VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MACHINE LEARNING (CO3117)

Assignment report

SEMANTICALLY CONSISTENT FEW-SHOT VIEW SYNTHESIS

Advisor: Nguyễn Đức Dũng
Students: Vũ Nguyễn Lan Vi - 2153094

HO CHI MINH CITY, JANUARY 2024

Contents

Chapter 1

Introduction

Gaining insight into the 3D structure of a natural scene is a critical step for numerous advanced computer vision applications such as object recognition, realistic rendering, autonomous driving, and virtual reality. While recent strides in deep learning have empowered high-quality 3D reconstruction and recognition, learning from 3D data poses inherent challenges compared to its 2D image-based counterpart. These challenges stem from the unstructured nature of data format, high memory requirement, and lack of principled architectures. Consequently, many researchers investigate the standard models with appropriate training algorithms and the methods for reducing their computational cost, while tackling a range of complex tasks.

In recent times, **Neural Radiance Fields (NeRF)** have emerged as a robust representation of photorealistic novel view synthesis given many images, up to 100 for challenging 360-degree scenes. However, a commonly observed failure mode of NeRF-based methods is fitting incorrect geometries in cases of insufficient input views and noisy camera poses, thus limiting their application in real-world scenarios, particularly when working with a constrained training dataset. One potential reason is that standard volumetric rendering does not enforce the constraint that most of a scene's geometry consist of empty space and opaque surfaces.

Several studies have sought to address the challenging few-shot neural rendering problem by leveraging extra information. One approach involves employing external models to acquire normalization-flow regularization, perceptual regularization, and cross-view semantic consistency. Another thread of work attempts to learn transferable models by training on a large, curated dataset instead of using an external model. Recent studies emphasize the significance of geometry in few-shot neural rendering and propose geometry regularization to enhance performance. However, these methods require expensive pre-training on tailored multi-view datasets or costly training-time patch rendering, introducing significant overhead in methodology, engineering implementation, and training budgets.

In this study, I'll present three highly efficient approaches that substantially reduce additional computational overhead with minimal adjustments while delivering similar performance to other complicated methods in the few-shot setting. These approaches involve: regularizing the frequency range of NeRF's inputs, imposing entropy constraints on density within each ray, and leveraging additional supervision from depth.

Chapter 2

Preliminaries

Neural Radiance Fields (NeRF) employ a multilayer perceptron (MLP) to encode a scene, mapping 3D positions and directions to densities and radiance. This encoding enables the synthesis of diverse and arbitrary new perspectives using volumetric rendering techniques. Typically this representation is trained per scene with a loss measuring photometric consistency with respect to a collection of posed RGB images. If the input images are dense and diverse enough, the scene is small enough, the camera poses are accurate enough, the camera exposure parameters are constant, and the scene is static, the original NeRF model can synthesize remarkably detailed and accurate novel views.
Neural Radiance Fields employ a neural network with parameters θ to model a volumetric scene based on a collection of arranged images $\{I_i\}_{i=1}^{N}$; i.e. with known intrinsic and extrinsic calibrations. When rendering an image, NeRF employs ray marching to sample the volumetric radiance field. It combines the sampled density and color to generate the incoming radiance for a specific ray., and supervises the training of $\boldsymbol{\theta}$ by an L2 photometric reconstruction loss:

$$\mathcal{L}_{\text{rgb}}(\boldsymbol{\theta}) = \sum_i \mathbb{E}_{\mathbf{r}\sim I_i}\left[\left\|\boldsymbol{C}(\mathbf{r}) - \boldsymbol{C}_i^{\text{gt}}(\mathbf{r})\right\|_2^2\right]$$

where $C_i^{\text{gt}}(\mathbf{r})$ is the ground truth color of ray $\mathbf{r}$ passing through a pixel in image i, and the color $\boldsymbol{C}(\mathbf{r})$ is computed by integrating the weighted volumetric radiance within the ray's near and far bounds t_n and t_f :

$$\boldsymbol{C}(\mathbf{r}) = \int_{t_n}^{t_f} w(t) \cdot \underbrace{\mathbf{c}(t)}_{\text{radiance}} \, dt$$

and $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ represents a ray with camera origin $\mathbf{o}$ oriented as $\mathbf{d}$, with volume rendering integration weights:

$$w(t) = \underbrace{\exp\left(-\int_{t_n}^{t} \sigma(s)ds\right)}_{\text{visibility of } \mathbf{r}(t) \text{ from } \mathbf{o}} \cdot \underbrace{\sigma(t)}_{\text{density at } \mathbf{r}(t)}$$

while the intermediate features $\mathbf{z}(t)$, the volumetric density $\sigma(t)$ and view-dependent radiance fields $\mathbf{c}(t)$ are stored within the parameters $\boldsymbol{\theta}$ of fully connected neural networks:

$$\mathbf{z}(t) = \mathbf{z}(\mathbf{r}(t); \boldsymbol{\theta}) : \mathbb{R}^3 \rightarrow \mathbb{R}^z$$
$$\sigma(t) = \sigma(\mathbf{z}(t); \boldsymbol{\theta}) : \mathbb{R}^z \rightarrow \mathbb{R}^+$$
$$\mathbf{c}(t) = \mathbf{c}(\mathbf{z}(t), \mathbf{d}; \boldsymbol{\theta}) : \mathbb{R}^z \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

To better represent high frequency details in the scene, positional encoding is applied to the inputs (More details in **Section 3.1.1**), and two stages of sampling are performed: In the first stage, the points are sampled uniformly while, in the second stage, the importance sampling is performed based on the density estimated in the first stage.

