataset (middle left), DTU Dataset (middle right), ScanNet (bottom left) and SceneNet (bottom right) for reference of real life natural scene datasets. One can see the dome-like layout of the Blender data, as well as the Forward facing camera layout of DTU and LLFF, with ScanNet and SceneNet as references, where there are more complex camera paths with different directions and less coverage from different perspectives



**Figure 4.2:** Different examples of Blender Dataset scenes.



**Figure 4.3:** Different examples of DTU-MVS Dataset scenes.

#### 4.1.3 LLFF (Local Light Field Fusion)

LLFF by Middenhall et al. [20] is a second real-life Dataset, that is strictly Forward-Facing, meaning that there is a real object in the front and the samples (nearly) stem from a plane in front of the object and are facing in exactly the same direction. This creates a high overlap in the farther regions and helps to correctly infer the geometry there. In contrast to the DTU and Blender Datasets that focus on a direct object in front, LLFF also has background depth that can be trained and evaluated. Figure 4.4 shows exemplary images from the first of eight scenes.

## 4.2 Preprocessing of Datasets

Each dataset is firstly read in, this contains images, depth, and camera positions as well as extrinsic and intrinsic parameters of the cameras. These parameters are necessary for computing ray sampling positions as well as pixel footprints for conical contributions.

Secondly the positions are recentered to zero with equal extends to each side, this allows for a better use of positional encodings. Using the recentered positions allows us also to use normalized device coordinates for some datasets. These NDC coordinates only make sense for fully forward facing



**Figure 4.4:** Different examples of LLFF Dataset scenes.

datasets as otherwise the space gets skewed based on the camera layout, this is a way to normalize all queried coordinates into a space between -1 and 1, which should stabilize training. In the training as described in chapter 6 NDC-Coordinates are only used selectively.



## 5 Training & Network Architecture

Capturing a detailed scene and being able to render it from just a few images is a very hard task, that will not be accomplished easily. This imposes the need for some intermediate goals and division into several components that can be optimized independently and interchanged, in order to find out what mechanisms can work together. Secondly also the training process needs to be broken down into different losses and backward optimizations to optimize each component independently and all together jointly.

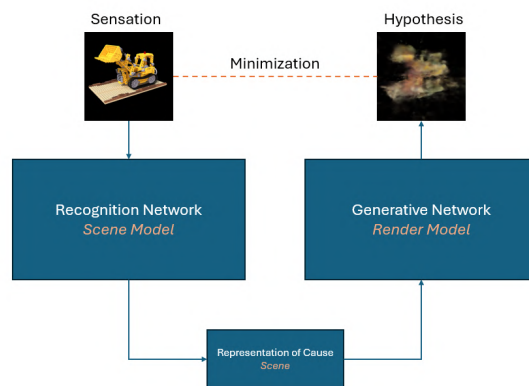
In this chapter I firstly explain the theoretical foundations on perception of three dimensional scenes from a theoretical perspective (Section 5.1), then in Section 5.3 the architectures used in scene perception will be presented, lastly Section 5.4 explains the different ways I tried to train the models, as well as the possible applied losses.

### 5.1 Conceptualization

The theoretical foundations from a neuro-theoretical perspective aim to explain on the one hand how stimuli can be transferred into a neural model of the reason behind this sensations. They build a cornerstone of moving from learning scenes to perceiving scenes, as illustrated in this section.

#### 5.1.1 Free Energy Interpretation

The Free energy principle, as reviewed by Karl Friston [9] explains the differences between perception and learning in the context of intelligent agents observing their environment. Friston defined **Free energy** as „an information theory measure that bounds or limits (by being



**Figure 5.1:** Free energy Principle of Recognition

greater than) the surprise on sampling some data, given a generative model “[9]. While the first aspect of the principle is widely implemented in reinforcement learning as the State-Action-Reward loop, the implications on perception without direct interaction are less obvious. Friston summarizes the Free Energy Principle as: *Minimization of the free energy of sensations and the representation of their causes*.

This idea can be applied to our problem using a dual network framework: a *Recognition Network* extracts a representation of causes on input of one sensation and a second *Generative Network* that forms assumptions about the representation of their causes that can be evaluated as free energy to new (or existing) sensory input. In short, the goal is to generate a representation that, on the one hand, can be derived from sensory input (e.g., images), and on the other, can drive a generative model to produce a corresponding sensory hypothesis. The objective is to minimize the disagreement between these two pathways.

In the context of 3D reconstruction, this framework involves two networks: one recognition network which takes input of sensory information (e.g. an image, depth data or additional information of the scene) and outputs a latent vector representing the scene abstractly. A second, generative network (in our case a NeRF model that generates a new image), is used to generate a new assumption, that can be compared to the the initial sensation. This network then minimizes free energy via gradient descend using a back pass of error units.

As illustrated in Figure 5.1, this framework provides a natural formulation and enables end-to-end training, closely aligned to the current approaches of generalizable neural fields.

### 5.1.2 Perception vs. Learning

Training Neural Radiance Fields on a large collection of scenes to generalize to previously unseen scenes is a current approach to shift from learning 3D-environments to actively perceiving 3D-environments. Classical NeRFs treat a scene composition as a fixed set of parameters, that can be encoded in the the network’s weights, from which novel views can be generated. However, their inability to interact with or adapt the world as it is reveals a fundamental limitation. In order to further develop this self-supervised spatial understanding mechanism towards new capabilities in real world applications the second part, a discriminative perceiver must be introduced. This addition enables the network to become more versatile and flexible in practical use. This gap in capabilities can be bridged by gradually introducing (1) more general priors - given a vast diversity of possible scenes. Rather than just focusing on e.g. a special object or scene-type, as done in many early approaches (Chapter 3), researcher now aim to generalize beyond a dataset to new scene types with inherently varying complexities, (2) more comprehensive priors - currently most networks predict a NeRF prior that is point-wise or ray-wise manner, which allows for strong local adaptability and fast training capabilities. However, this approach is problematic, when generating truly novel views. Such networks often fail to come up with features that explicitly represent the scene geometry, but rely on a fine grained raster, to interpolate priors from, effectively adhering to a cost volume approach, with only smoothing factors inserted through the NeRF rendering.

To truly perceive a scene and operate within a generative setting, not only a local understanding of geometry, but moreover a global understanding is needed, to adapt the network to new and more complicated scenes that differ from the training data.

As this represents a significant departure from existing models, I focus on the more managable objective of training multiple scenes in a network and use a (re-)cognition network to identify the

scene based on observation, produce a meaningful feature representation, and pass it to a rendering model. The challenge lies in transitioning a NeRF from learning a scene towards learning the perception of scenes - i.e. perceptual learning - while retaining key advantages such as compact network size or realistic lighting extrapolation.

## 5.2 Predictive Coding - Applying free energy

In machine learning the free energy is applied using predictive coding, this approach to visual processing was originally proposed in 1999 by Rao et al. [24] using simple machine learning model to make a generative counterpart in order to propagate errors more efficiently to capture scenes better.

Currently predictive coding experiences a surge due to the interest in active learning as an area in reinforcement learning in interactive environments.

Gornet et Thomsom [12] provide an auto-encoder framework to capture positional scene maps, this is interesting since the Encoder has to capture a inherent scene and predict the next image, similar to the functionality of NeRFs. The difference is firstly that temporal relations are much more relevant, as the trajectory provides the next logical image position and from that the next image is extracted and secondly that a reversed ResNet is used to generate images instead of a NeRF, which reduces the training complexity a lot but does not provide explicit 3D information.

This first paper indicates that the predictive learning framework can be efficiently generalized to arbitrary scenes, from this we could conceptually also formulate especially the scene capture part of our task to a offline predictive coding task.

## 5.3 Architectures

The basic architecture consists of a cognition model and a generative model, as previously discussed. Since NeRFs are typically effective generative models and serve as foundational building blocks, constructing the generative component is relatively straightforward. However, designing the cognition model is significantly more challenging, as it requires addressing the ill-posed problem of 3D reconstruction—specifically within a latent feature space.

### 5.3.1 Recognition Model

The cognition model has to generate a description of the scene, that is consistent, clearly separates the training scenes, and describes the geometry of the scene well enough that it can be rendered by the render model.