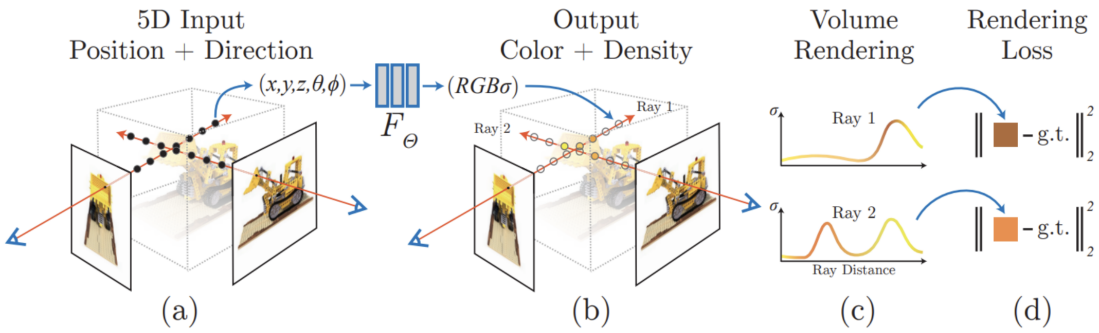

Figure 2.1: **The NeRF volume rendering and training process.** (a) illustrates the selection of sampling points for individual pixels in a to-be-synthesized image. (b) illustrates the generation of densities and colors at the sampling points using NeRF MLP(s). (c) and (d) illustrate the generation of individual pixel color(s) using in-scene colors and densities along the associated camera ray(s) via volume rendering, and the comparison to ground truth pixel color(s), respectively.

Chapter 3

Methods

3.1 Enhancing Few-shot Neural Rendering through Free Frequency Regularization

Neural Radiance Fields (NeRF) often face overfitting to the views seen during training and struggle when generating new views with limited input data. To tackle this, existing methods employ diverse strategies like pre-training on extensive multi-view datasets or introducing estimated depth as an additional signal. However, these approaches lead to complex training pipelines and computational overhead. Yang et al. [2]identified a crucial aspect in few-shot neural rendering: the role of frequency in NeRF's training process.
The challenge lies in the overfitting issue, particularly with high-frequency inputs. Higher-frequency data allows faster convergence for those components, yet this quick convergence restricts NeRF from exploring low-frequency information. Consequently, NeRF becomes biased towards unwanted high-frequency artifacts and noise, hindering its ability to learn cohesive geometry.

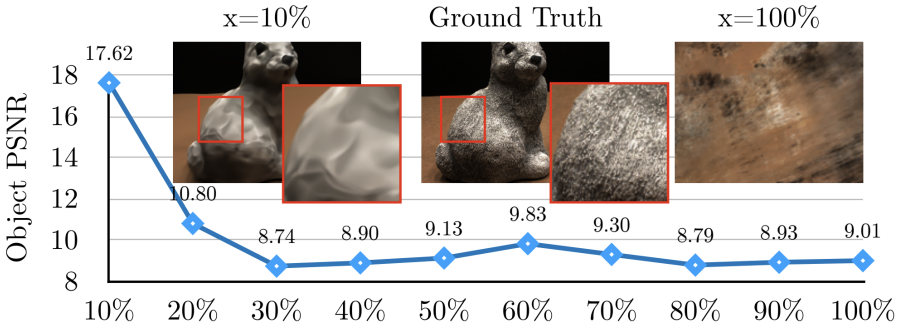

Figure 3.1: ***Visible positional encoding ratio x.*** *Examining NeRF's performance with positional encodings across various masking ratios using the DTU dataset and three input views. Despite its over-smoothness, the plain NeRF succeeds in the few-shot setting when only low-frequency inputs are visible.*

Let's explore an experiment highlighting how high frequencies affect the performance of the Nerf model. We introduced masks for high-frequency positional encoding inputs, allowing the model to focus on learning low frequencies. By setting certain positional encodings to zero based on a visible ratio and the length of the encoded coordinates, we observed a notable impact on NeRF's performance.
When assessing the DTU dataset with three input views, exposing the model to higher-frequency inputs resulted in a considerable decline in NeRF's performance. For instance, when utilizing only 10% of the total embedding bits, mipNeRF achieved a high PSNR of 17.62, surpassing plain NeRF, which achieved a PSNR of only 9.01 at 100% visible ratio (See **Figure 3.1**). This emphasizes that in few-shot scenarios, models trained on low-frequency inputs tend to offer superior representations. However, these models often produce over-smoothened images.

3.1.1 Frequency Regularization

- **Original positional encoding method:** Using sinusoidal functions with different frequencies to map the inputs into a higher-dimensional space.

$$\gamma_L(\mathbf{x}) = \left[\sin(\mathbf{x}), \cos(\mathbf{x}), \ldots, \sin\left(2^{L-1}\mathbf{x}\right), \cos\left(2^{L-1}\mathbf{x}\right)\right]$$

  Where L represents a hyper-parameter that regulates the maximum encoded frequency and might vary for coordinates x and directional vectors d. A common practice is to concatenate the raw inputs with the frequency-encoded inputs as follows:

$$\mathbf{x}' = [\mathbf{x}, \gamma_L(\mathbf{x})]$$

- **Applying frequency regularization method:** With a positional encoding of length L + 3, a linearly increasing frequency mask α is employed to control the visible frequency spectrum according to the training time steps in the following manner:

$$\gamma'_L(t, T; \mathbf{x}) = \gamma_L(\mathbf{x}) \odot \boldsymbol{\alpha}(t, T, L)$$

  , where:

$$\boldsymbol{\alpha}_i(t, T, L) = \begin{cases} 1 & \text{if } i \leq \frac{t \cdot L}{T} + 3 \\ \frac{t \cdot L}{T} - \left\lfloor \frac{t \cdot L}{T} \right\rfloor & \text{if } \frac{t \cdot L}{T} + 3 < i \leq \frac{t \cdot L}{T} + 6 \\ 0 & \text{if } i > \frac{t \cdot L}{T} + 6 \end{cases}$$

  $\alpha_i(t, T, L)$ represents the i-th bit value of $\alpha(t, T, L)$; where t and T refer to the current training iteration and the final iteration of frequency regularization, respectively. Specifically, the method starts with raw inputs devoid of positional encoding, gradually increasing the visible frequency by 3 bits at each training step. This process is succinctly captured in a single line of code, as shown in **Figure 3.2**. Frequency regularization addresses the instability arising from high-frequency signals at the beginning of training. It progressively introduces high-frequency information to NeRF, avoiding over-smoothness. The work focuses on the few-shot neural rendering problem, revealing the catastrophic failure patterns caused by

high-frequency inputs and outperforming previous methods with minimal modifications to plain NeRF.

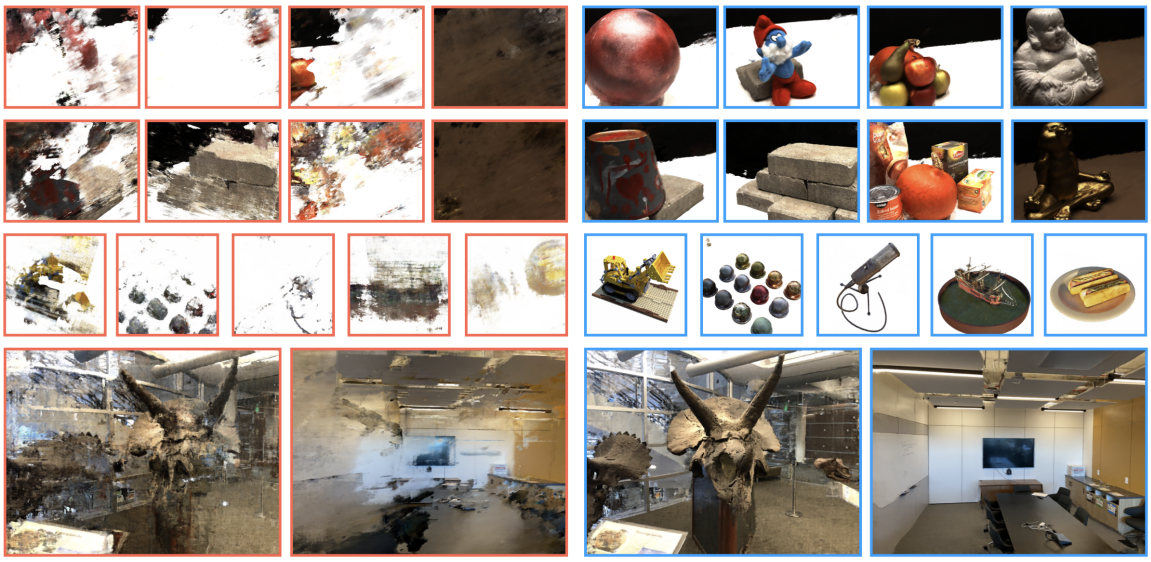

Figure 3.2: *Turning the left to the right by adding one line of code:* `pos_enc[int(t/T*L)+3:]=0`

3.1.2 Occlusion Regularization

Given the limited training views and the inherently ill-posed nature of the problem, certain distinct artifacts might persist in novel views. These issues typically materialize as "walls" or "floaters" positioned extremely close to the camera, as depicted in the left image of **Figure 3.2**. In such scenarios, a NeRF model might interpret these unexplored areas as densely packed volumetric floaters situated near the camera.
These floaters stem from areas with minimal overlap between imperfectly captured training views. These regions pose challenges in estimating their geometry due to severely restricted information (such as one-shot scenarios), often resulting in their prediction as "white walls" near the camera.
To address this, occlusion regularization is employed to penalize dense fields located in close proximity to the camera. This regulation is defined by adding an **occlusion regularization loss** function as follows:

$$\mathcal{L}_{occ} = \frac{\boldsymbol{\sigma}_K^\top \cdot \mathbf{m}_K}{K} = \frac{1}{K}\sum_K \sigma_k \cdot m_k$$

Here, $\mathbf{m}_K$ represents a binary mask vector determining whether a specific point will be penalized. Meanwhile, σ_k indicates the density values of K sampled points along the ray, ordered from closest to farthest from the origin. To reduce solid floaters near the camera, setting the values of $\mathbf{m}_K$ up to index M, termed as regularization range, to 1 and the rest to 0.

3.2 Improving Few-shot Neural Rendering with Ray Entropy Minimization

In the line of the research, efforts are made to address potential reconstruction inconsistencies arising from limited viewpoints and the risk of degeneracy when training images come from nearly identical viewpoints. Several prior works have tackled this challenge. Certain approaches utilize features extracted from observed images to compensate for missing information in unseen views, often focusing on specific object classes, such as humans, in novel view synthesis. However, they either work barely with few examples or require narrow baseline assumptions to find correspondences using an external module.
Kim et al. [3]. introduced a method that operates solely with a few input images and lacks prior scene knowledge. They proposed information-theoretic regularizations applied to NeRF-based models to mitigate reconstruction inconsistencies caused by the scarcity of input views and the potential degeneracy resulting from overfitting (See results in **Figure 3.3**).

Figure 3.3: **A qualitative comparison** of InfoNeRF (employing Ray Entropy Minimization) with other NeRF-based models on the Lego, Materials, and Ficus scenes in 4-view setting. Existing works often suffer from noise (b), color distortion (c), or blur effect (d), while InfoNeRF achieves outstanding quality of rendered images and estimated depth maps with clear boundaries and fine details.