One of the initial challenges is that only a limited number of images can be input into the model. Since we aim to avoid reliance on multiple simultaneous views to establish a consistent scene representation, the solution is to maintain and iteratively refine a latent representation for each scene. This means that during each pass through the neural network, an already existing latent is provided as input. The resulting updated latent is not only forwarded to the generative model but also stored for future refinement. This approach stabilizes the training procedure overall, even though no explicit experiments are conducted to evaluate the effectiveness of this approach isolated

of other parameters, as a closer evaluation would be beyond the scope of this thesis. As a result, the NeRF struggles to maintain separation between different scene geometries. As a regularizing mechanism, this latent is reset to a fixed starting value with a certain percentage, so the system does not just learn to interpret these feature vectors, but is continuously capable of generating feature representations from a zero-initialized state.

Another critical component is the image itself. Each model receives a single scene image per step to refine the latent representation, using a pretrained encoder to extract high-level visual features. Additionally to the image we try to augment position and direction data (our 5 DOF,  $x, y, z, \phi, \gamma$ ) and optionally a depth map, depending on configuration and data availability.

The RGB and depth encoders include variants of ResNet (e.g., ResNet18, ResNet38, ResNet50, ResNet101) or CLIP. Since CLIP employs explicit type embeddings, we use a frozen version to reduce computational overhead from backpropagation.

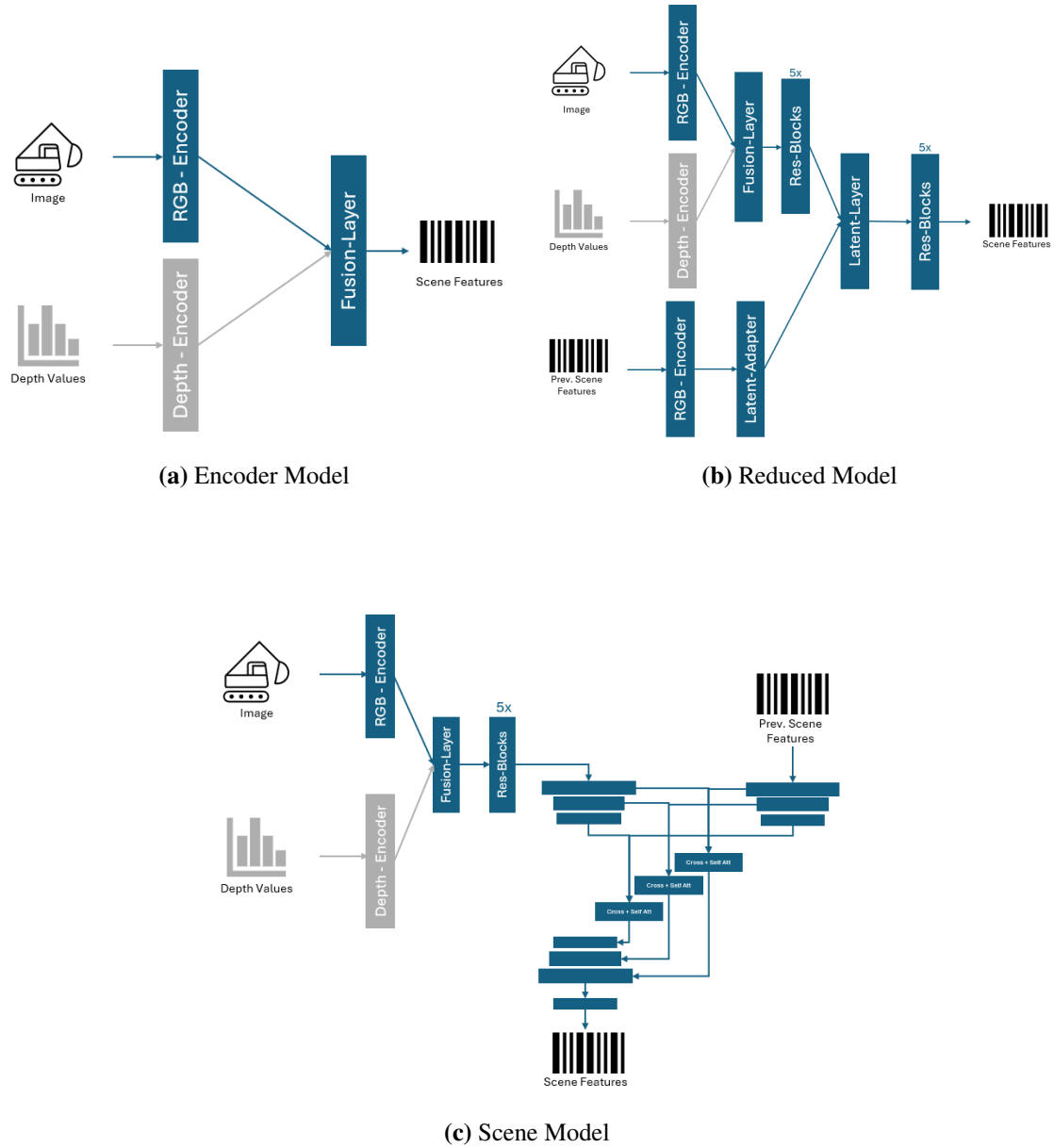
Overall, neither position nor depth data contribute significantly to training. This is likely because these additional modalities introduce noise. Similarly, the choice of encoder has shown limited impact on recognition accuracy in my preliminary experiments and qualitative evaluation.

Due the lack of comparable targets and existing architectures, I draw from more general-purpose 3D reconstruction architectures such as [3], to identify structural components useful for our training setup.

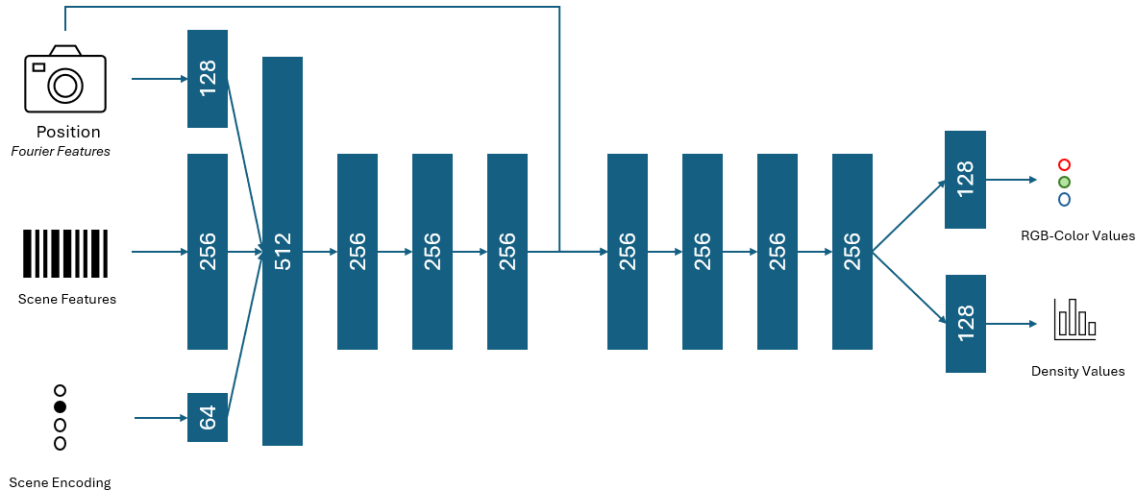
- Feature Pyramid Network (FPN) - Mapping global image features to a global latent representation is particularly complex. Using a smaller latent vector helps simplify this mapping, making it easier for the model to learn the association. However, small latent vectors tend to discard fine-grained details, which can degrade reconstruction quality. To address this issue the features get gradually down sampled in fully connected layers. These are then fused at the coarsest level and then gradually up sampled together with residual connections to keep details in place.
- Residual Blocks - These connections conceptually provide simple implicit operations to refine existing intermediary representations between building blocks such as encoders. Residual connections are considered especially useful due to their capability to stabilize training and enhancement of features.
- Cross Attention - This feature is especially useful to fuse new information into a combined representation. In our case, they are used to incorporate current scene features into the existing latent vector.
- Self Attention - Self attention is used to sort global information within the feature vector efficiently, it is considered much more expressive than Residual Block or even fully connected layers, but tends to introduce more unstable behavior within the network.

I combine these building blocks into three networks that are explored in this thesis: (1) the scene model in the preferred layout using a out of the box encoder, FPN for both latent and image feature extraction, as well as cross and self attention to fuse latent and image on several scales. (2) A reduced model that only consists of a given Encoder an adapter layer and Residual Blocks, this is a baseline model to access impact of different losses, training procedures or dataset changes without overfitting to a specific architecture. (3) A Encoder model, that contains the minimum information to extract from a image, it is just a encoder and one adapter layer.

A schematic overview of all models is provided in Figure 5.2. Since the general setup is still very explorative and there are some overall issues in the training algorithm, there is no extensive comparative experiment between these models done.



**Figure 5.2:** The three different scene models that I have tried in my initial research. Grey instances are optional parts. If not specified otherwise the layers are fully connected.



**Figure 5.3:** The Render Model, the scene encoding is described closer in chapter 6.

### 5.3.2 Generative Model

The Generative / Render Model is closely related to the original NeRF Model, it mainly contains eight fully connected layers. This is once since I did not find any significant improvement in changing layer sizes or types and secondly since it we definitely want to focus on the quality of the scene features as 3D capture object.

The generative or render model is closely based on the original NeRF, consisting of eight fully connected layers. This design choice is driven especially by the overfitting behavior of any larger network as well as the incorrect generalization with less layers.

## 5.4 Training

I implemented a custom NeRF Framework based on the original paper [21]. In order to optimize training time and efficiency I incorporated IPE and cone tracing as described in [1], and occlusion regularization as in [33]. Both methods are described in detail in chapter 2. I also experimented with frequency scheduling to mitigate specific training artifacts but observed significant instabilities when applying it to multi-scene training.

An optional depth encoder was implemented to incorporate depth information into the training process. I chose to do a simple late fusion, which means two separate encoders are used to process RGB and Depth data and then they are later on fused using one fully connected adapter layer. This design facilitates the reuse of pretrained RGB encoders, which is advantageous given the lack of high-quality pretrained RGB-D encoders.

### 5.4.1 Contrastive Training

In order to realize a contrastive training component, I used two different training schedules. First, I performed disjoint contrastive pretraining with batch sizes ranging from 80 to 240 images. Second, I tested a schedule that interleaves epochs of contrastive training with NeRF training. Each time the loss is computed based on the cosine similarity of the latent feature representation generated by the recognition model. Positive samples are images that are taken from the same scene, while negative samples are images from different scenes. This encourages the model to learn a feature representation that is not view-dependent, but instead captures the core semantics and geometry of the scene.

### 5.4.2 Losses

While normal supervised-learning typically has a simple loss definition, in the self-supervised NeRF setting the loss is less straightforward, since it is not applied directly to the predictions of the Neural Network, but there is only a ground truth for the processed result of several predictions at once.

Several losses that could help extract a meaningful scene from the network, I have implemented the standard NeRF MSE Loss considering the ground truth pixel color vs. the volume rendered ray color.

Additionally I have implemented and tested a depth supervision loss, which uses the weighted densities of a ray to compare the estimated depth to a ground truth depth, which could help not only to accelerate the convergence of neural networks, but also prevents wrong geometry extractions. (compare [11] and [8])

Occlusion regularization following [33] was implemented to prevent predictions that are wrongly close to the near plane and occlude the predictions on the space of the actual geometries. This is a way to regularize for wrong predictions in early training, that are especially hindering the training process. These “floaters” are mostly an artifact of insufficient overlaps close to the cameras.

Lastly, a consistency loss similar to the loss described in [22] was introduced, to regularize based on appearance not only single rays. In order to simplify this procedure only a very low resolution image from a known pose is rendered and passed on through the recognition network. Then similar to contrastive learning the network is penalized with cosine similarity of this latent with the original latent. This method can be interpreted as an inverse application of the principle illustrated in Figure 5.1. Despite its theoretical advantages, it is impractical to train with this method, as the gradients and intermediary steps require too much memory with a naive implementation.





## 6 Experiments

In this chapter I evaluate the network architectures and training processes described in the last chapter. While the initial goals of robust cross-scene generalization were only partially achieved, the experiments reveal valuable insights and highlight challenges that future work can build upon. The training of multi-scene NeRFs is obviously more resource-intensive than training a single scene NeRF, since there are more complex requirements and more images in a disjunct setting are given. Additionally the separation of incoming scenes needs to be learned, so that the rendering model is able to distinguish the scene that is put in and should be retrieved (ideally not only from the One-Hot Encoding).

All this together makes it necessary to investigate how tuning of the parameters influences the training results and how we can achieve the best performance with a limited number of resources.

**General Training Setup** If not specified otherwise all training steps were conducted on a Nvidia A100 (40GB) with a positional encoding of 5 cosine and sine-frequencies using Integrated Positional Encoding as described in [1]. We use two step hierarchical ray sampling with 128 first step samples and additionally 128 second step samples. The initial rays are always sampled uniformly with slight perturbation, the second sampling is done using importance sampling on a PDF generated by the first ray density function. Each image is sampled with 2048 rays, this is the so called „Image Batch Size“. These parameters are not optimal and are not tuned robustly, but I found them to fit best with the given memory constraints. A robust investigation of these parameters would exceed computation capacities as well as the given time, especially for large scene sizes.

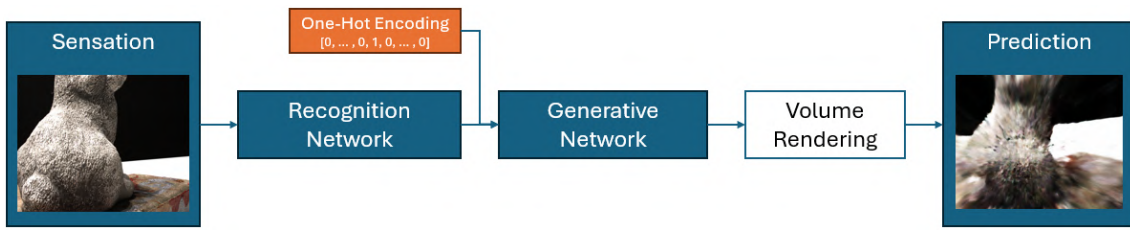
Furthermore I train the NeRF by default with a batch size of 16 (this includes a mini-batch size of 4, i.e. each backward pass consists of 16 images with each 2048 ray samples), a Mean-Squared-Error loss, with ADAM-Optimizer. I tried several learning rate schedulers, such as Exponential, CosineAnnealing, CosineAnnealingWarmRestarts and the custom MipNeRF Scheduler, but again did not investigate them exhaustively. I found the CosineAnnealing-Scheduler to work best and decided on an initial learning rate of  $5 \cdot 10^{-4}$  as given in [1], but a final learning rate of only  $10^{-5}$ , to ensure a more stable training. I worked with 250 Epochs as standard measure, but evaluated several lengths between 100 and 1000 epochs.

If not specified otherwise I used the encoder model with a ResNet-18 RGB encoder and preloaded, but not frozen, weights. This is done to investigate the effects using the simplest possible model. By default only the MSE-Loss is applied.

### 6.1 Improving Distinguishability of Scenes

When interpreting several scenes, the novel challenge for the network is to recognize a scene and from there on render its estimation of this scene.

This is seen when naively training a NeRF with several scenes, e.g. DTU, where the background is



**Figure 6.1:** Illustration of the new latent component

the same, the NeRF will converge to the same solution, while areas that differ will be represented with blurred see through color and density color values.

Since the goal of this thesis is provide a path that allows neural radiance fields to perceive scenes rather than learning scenes, any distinguishing mechanism should work with features describing the scene rather than learning features per scene (which would only relay the problem), for this the Recognition model is described earlier in Section 5.3.1 with still leaving the training procedure open.

The following methods did **not** achieve the required distinguishability, in order to render scenes right:

- Contrastive pretraining solely with the recognition model. Here we see a mode collapse with vanishing losses even for the Blender Dataset (which contains several major differences in objects). This is irregardless of the Model Type.
- Interleaved training of NeRF and contrastive epochs. This should theoretically help guide the recognition network to build a meaningful latent structure in the NeRF training as well as scene-wise different priors in the contrastive training. With the hope, that the large batch sizes help clearly training with differences in the scene structure. Due to the ray batching and limited memory a comparable size of different scenes would come with other drawbacks in the NeRF training.
- Consistency loss, as here the loss on the scene representation is applied from both sides (in reference to Figure 5.1). Once from the generative network, using the latent from rendered patches and rendering the latent of the recognition network. This is a much tighter coupling of the networks and should therefore be more aligned with our target but also fails.

While the aforementioned approaches fail to generate a meaningful implicit prior, that can be used to distinguish the scenes, the next step is to concede that this task in itself might be too hard for the network and a single implicit approach alongside NeRF building up is too complicated for a neural network.

In order to simplify this training procedure, I included a scene encoding to the method that should function as a training aid for now and could be removed later, when the Generative Model is able to render scenes based on the prior features as well as the scene encoding.