3.2.1 Regularization by Ray Entropy Minimization

To tackle the issue of reconstruction inconsistency, they introduce a sparsity constraint on the reconstructed scene by minimizing the entropy of each ray. To achieve this, a regularization term í included in the loss function. This term ensures that the density of a ray remains low in entropy.

This constraint is logical because along a ray, only a small subset of sampled points typically intersect with objects or the background in a scene. The remaining points are more prone to capturing noise.

- **Ray density:** Normalized ray density denoted by $p(\mathbf{r})$ is defined as follows:

$$p(\mathbf{r}_i) = \frac{1 - \exp(-\sigma_i \delta_i)}{\sum_j 1 - \exp(-\sigma_j \delta_j)}$$

  where r_i (i = 1, . . . , N) is a sampled point in a ray, σ_i is the observed density at r_i , and σ_i is at sampling interval around r_i .
  NeRF employs a two-stage sampling strategy for points along a ray. Initially, it draws samples from a uniform distribution, followed by sampling from the opacity density, denoted by $\alpha_i \equiv 1 - \exp(-\sigma_i \delta_i)$. The second process involves computing the normalized opacity at each sampled point and subsequently estimating its normalized density along the ray since the opacity is closely related to the occupancy of a scene and the rendered color corresponding to the ray.

- **Ray entropy:** In alignment with Shannon Entropy principles, the entropy for a discrete ray density function as defined by:

$$H(\mathbf{r}) = -\sum_{i=1}^{N} p(\mathbf{r}_i) \log p(\mathbf{r}_i)$$

- **Disregarding non-hitting rays:** In the context of ray entropy minimization, a concern arises where certain rays are enforced to have low entropy though they do not hit any objects in the scene. To counter the potential artifacts stemming from this, rays with low density will be excluded from the entropy minimization process.
  This is achieved formally by introducing a mask variable, denoted as M(·), which identifies the rays with sufficient scene observations. The determination of these rays relies on the opacity, characterized as follows:

$$M(\mathbf{r}) = \begin{cases} 1 & \text{if } Q(\mathbf{r}) > \epsilon \\ 0 & \text{otherwise} \end{cases},$$

  where

$$Q(\mathbf{r}) = \sum_{i=1}^{N} 1 - \exp(-\sigma_i \delta_i)$$

  denotes the cumulative ray density.

- **Ray entropy loss:** Based on the computed ray entropy computed, ray entropy minimization loss is defined as follows:

$$\mathcal{L}_{\text{entropy}} = \frac{1}{|\mathcal{R}_s| + |\mathcal{R}_u|} \sum_{\mathbf{r} \in \mathcal{R}_s \cup \mathcal{R}_u} M(\mathbf{r}) \odot H(\mathbf{r})$$

  where $\mathbf{R_s}$ represents a collection of rays derived from training images, while $\mathbf{R_u}$ denotes a set of randomly sampled rays from unseen images. The symbol $\odot$ denotes element-wise multiplication. It's important to note that NeRF-based models cannot leverage rays from unseen images due to the absence of their

pixel color ground-truths. However, models based on entropy methods can utilize these rays because entropy regularization doesn't rely on ground-truth data. Leveraging these rays, even from the unobserved viewpoint, proves beneficial for enhancing scene reconstruction.

3.2.2 Regularization by Information Gain Reduction

When all training images share similar viewpoints, models tend to overfit to these seen images, leading to a struggle in generalizing to unseen views. This limitation arises due to the lack of diverse observations, pushing the trained models toward degenerate or trivial solutions.
To overcome the aforementioned limitations, an extra regularization term is introduced to ensure a uniform density distribution among neighboring rays. This method involves taking an observed ray, $\mathbf{r}$, and sampling another ray, denoted by $\tilde{\mathbf{r}}$, with a slightly different viewpoint. The goal is to minimize the KL-divergence between the density functions of these two rays.
The motivation behind this objective is to promote consistency between observations from two similar views, enabling the model to generalize to nearby viewpoints. This is achieved by reducing the information gained when adding a new viewpoint:

$$\mathcal{L}_{\mathrm{KL}} = D_{\mathrm{KL}}(P(\mathbf{r}) \| P(\tilde{\mathbf{r}})) = \sum_{i=1}^{N} p\left(\mathbf{r}_i\right) \log \frac{p\left(\mathbf{r}_i\right)}{p\left(\tilde{\mathbf{r}_i}\right)}$$

where $\tilde{\mathbf{r}_i}$ is a sampled point for observation in ray $\tilde{\mathbf{r}}$, $\tilde{\mathbf{r}}$ is obtained by slightly rotating the camera pose of $\mathbf{r}$ in the range from $-5°$ to $5°$.

3.3 Fewer Views and Faster Training using Depth Supervision

One reason for this lies in the traditional volumetric rendering approach lacking a critical constraint: acknowledging that most of a scene consists of empty space and solid surfaces. Deng et al. [4] address this by introducing **DS-NeRF** (Depth-supervised Neural Radiance Fields), a framework that enhances radiance field learning using readily-accessible depth supervision. They capitalize on the fact that existing NeRF pipelines require images with known camera poses, often estimated via structure-from-motion (SFM) techniques. Notably, SFM also generates sparse 3D points, offering "free" depth supervision during training.
DS-NeRF incorporates a depth loss that encourages the distribution of a ray's termination to match the 3D keypoint, incorporating reprojection error as an uncertainty measure. This approach provides a much stronger signal compared to solely reconstructing RGB information. Fewer required training views and a training speed improvement of **2-3 times**.
As depth data becomes more accessible, being able to apply depth supervision becomes increasingly more powerful. A probabilistic formulation of depth supervision shows meaningful improvements to NeRF and its variations. Recent studies illustrate how depth extracted from sensors like time-of-flight cameras or RGB-D Kinect sensors can be effectively employed to fit implicit functions.

3.3.1 Motivation

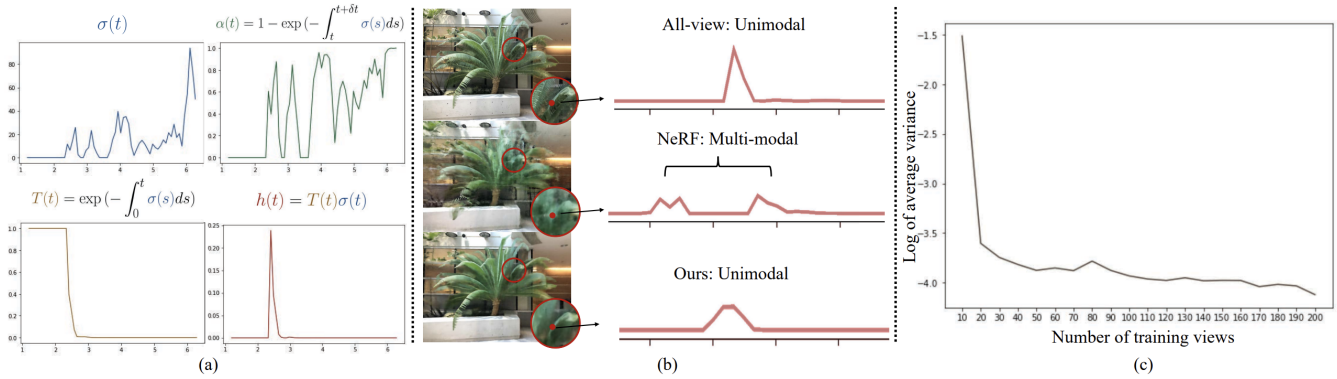

Figure 3.4: **Ray Termination Distribution.** (a) Visualizing various NeRF components over the distance traveled by the ray. Despite a ray passing through multiple objects (as shown by multiple density peaks in $\sigma(\mathbf{t})$, it is observed that the termination distribution $\mathbf{h}(\mathbf{t})$ remains unimodal. Additionally, NeRF models trained with sufficient supervision tend to have peaky, unimodal ray termination distributions as seen by the decreasing variance with more views in (c). This leads us to propose that the ideal ray termination distribution tends towards a Dirac function (δ).

- **Ray distribution:** Let's call $h(t) = T(t)\sigma(t)$ (where $T(t) = \exp\left(-\int_0^t \sigma(s)ds\right)$ checks for occlusions by integrating the differential density between 0 to t) is a continuous probability distribution over ray distance t that describes the likelihood of a ray terminating at t. Due to practical constraints, NeRFs assume that the scene lies between a near and far bound (**tn**, **tf**). To ensure **h(t)** sums to one, NeRF implementations often treat **tf** as an opaque wall. With this definition, the rendered color can be written as an expectation:

$$\hat{\mathbf{C}} = \int_0^\infty h(t)\mathbf{c}(t)dt = \mathbb{E}_{h(t)}[\mathbf{c}(t)]$$

- **Idealized distribution:** The distribution **h(t)** describes how much the sampled radiance along a ray contributes to the final rendered result. In many scenes, there are vast empty areas and solid surfaces that limit this contribution, emphasizing the significance of the nearest surface. This implies that the perfect distribution of rays for an image point nearest to a surface at depth D should resemble a Dirac function $\delta(t - \mathbf{D})$. In **Figure 3.4**(c), it is noticed that as the number of training views increases, the empirical variance of NeRF termination distributions decreases. This suggests that high-quality NeRF models (trained with numerous views) tend to approach ray distributions resembling the Dirac function. This observation motivates an implementation of depth-supervised ray termination loss.

3.3.2 Deriving depth-supervision

- **Depth uncertainty:** Structure From Motion technique returns a set of 3D keypoints $\{\mathbf{X} : \mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathbb{R}^3\}$ and visibility flags for which keypoints are seen from camera $j : \mathbf{X}_j \subset \mathbf{X}$. Given image I_j and its camera $\mathbf{P}_j$,

we estimate the depth of visible keypoints $\mathbf{x}_i \in \mathbf{X}_j$ by simply projecting $\mathbf{x}_i$ with $\mathbf{P}_j$, taking the re-projected z value as the keypoint's depth $\mathbf{D}_{ij}$ (See **Figure 3.5**). As expected, $\mathbb{D}_{ij}$ are inherently noisy estimates due to spurious correspondences, noisy camera parameters. The reliability of a particular keypoint $\mathbf{x}_i$ can be measured using the average reprojection error $\hat{\sigma}_i$ across views over which the keypoint was detected. Specifically, we model the location of the first surface encountered by a ray as a random variable $\mathbb{D}_{ij}$ that is normally distributed around the estimated depth $\mathbf{D}_{ij}$ with variance $\hat{\sigma}_i$: $\mathbb{D}_{ij} \sim \mathbb{N}(\mathbf{D}_{ij}, \hat{\sigma}_i)$. Drawing from the concept of an ideal termination distribution, we can minimize the KL divergence between the rendered ray distribution $h_{ij}(t)$ of $\mathbf{x}_i$'s image coordinates and the noisy depth distribution:

$$\mathbb{E}_{\mathbb{D}_{ij}} \mathrm{KL}\left[\delta\left(t-\mathbb{D}_{ij}\right) \| h_{ij}(t)\right] = \mathrm{KL}\left[\mathbb{N}\left(\mathbf{D}_{ij}, \hat{\sigma}_i\right) \| h_{ij}(t)\right] + const$$