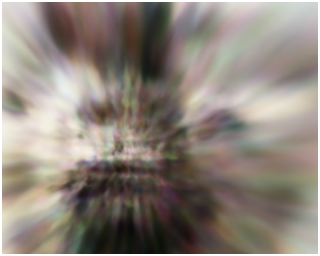

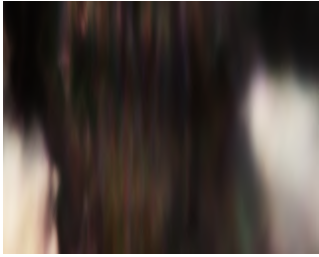









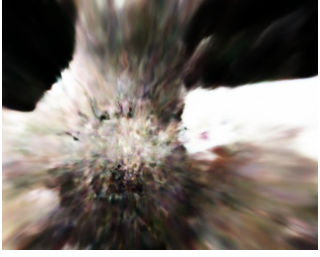



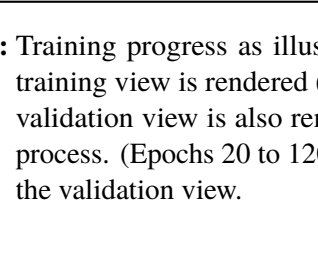
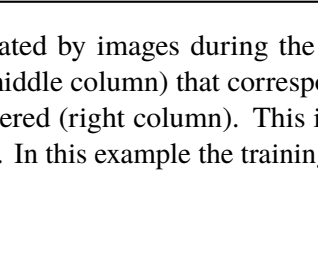
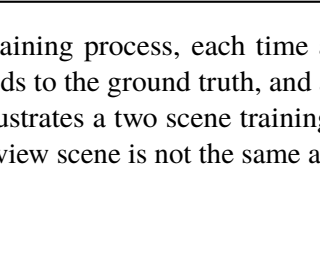



The One-Hot Encoding is used for this approach, so each scene is given an number and depending on the number of the scene the latent is augmented with the neuron in the scene encoding (which is appended to the latent) is set to one, while all others remain zero. Initially I use 64 neurons for a maximum of 64 scenes. This second part of the latent is shown in Figure 6.1

With this mechanism the Generative Network is able to distinguish scenes robustly, as seen by the results in for example Figure 6.2 - even though the renderings quality itself is not good. In order to

test this distinguishability I scaled the training from one to 64 scenes in parallel training, without observing any problems, even though DTU contains also scenes from the same objects in different constellations.

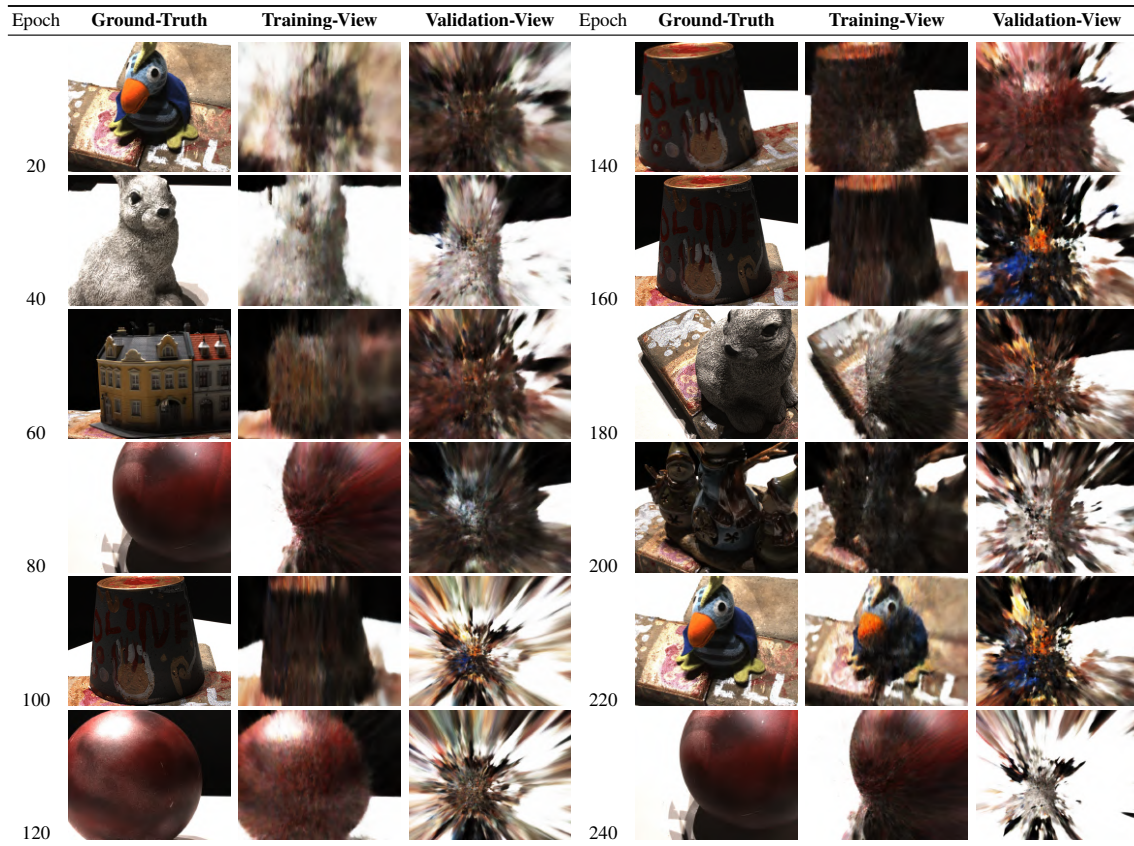
### **6.2 Predicting the correct geometries.**

As I have found a way to train one network with several scenes interleaved some problems emerged. The Network tends to overfit, i.e. have most of the results at the near plane, rather than build consistent geometry predictions, this is investigated in the next subsection. Also I have found that the meaning and interpretable results of the output generated by the Recognition Network is most likely not as strong as I hypothesized.

Epoch	Ground-Truth	Training-View	Validation-View
20			
			
40			
			
60			
			
80			
			
100			
120			

**Table 6.1:** Training progress as illustrated by images during the training process, each time a training view is rendered (middle column) that corresponds to the ground truth, and a validation view is also rendered (right column). This illustrates a two scene training process. (Epochs 20 to 120). In this example the training view scene is not the same as the validation view.






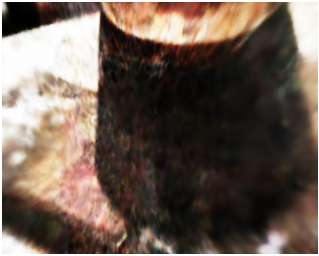


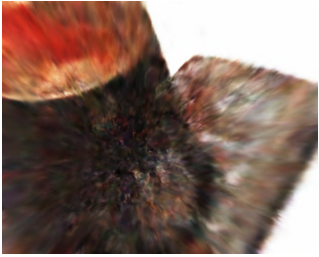






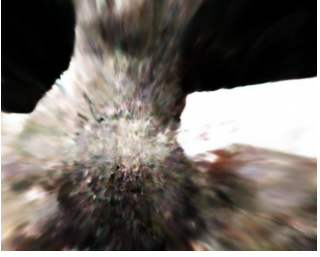


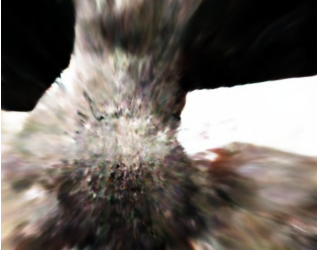


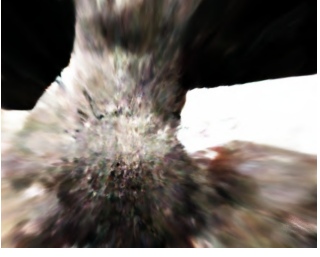
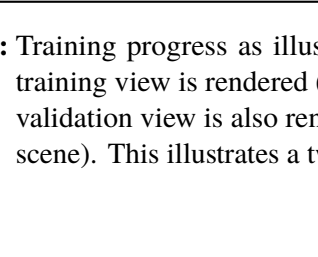
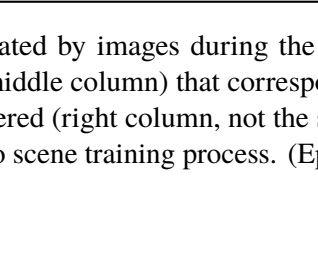
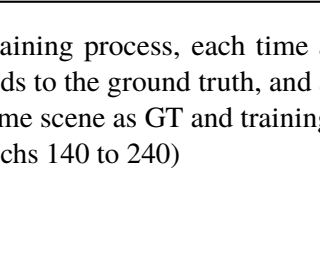



**Figure 6.2:** Training progress similar to Tabelle 6.1, but with eight scenes to train, rather than two.

### 6.2.1 Overfitting to input images

As the problem of scene geometry extraction gets more challenging the network tends to simply memorize training and reference data. This overfitting issue often results in a failure to learn the specific geometry and therefore a complete failure to generate novel views.

This issue can be addressed with a certain number of measures:

- *Standard network regularization via dropout during training and gradient clipping.* While this measure is certainly helpful, the effect on the network is only limited, especially since it does not help finding the actual geometries
- *Downsizing the Generative Model* While I observed no meaningful effect in changing the Recognition Model, the size and depth of the Generative Model is very important. A large Generative Model leads to overfitting, while a small number of parameters force the network to interpret the embedding given to the image, in order to extract useful geometric data.

Epoch	Ground-Truth	Training-View	Validation-View
140			
			
160			
			
180			
			
200			
			
220			
240			

**Table 6.2:** Training progress as illustrated by images during the training process, each time a training view is rendered (middle column) that corresponds to the ground truth, and a validation view is also rendered (right column, not the same scene as GT and training scene). This illustrates a two scene training process. (Epochs 140 to 240)

- *Adaption of Latent Size* While I only tested this parameter quantitatively, the results strongly suggest, that a smaller latent provides less overfitting, than a large one. So while I tried sizes 256, 512 and 1024, the smallest size seems to be the most sensible choice. The correct choosing of this parameter requires further consideration and experimentation, a coarse to fine approach might be beneficial here.
- *Division of the Dataset* While I initially only claimed two holdout images as validation data, now as a second measure the dataset is again split in reference views and training views, where reference views are input to the Recognition model and the training Models are held for the loss generation. While the last two frames in every dataset are training views the reference and training views are interleaved, assuring that they are equally distributed around the total inference space.

It remains open, if there is another way to regularize the network to prevent overfitting and still ensure a large enough size to preserve good rendering capabilities over a large set of scenes.

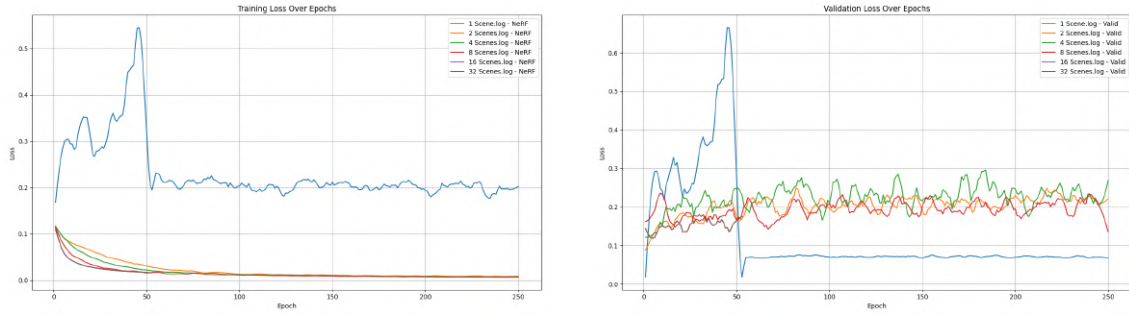
**Working with a small dataset** I further analyzed the problem by trying training with 2 frames (as reference, validation and training view shared), interestingly this helped extracting the correct geometry (with 16 scenes at the same time) not only directly in this position but also around the object. This suggests that the training problems stem, not from the model or dataset, but from complexity issues during the training procedure.

**Adding additional training steps** Since the training procedure, consistently seems too complex, to extract the meaningful geometry and the model always falls in a local minimum, either overfitting to the image data or getting stuck in a local minimum, there are several ways, how classical NeRFs can be aided to train the right way. I have tried using the following aids during the training in order to ensure a more consistent geometry prediction:

- *Depth Supervision* I used the depth supervision as described in chapter 5 in order to prevent overfitting to near data. This did prevent overfitting, but also did not help to converge to a sensible solution.
- *Occlusion Loss* As most of the overfitting happens very close to the near bound of each image (which is due to the lack of overlap of the training images in this region), the occlusion loss could be a sensible way, to encourage consistency by eliminating near camera predictions early in training. I tried to integrate this method, as described earlier, but observed the training-loss to become unstable very soon, in both DTU-MVS and LIFF.
- *Contrastive Training* While in theory, contrastive learning is a good way to generate meaningful and distinguishable per-scene priors, the NeRF loss seems to be unaffected by pretraining the Recognition Model as well as interleaved training of NeRF- and Contrastive-Epochs, while we can clearly observe a contrastive convergence in both cases, the NeRF loss seems unaffected, even with varying contrastive batch sizes and learning rates.

In summary I can say, that while it is certainly possible to train a network to extract and encode geometries from different scenes in a NeRF-Model, I have found no reliable training method for this.





**Figure 6.3:** Training loss (left) and Validation loss (right) with respect to a different number of scenes.

### 6.2.2 Validation and Rendering of Novel Views

There are several ways to monitor the progress of the training and evaluate the capabilities of the model to render novel views.

**Rendered images during Training** During the training process I rendered an validation and a training view to visually control for overfitting. The results are shown for the 2 scene training in Table 6.1 and for training with 8 scenes in Figure 6.2. As seen in the images there is still a lot of overfitting that I could not mitigate. Interestingly some of the validation views work better than others, with my hypothesis being that these are the places, where a training view is close by. We can also observe the „smearing“or „smudging“artifacts in the validation views, these probably result from overfitting to a near image plane, which is then viewed from the side.

Additionally to visual confirmation I also tracked training and validation loss for several examples, the training loss is shown in Figure 6.3, interestingly as I increased batch size to accommodate a multi scene setting the original training procedure sometimes becomes instable and fails to produce any solution at all. Also for reference, the validation loss is shown.

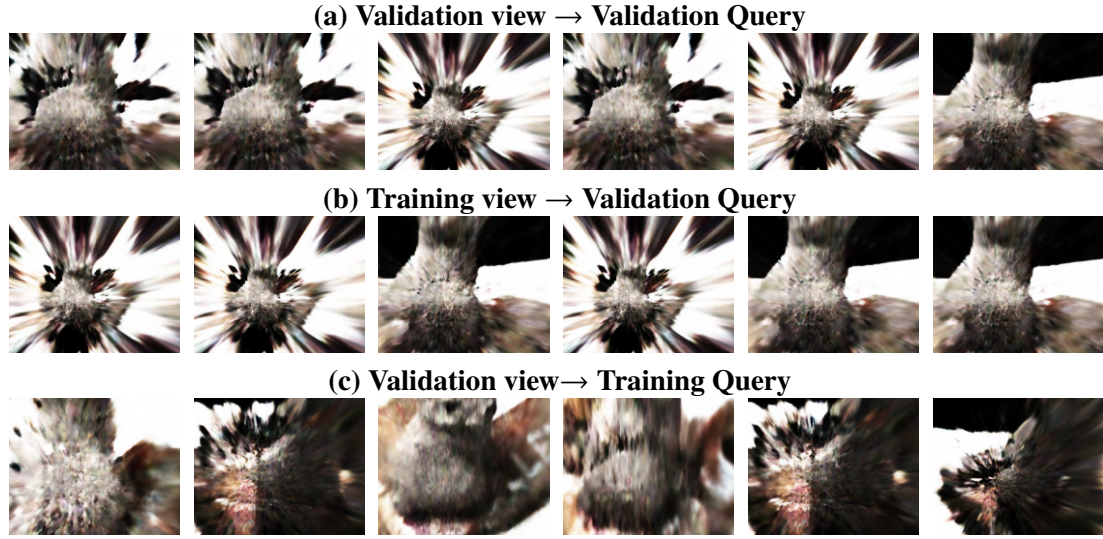
**Rendered images during Testing** After the training, we perform several tests to further determine what parts of the current model creates certain properties of the results. I test first properties within a scene and then across scenes.

Firstly I investigate the quality of input and query images from different sets as shown in Figure 6.5. The result of each scenario is documented with 6 images in Figure 6.4. As we can see there is no visible degradation from scenario (b), that uses a training view as input data to scenario (a) in which a validation view is used, this is a strong indicator, that the overfitting is happening mostly in the Rendering Model. We can see in scenario (c) that the model is still able to render useful, i.e. overfitted, training views from validation input data, which further indicates that overfitting happens in the rendering model.