- **Ray distribution loss:** The above equivalence allows the termination distribution $\mathbf{h(t)}$ to be trained with probabilistic depth supervision (which will be added and scaled to the overall training loss for NeRF):

$$\begin{aligned}\mathcal{L}_{\text{Depth}} &= \mathbb{E}_{x_i \in X_j} \int \log h(t) \exp\left(-\frac{\left(t-\mathbf{D}_{ij}\right)^2}{2\hat{\sigma}_i^2}\right) dt \\ &\approx \mathbb{E}_{x_i \in X_j} \sum_k \log h_k \exp\left(-\frac{\left(t_k-\mathbf{D}_{ij}\right)^2}{2\hat{\sigma}_i^2}\right) \Delta t_k.\end{aligned}$$

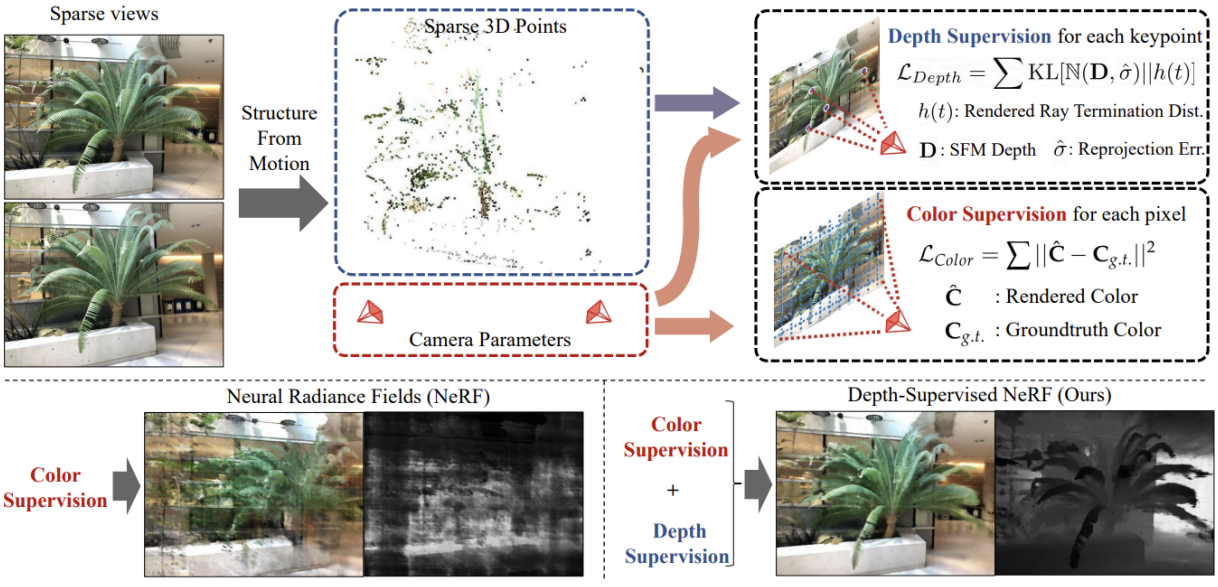

Figure 3.5: **Utilizing additional supervision from depth** recovered from 3D point clouds estimated from running structure-from-motion and impose a loss to ensure the rendered ray's termination distribution respects the surface priors given by the each keypoint.