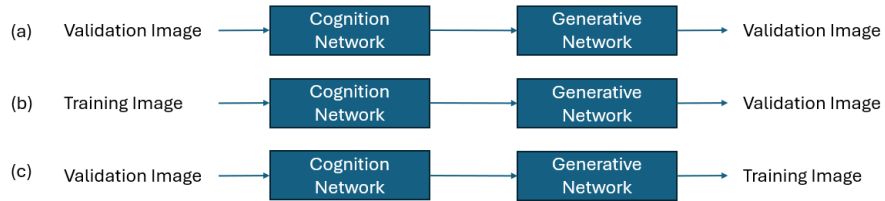
Secondly I investigate what effect the different components of the latent have, with the intention of finding out wether the Recognition Model produced a meaningful output, that is interpreted by the Generative Model.

As a first experiment I queried the bunny scene from DTU with a different - i.e. conflicting - latent components. In Figure 6.6a we can see, that in changing the latent representation does not affect the rendered scene, this is unsurprising, since we explicitly state another scene in the One-Hot-Encoding, which logically overrules the latent results from the Recognition model. We can see that vice versa Figure 6.6b changing the One-Hot-Encoding prompts the network to generate

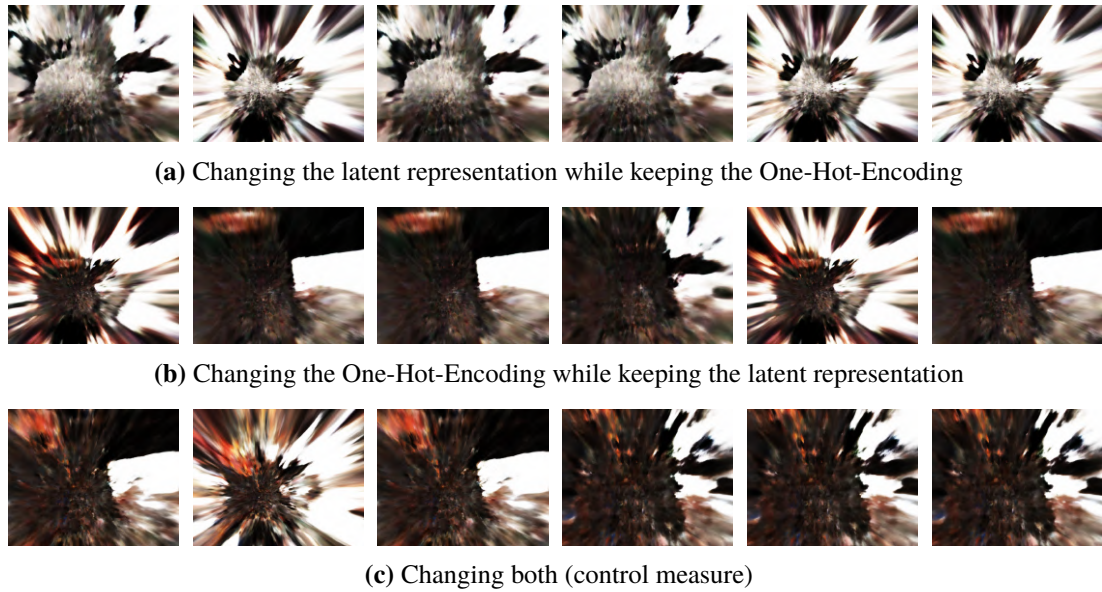




**Figure 6.4:** Comparison across three combinations of validations and training data to indicate causes of overfitting.



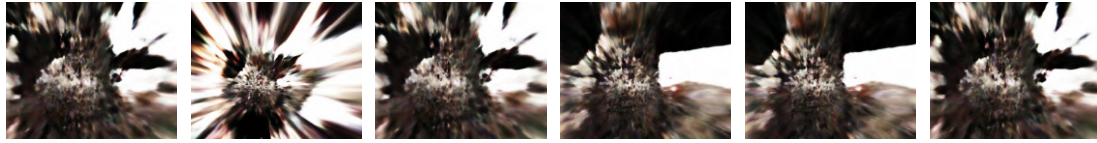
**Figure 6.5:** Different setups to test the position of overfitting.



**Figure 6.6:** Comparison of model renderings when prompted with inconsistent latent

another scene and as a control changing both also clearly changes the scene as seen in Figure 6.6c. This is what has to be expected.

In order to find out what effect the latent has on the rendering we can simply leave out the



(a) Rendering the bucket scene with no One-Hot-Encoding given.



(b) Rendering the bunny scene with no One-Hot-Encoding given.

**Figure 6.7:** Comparison of model renderings when prompted with no scene encoding.

One-Hot-Encoding and see if the scenes are still rendered from different results. The results can be seen in Figure 6.7, they show that there is no real measurable effect of different latent

## 7 Results and Future Work

This final chapter summarizes the results of the proposed framework and the experiments and shows the observed strengths & weaknesses as well as the current problems that lead the current generalization to fail. Moreover it suggest how to divide up and approach each part of this challenge in future works on its own.

### 7.1 Results

The training of NeRFs proved significantly more resource-intensive compared to single Scene training due to the increased number of images in the dataset. This is combined with the problem, that we have a rather big recognition network and a small generative network, which means many gradients have to be stored simultaneously, requires careful handling of memory and limited ray sampling possibilities.

Initial efforts where aimed towards improving distinguishability between scenes, I found that the most (self-) supervised training approaches fail to establish a distinguishable latent and rather adopt a averaged mean of the scene. This is possibly due to a complex task and the postprocessing of the results, that make it harder for the model to learn concrete examples.

While a explicit embedding helps to distinguish the scene during training processing the, it is does not particularly aid in the buildup of implicit features that help render the current scene. Here a more extensive look in pretrained recognition models and the differences of the current examples could be interesting. Pretraining could reach from unsupervised clustering, autoencoding of different images or even some enhanced consistency prediction as in [22], may help.

The second problem that was investigated was the overfitting issues and the connected extraction of scene geometry. I observed that the problem was mainly due to the low overlap in the near plane / regions of the training material. This lead to most predictions not being consistently around all geometries. I showed that this is a problem of the generative model, that is arises firstly from training with multi scene images. We can mitigate this issue by reducing the number and size of the layers and therefore forcing the network to actually learn the generalization rules. Nevertheless, this mostly works with the objects itself, while plain areas such as the white and black background may be anywhere and is simplified to the near plane in the model. While initial tries with depth guidance or occlusion loss fail mostly due to training instabilities, this is still a promising approach that could be analyzed further. More interestingly, superfitting the model to exactly two nearby images, yielded the best result with regard to extraction of geometry of my training (without actually being included in the experiments), which further suggests that this is mostly a complexity issue.

### 7.2 Future Work

This section outlines a possible path further develop this technology and improve the generalization and visual performance of novel scenes and views. With generalizable NeRFs we often rely on indirect training, a lot of parameters to develop for each Network, and resource intensive training, so the main way to facilitate further research includes splitting this topic into several issues that can be tackled separately.

One intermediary problem to tackle, is how we can regularize a network to memorize different scenes and recall them by a key, that is represented in a explicit scene embedding, such as the One-Hot-Encoding. The focus here should be on finding training methods and tuning of NeRF training-parameters such that the extraction of multiple complex geometries based on images is facilitated in a robust manner.

One more step to further inspect the models as presented in chapter 5, especially the more robust form of training using a latent storage that refines a latent using different training images.

In a parallel step the generation of good encodings, that can be extracted from several scene images and encode not only the images itself but the relations between the images and the geometries that are depicted in there, needs to be explored. As shown in this work, the sole extraction with guided NeRFs by other components, or contrastive learning for a converging scene encoding alone do not suffice. In this task, maybe some other pretext tasks such as clustering or auto-encoding, combined with new priors, including depth or image position relations might help here.

## Bibliography

- [1] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, P. P. Srinivasan. “Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields”. In: *CoRR* abs/2103.13415 (2021). arXiv: 2103.13415. URL: <https://arxiv.org/abs/2103.13415> (cit. on pp. 14, 15, 38, 41).
- [2] C. Cao, X. Ren, Y. Fu. “MVSFormer: Multi-View Stereo by Learning Robust Image Features and Temperature-based Depth”. In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. URL: <https://openreview.net/forum?id=2VWR6JfwNo> (cit. on p. 24).
- [3] C. Cao, X. Ren, Y. Fu. “MVSFormer++: Revealing the Devil in Transformer’s Details for Multi-View Stereo”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: <https://openreview.net/forum?id=wXWfvSpYHh> (cit. on pp. 24, 25, 36).
- [4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, F. Yu. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR]. URL: <https://arxiv.org/abs/1512.03012> (cit. on p. 24).
- [5] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, H. Su. “Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 14124–14133 (cit. on pp. 19, 21).
- [6] T. Chen, S. Kornblith, M. Norouzi, G. Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *arXiv preprint arXiv:2002.05709* (2020) (cit. on p. 16).
- [7] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017 (cit. on p. 23).
- [8] K. Deng, A. Liu, J.-Y. Zhu, D. Ramanan. “Depth-supervised NeRF: Fewer Views and Faster Training for Free”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 12872–12881. doi: 10.1109/CVPR52688.2022.01254 (cit. on pp. 16, 39).
- [9] K. Friston. “Friston, K.J.: The free-energy principle: a unified brain theory? Nat. Rev. Neurosci. 11, 127-138”. In: *Nature reviews. Neuroscience* 11 (Feb. 2010), pp. 127–38. doi: 10.1038/nrn2787 (cit. on pp. 33, 34).
- [10] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, J. Li. *NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review*. 2023. arXiv: 2210.00379 [cs.CV]. URL: <https://arxiv.org/abs/2210.00379> (cit. on p. 13).
- [11] B. G. A. Gerats, J. M. Wolterink, I. A. M. J. Broeders. “Dynamic Depth-Supervised NeRF for Multi-view RGB-D Operating Room Videos”. In: *Predictive Intelligence in Medicine*. Ed. by I. Rekik, E. Adeli, S. H. Park, C. Cintas, G. Zamzmi. Cham: Springer Nature Switzerland, 2023, pp. 218–230. ISBN: 978-3-031-46005-0 (cit. on pp. 16, 39).

- [12] J. Gornet, M. Thomson. “Automated construction of cognitive maps with visual predictive coding”. In: *Nature Machine Intelligence* 6.7 (July 2024), pp. 820–833. ISSN: 2522-5839. DOI: [10.1038/s42256-024-00863-1](https://doi.org/10.1038/s42256-024-00863-1). URL: <https://doi.org/10.1038/s42256-024-00863-1> (cit. on p. 35).
- [13] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu koray, R. Munos, M. Valko. “Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 21271–21284. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf) (cit. on p. 17).
- [14] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, H. Aanæs. “Large scale multi-view stereopsis evaluation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2014, pp. 406–413 (cit. on pp. 23, 29).
- [15] M. M. Johari, Y. Lepoittevin, F. Fleuret. “GeoNeRF: Generalizing NeRF With Geometry Priors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 18365–18375 (cit. on pp. 19, 22).
- [16] K. Li, T. Rolff, S. Schmidt, R. Bacher, W. Leemans, F. Steinicke. “Interacting with Neural Radiance Fields in Immersive Virtual Reality”. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI EA ’23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9781450394222. DOI: [10.1145/3544549.3583920](https://doi.org/10.1145/3544549.3583920). URL: <https://doi.org/10.1145/3544549.3583920> (cit. on p. 12).
- [17] Y. Liu, S. Peng, L. Liu, Q. Wang, P. Wang, C. Theobalt, X. Zhou, W. Wang. “Neural Rays for Occlusion-aware Image-based Rendering”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2022, pp. 7814–7823. DOI: [10.1109/CVPR52688.2022.00767](https://doi.org/10.1109/CVPR52688.2022.00767). URL: <https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00767> (cit. on pp. 19, 23).
- [18] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, D. Duckworth. *NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections*. 2021. arXiv: [2008.02268](https://arxiv.org/abs/2008.02268) [cs.CV]. URL: <https://arxiv.org/abs/2008.02268> (cit. on p. 12).
- [19] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 24).
- [20] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, A. Kar. “Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines”. In: *ACM Transactions on Graphics (TOG)* (2019) (cit. on pp. 23, 31).
- [21] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: [2003.08934](https://arxiv.org/abs/2003.08934) [cs.CV]. URL: <https://arxiv.org/abs/2003.08934> (cit. on pp. 13, 14, 23, 29, 38).
- [22] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. M. Sajjadi, A. Geiger, N. Radwan. “RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022 (cit. on pp. 39, 51).



- [23] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, A. Geiger. “Convolutional Occupancy Networks”. In: *Computer Vision – ECCV 2020*. Ed. by A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm. Cham: Springer International Publishing, 2020, pp. 523–540. ISBN: 978-3-030-58580-8 (cit. on p. 24).
- [24] R. P. N. Rao, D. H. Ballard. “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects”. In: *Nature Neuroscience* 2.1 (Jan. 1999), pp. 79–87. ISSN: 1546-1726. DOI: [10.1038/4580](https://doi.org/10.1038/4580). URL: <https://doi.org/10.1038/4580> (cit. on p. 35).
- [25] A. Rau, J. Aklilu, F. Christopher Holsinger, S. Yeung-Levy. “Depth-Guided NeRF Training via Earth Mover’s Distance”. In: *Computer Vision – ECCV 2024*. Ed. by A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, G. Varol. Cham: Springer Nature Switzerland, 2025, pp. 1–17. ISBN: 978-3-031-73039-9 (cit. on p. 16).
- [26] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordon, P. Labatut, D. Novotny. “Common Objects in 3D: Large-Scale Learning and Evaluation of Real-Life 3D Category Reconstruction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 10901–10911 (cit. on p. 23).
- [27] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, R. Ng. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *CoRR* abs/2006.10739 (2020). arXiv: [2006.10739](https://arxiv.org/abs/2006.10739). URL: <https://arxiv.org/abs/2006.10739> (cit. on p. 14).
- [28] A. Trevithick, B. Yang. “Grf: Learning a general radiance field for 3d representation and rendering”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15182–15192 (cit. on pp. 19, 20).
- [29] C. Wang, J. Sun, L. Liu, C. Wu, Z. Shen, D. Wu, Y. Dai, L. Zhang. “Digging into Depth Priors for Outdoor Neural Radiance Fields”. In: *Proceedings of the 31st ACM International Conference on Multimedia*. MM ’23. Ottawa ON, Canada: Association for Computing Machinery, 2023, pp. 1221–1230. ISBN: 9798400701085. DOI: [10.1145/3581783.3612306](https://doi.org/10.1145/3581783.3612306). URL: <https://doi.org/10.1145/3581783.3612306> (cit. on p. 16).
- [30] G. Wang, L. Pan, S. Peng, S. Liu, C. Xu, Y. Miao, W. Zhan, M. Tomizuka, M. Pollefeys, H. Wang. “NeRF in Robotics: A Survey”. In: *ArXiv* abs/2405.01333 (2024). URL: <https://api.semanticscholar.org/CorpusID:269501937> (cit. on p. 12).
- [31] Q. Wang, S. Guo, H. Wu, R. Xie, L. Song, W. Zhang. “NeRF-SDP: Efficient Generalizable Neural Radiance Field with Scene Depth Perception”. In: *Proceedings of the 5th ACM International Conference on Multimedia in Asia*. MMAsia ’23. Tainan, Taiwan: Association for Computing Machinery, 2024. ISBN: 9798400702051. DOI: [10.1145/3595916.3626380](https://doi.org/10.1145/3595916.3626380). URL: <https://doi.org/10.1145/3595916.3626380> (cit. on p. 16).
- [32] H. Yang, L. Hong, A. Li, T. Hu, Z. Li, G. H. Lee, L. Wang. “Contranerf: Generalizable neural radiance fields for synthetic-to-real novel view synthesis via contrastive learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16508–16517 (cit. on p. 20).
- [33] J. Yang, M. Pavone, Y. Wang. *FreeNeRF: Improving Few-shot Neural Rendering with Free Frequency Regularization*. 2023. arXiv: [2303.07418](https://arxiv.org/abs/2303.07418) [cs.CV]. URL: <https://arxiv.org/abs/2303.07418> (cit. on pp. 15, 38, 39).