Chapter 4

Experiments

4.1 My experiments with Method Combinations

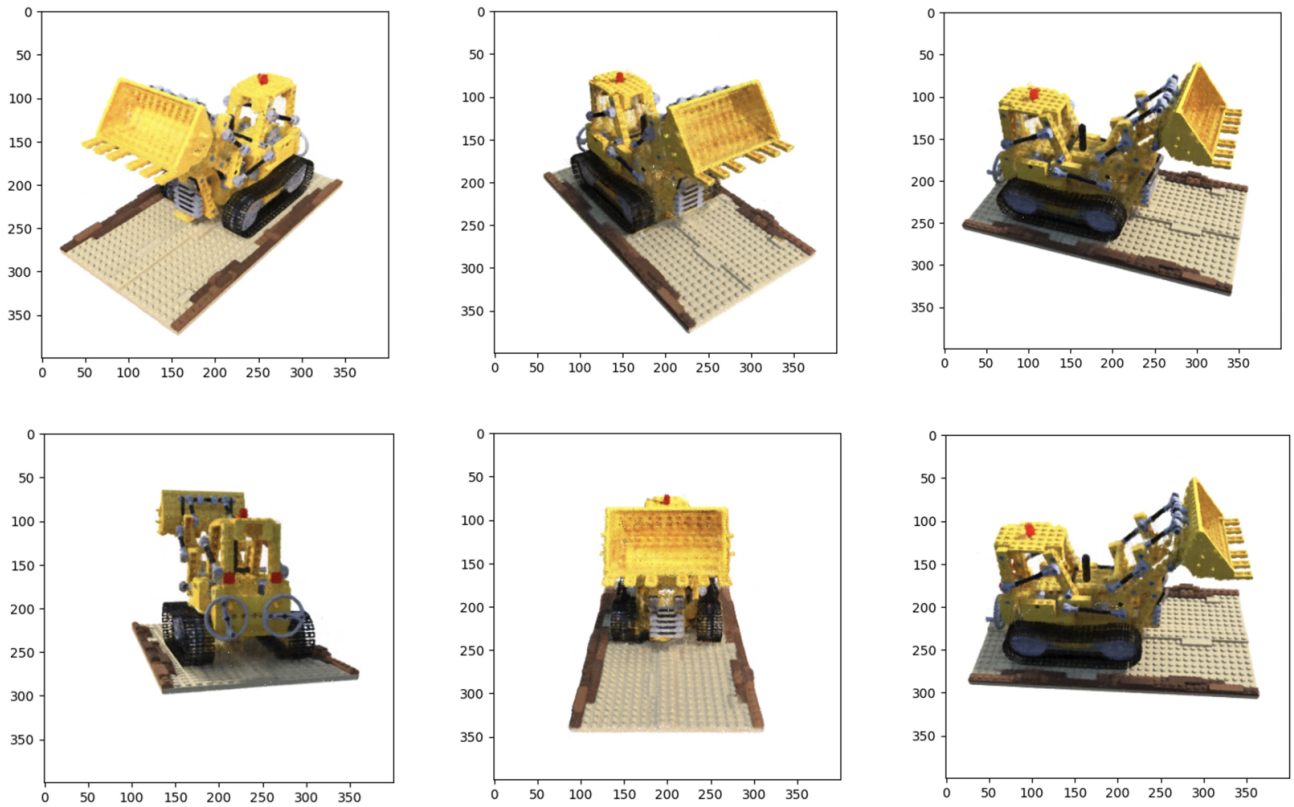

Figure 4.1

Figure 4.1 presents 6 out of 200 synthesized views rendered during testing by leveraging the combined methodologies of **ray entropy minimization** and **frequency regularization**. The training process was conducted over a duration of **3 hours** on an NVIDIA RTX **1080Ti** GPU, utilizing the lego sub dataset from `nerf_synthetic`. During training, **four** arbitrary input views were employed. Despite the limited number of images and significantly shorter training time compared to plain NeRF, the outcomes revealed substantially reduced artifacts across most views. The synthesized images from novel views exhibited clarity and maintained consistency in their semantic meanings without distortion. However, some newly generated synthesized views remained imperfect (refer to **Figure 4.2**), commonly exhibiting color inaccuracies (where the crane's color was confused by the red color of the tail button) or loss of details due to inaccurate radiance modeling for these pixels.

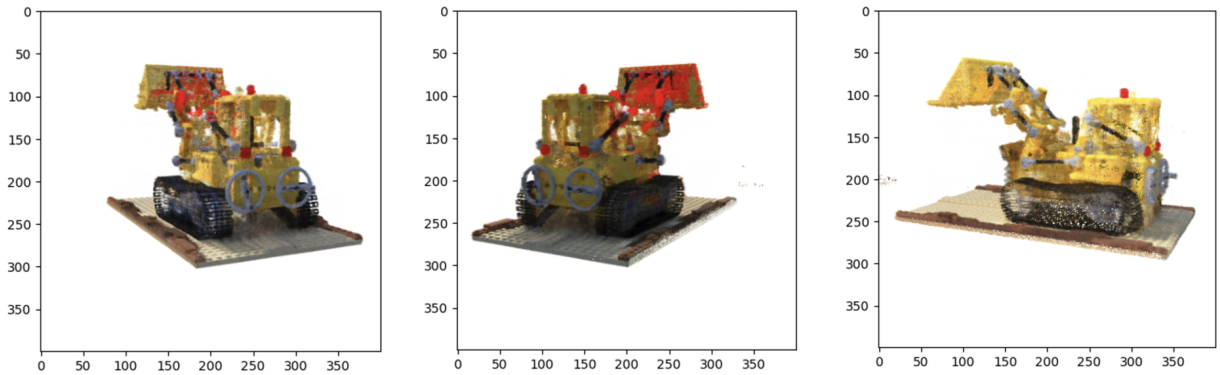

Figure 4.2

4.2 Experiments on depth-supervised method

To illustrate the effectiveness of KL divergence, the authors include a variant of DS-NeRF with an MSE loss between the SFM-estimated and the rendered depth. **Figure 4.3** qualitatively shows that KL divergence penalty produces views with less artifacts on NeRF Real sequences. DS-NeRF achieves a particular test PSNR threshold using **2-3x** less training iterations than NeRF. These benefits are significantly magnified when given fewer views. In the extreme case of only **two** views, NeRF is completely unable to match DS-NeRF's performance. On a single RTX **A5000**, a training loop of DS-NeRF takes $\approx$ 362.4 ms/iter while NeRF needs $\approx$ 359.8 ms/iter.

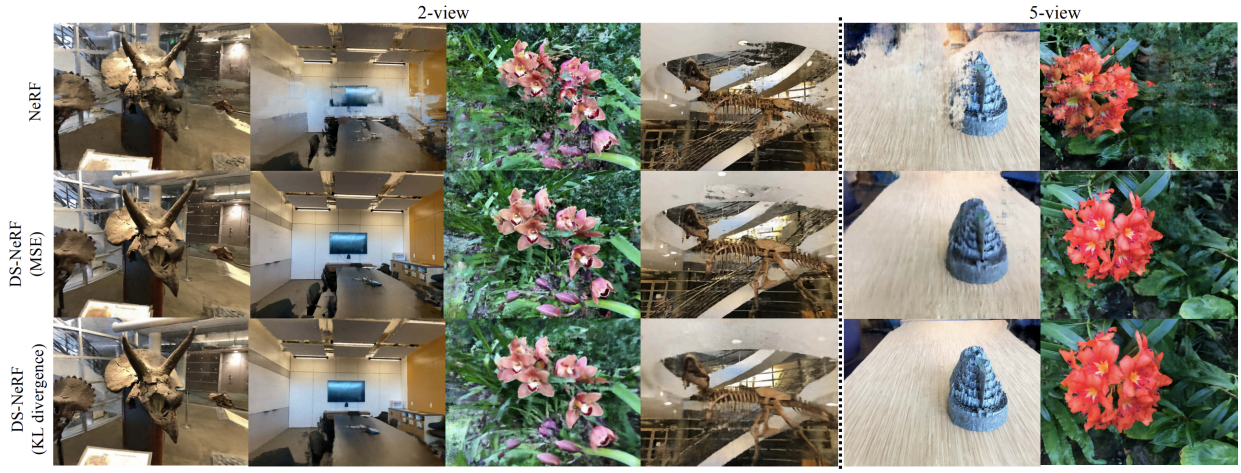

Figure 4.3: **Depth Supervision Ablations.** Nnovel views are rendered for NeRF and DS-NeRF trained on 2 views and 5 views. NeRF fails to render novel views as evident by the many artifacts. Using MSE between rendered and sparse depth improves results slightly, but with KL Divergence, DS-NeRF is able to render images with the fewest artifacts.

4.3 Comparisons

In **Figure 4.4**, it appears that PixelNeRF tend to overfit to the training data, leading to poor quantitative results, this conditional models seem to benefit from additional fine-tuning. For 3 input views, NeRF, MipNeRF, and DietNeRF perform worse than conditional models. InfoNeRF(applying ray entropy minimization), FreeNerf(applying frequency regularization), DS-NeRF and SPARF(depth-based method), nevertheless outperform the best conditional model, i.e. PixelNerf. SPARF and FreeNerf outperforms all others on all metrics in the sparsest scenario, i.e. when considering 3 input views.
All 3 approaches could in theory be applied to any base network, for example, MipNeRF. As a result, I believe combining theese approaches with any NeRF base architecture could lead to even better rendering quality in case of few-shot view synthesis.

Dataset	DTU			LLFF		
Method	LSNR	SSIM	LPIPS	LSNR	SSIM	LPIPS
PixelNeRF-ft	**19.36**	0.7	0.32	16.17	0.44	0.51
MipNeRF	7.64	0.23	0.66	14.62	0.35	0.5
NeRF	8.41	0.31	0.71	13.61	0.28	0.56
DietNeRF	10.01	0.35	0.57	14.94	0.37	0.5
InfoNeRF	11.23	0.44	0.54			
RegNeRF	15.33	0.62	0.34	19.08	0.59	0.34
DS-NeRF	16.52	0.54	0.48	18	0.55	0.27
SPARF	18.3	0.78	0.21	**20.2**	**0.63**	**0.24**
FreeNeRF	19.32	**0.79**	**0.2**	19.63	0.61	0.31

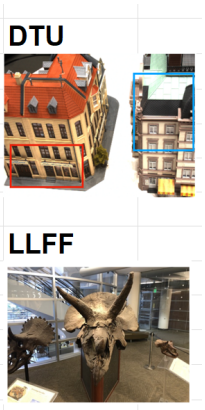

Figure 4.4: Evaluation on DTU and LLFF datasets.

Chapter 5

Comparisons with Classic Machine Learning approaches

There exist classic machine learning methods aimed at handling certain tasks within 3D reconstruction, such as processing input point clouds for 3D representation, rather than entirely solving the 3D reconstruction problem (without addressing the few-shot problem). Each method offers unique capabilities for handling and analyzing 3D data, catering to various aspects such as classification, clustering, dimensionality reduction, and probabilistic modeling, based on the specific requirements of the problem being addressed:

- **Random Forests:** An ensemble learning method built on decision trees. In 3D representation, they're effective for segmenting and categorizing 3D data points, distinguishing between different classes or categories within a 3D space.
- **Principal Component Analysis (PCA):** PCA is used for dimensionality reduction by transforming data into a lower-dimensional space. In 3D representation, PCA reduces the dimensionality of 3D data while preserving important information, enabling easier visualization or computation.
- **Gaussian Processes:** Gaussian Processes model the distribution of data points and make predictions based on the Gaussian distribution. They're employed in 3D representation for probabilistic modeling and prediction of features within a 3D space.
- **Support Vector Machines (SVMs):** SVMs create hyperplanes to classify data by maximizing the margin between different classes. SVMs are applied for 3D data classification, dividing data points in a 3D space based on their features into distinct classes.
- **K-Means Clustering:** In 3D representation, K-Means clusters similar 3D data points together, creating groups or clusters based on their spatial similarities.

In the realm of 3D reconstruction, particularly in few-shot synthesis scenarios, Deep Learning methods (NeRF-based methods) outperform classical Machine Learning techniques for several reasons:

- **Handling higher-dimensional complexities**: NeRF-based methods, especially deep neural networks, can learn and represent more intricate models with multiple hidden layers and features, enabling them to capture complex information within 3D data.
- **Feature learning:** Deep learning has the ability to autonomously learn features and represent data more naturally from the input, allowing it to extract crucial information from 3D point clouds or images.
- **Performance on large-scale data:** Deep Learning tends to perform better with large datasets. In few-shot 3D reconstruction, DL can efficiently learn from fewer images and leverage knowledge from previously seen large datasets.
- **Faster training process:** classic Machine Learning methods commonly address 3D reconstruction problems based on mesh structures or point cloud representations. In contrast, the approach of Neural Radiance Fields restricts the use of point cloud data, which leading to specific limitations and often resulting in slower training processes.
- **Flexibility and automatic feature selection:** DL allows models to autonomously discover necessary features from data, providing more flexibility and automatic adaptation to different types of data and tasks compared to classical ML.

Chapter 6

References

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. Commun. ACM, 65(1):99-106, 2022.

[2] Jiawei Yang, Marco Pavone, Yue Wang. FreeNeRF: Improving Few-shot Neural Rendering with Free Frequency Regularization, March 2023.

[3] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 12902-12911. IEEE, 2022.

[4] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2022.