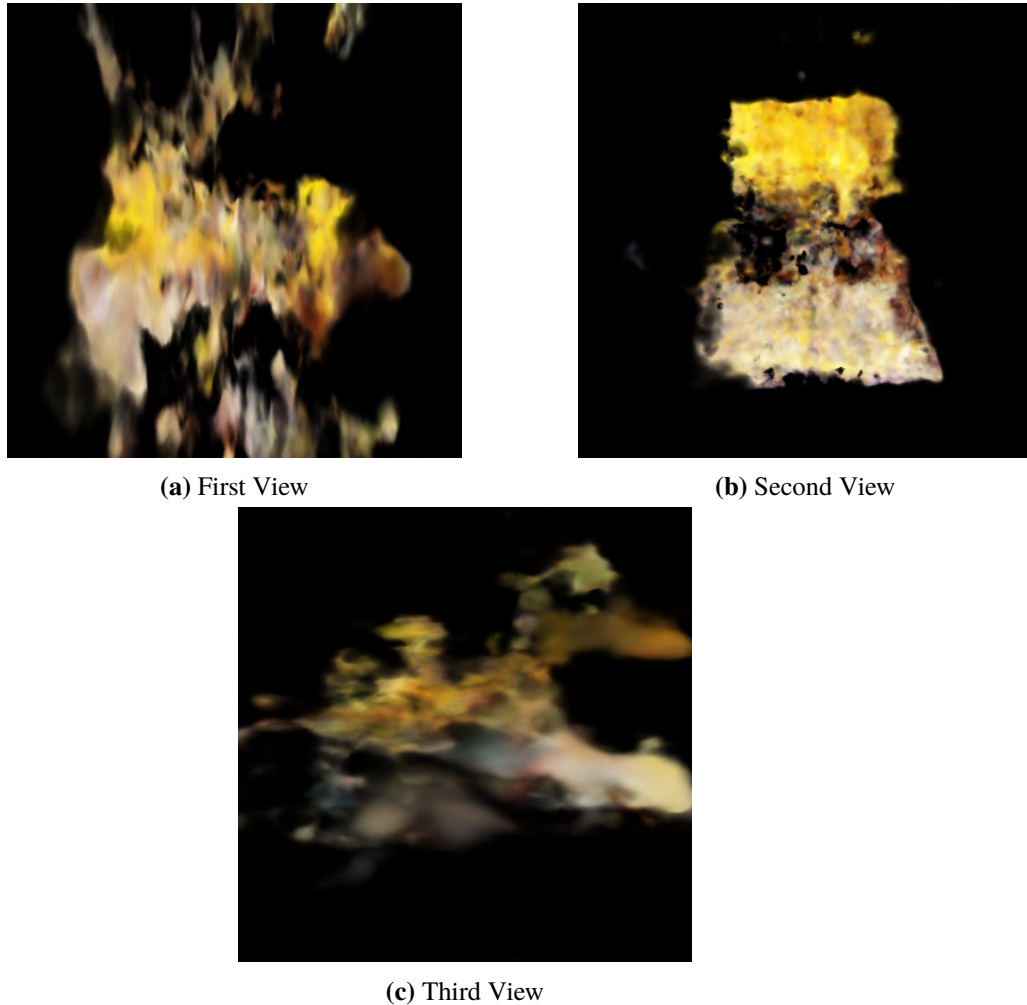
- [34] Y. Yao, Z. Luo, S. Li, T. Fang, L. Quan. “Mvsnet: Depth inference for unstructured multi-view stereo”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 767–783 (cit. on p. 22).
- [35] K. Ye, H. Wu, X. Tong, K. Zhou. *A Real-time Method for Inserting Virtual Objects into Neural Radiance Fields*. 2023. arXiv: [https://arxiv.org/help/api/index={2310.05837}\[cs.CV\]](https://arxiv.org/help/api/index={2310.05837}[cs.CV]). URL: <https://arxiv.org/abs/2310.05837> (cit. on p. 12).
- [36] A. Yu, V. Ye, M. Tancik, 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 (cit. on p. 19).
- [37] X. Zhang, S. Bi, K. Sunkavalli, H. Su, Z. Xu. “NeRFusion: Fusing Radiance Fields for Large-Scale Scene Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 5449–5458 (cit. on p. 23).
- [38] F. Zhu, S. Guo, L. Song, K. Xu, J. Hu. “Deep Review and Analysis of Recent NeRFs”. In: *APSIPA Transactions on Signal and Information Processing* (2023). URL: <https://api.semanticscholar.org/CorpusID:257420578> (cit. on p. 12).



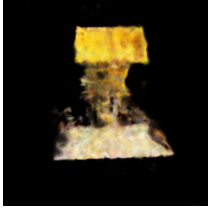
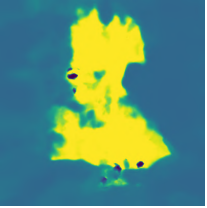

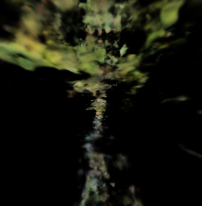

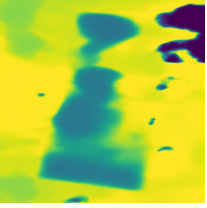
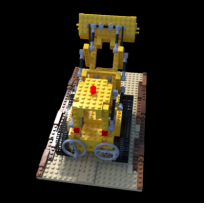
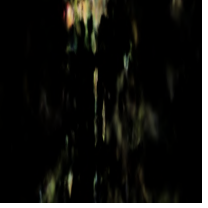

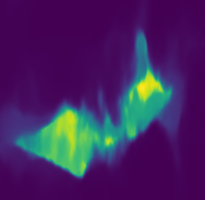

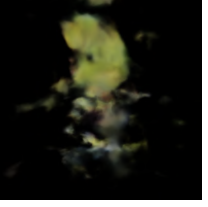
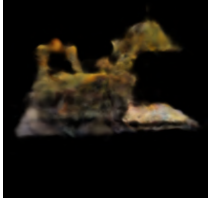
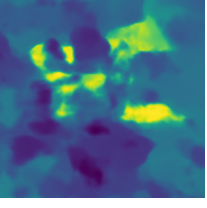

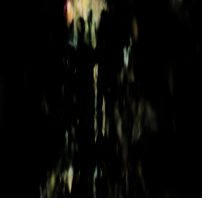

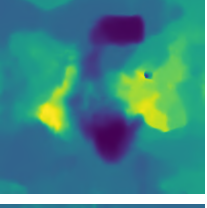

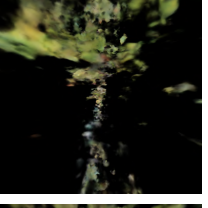
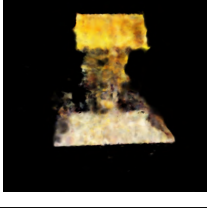
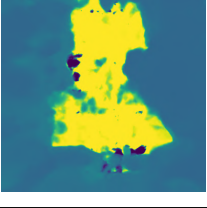
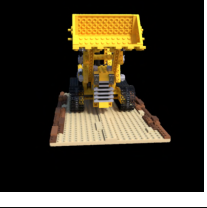
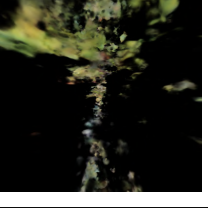
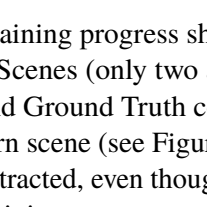
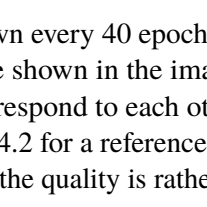
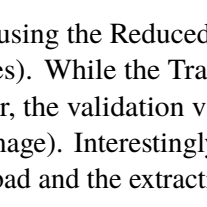
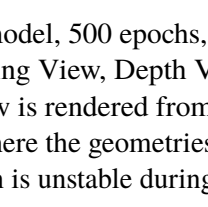
## A Appendix

The appendix includes different illustrations from training.

Tabelle A.1 shows the training using the Blender Dataset, as you can see during the process the geometry gets somewhat extracted by the Rendering Network. The validation views of the fern on the other hand are not as good as the trained positions. As seen in Figure A.1 the validation views of the lego scene show that there are regions from which a good image can be rendered and there are regions where rendering still fails due to wrong geometry extraction.



**Figure A.1:** The validation views of the lego scene from blender

Epoch	Training View	Depth View	Ground Truth	Validation View
280				
				
360				
				
440				
				
480				

**Table A.1:** Training progress shown every 40 epochs, using the Reduced model, 500 epochs, and 8 Scenes (only two are shown in the images). While the Training View, Depth View and Ground Truth correspond to each other, the validation view is rendered from the fern scene (see Figure 4.2 for a reference image). Interestingly here the geometries get extracted, even though the quality is rather bad and the extraction is unstable during the training process.

### **Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature