S3IM: Stochastic Structural SIMilarity and Its Unreasonable Effectiveness for Neural Fields

Zeke Xie*, Xindi Yang*, Yujie Yang,
Qi Sun, Yixiang Jiang, Haoran Wang, Yunfeng Cai, and Mingming Sun

Baidu Research

Abstract

Recently, Neural Radiance Field (NeRF) has shown great success in rendering novel-view images of a given scene by learning an implicit representation with only posed RGB images. NeRF and relevant neural field methods (e.g., neural surface representation) typically optimize a point-wise loss and make point-wise predictions, where one data point corresponds to one pixel. Unfortunately, this line of research failed to use the collective supervision of distant pixels, although it is known that pixels in an image or scene can provide rich structural information. To the best of our knowledge, we are the first to design a nonlocal multiplex training paradigm for NeRF and relevant neural field methods via a novel Stochastic Structural SIMilarity (S3IM) loss that processes multiple data points as a whole set instead of process multiple inputs independently. Our extensive experiments demonstrate the unreasonable effectiveness of S3IM in improving NeRF and neural surface representation for nearly free. The improvements of quality metrics can be particularly significant for those relatively difficult tasks: e.g., the test MSE loss unexpectedly drops by more than 90% for TensoRF and DVGO over eight novel view synthesis tasks; a 198% F-score gain and a 64% Chamfer L_1 distance reduction for NeuS over eight surface reconstruction tasks. Moreover, S3IM is consistently robust even with sparse inputs, corrupted images, and dynamic scenes.

Figure 1. Qualitative comparison of standard training and *multiplex training* for neural radiance field on Replica Dataset [37] and T&T-Advanced Dataset [15]. Model: DVGO, TensoRF, and NeRF.

1. Introduction

Synthesizing novel-view images of a 3D scene from a group of images is a long-standing task in computer vision and computer graphics [4, 6, 17, 10, 33]. This long-standing task has recently made significant progress due to advances in learning-based neural rendering methods [27, 18, 20]. Learning-based neural field methods can represent 3D scenes and even the corresponding surfaces from posed images toward photorealistic novel-view synthesis.

Particularly, benefited from strong representations of deep neural networks (DNNs), Neural Radiance Field (NeRF) [20] has shown impressive success in synthesizing novel view synthesis of a given scene by implicitly encoding volumetric density and color through a fully connected neu-

*Equal Contributions. *{xiezeke,yangxindi}@baidu.com.*

ral network (often referred to as a multilayer perceptron or MLP). NeRF regresses from a single 5D representation (x, y, z, θ, ϕ)- 3D coordinates $\boldsymbol{x} = (x, y, z)$ plus 2D viewing directions $\boldsymbol{d} = (\theta, \phi)$- a single volume density σ and a view-dependent RGB color $\boldsymbol{c} = (r, g, b)$, computed by fitting the model to a group of pixels (from training images). NeRF approximates this continuous 5D scene representation with an MLP network $f_\Theta : (\boldsymbol{x}; \boldsymbol{d}) \rightarrow (\boldsymbol{c}; \sigma)$ and optimizes its weights Θ to map each input 5D coordinate to the corresponding volume density and directional emitted color.

Without loss of generality, we focus on the learning module of NeRF and write the loss optimized by NeRF as

$$L(\Theta) = \frac{1}{\|\mathcal{R}\|} \sum_{\boldsymbol{r} \in \mathcal{R}} l_{\mathrm{MSE}}(\Theta, \boldsymbol{r}), \tag{1}$$

where l_{MSE} is the point-wise Mean Squared Error (MSE) loss for each pixel/ray $\boldsymbol{r} = (\boldsymbol{x}_{\boldsymbol{r}}, \boldsymbol{d}_{\boldsymbol{r}}, \boldsymbol{c}_{\boldsymbol{r}})$ in the training data or data minibatch $\mathcal{R}$. Obviously, the MLP of NeRF learns and makes inference point-wisely, because one data point of the MLP corresponds to one pixel's information.

NeRF and neural surface representation are two of the most representative neural field methods [45], which are also called implicit neural representation methods [34] in computer vision. Neural surface representation [19, 23, 48, 39] is another important and long-standing problem orthogonal to NeRF. The point-wise MSE loss not only measures the accuracy of predictions in NeRF, but is also widely used in other relevant neural field methods. However, only optimizing point-wise loss can be a serious but overlooked pitfall of training NeRF and relevant methods.

In image quality assessment, the point-wise MSE-based metric Peak Signal-to-Noise Ratio (PSNR) [7] is widely used but do not correlate well with perceived image quality [26, 41], since point-wise metrics do not reflect structural information [16, 42, 43, 32] that is rich in the physical world.

The Structural Similarity (SSIM) index not only reflects the structure of a group of pixels, but also correlates with human visual systems significantly better than PSNR/MSE [42, 13]. SSIM is an important performance metric for evaluating NeRF models, but it is not used for training NeRF models. The conventional training paradigm of NeRF has unfortunately overlooked the structural information, as it only optimizes the point-wise MSE loss but not using the collective supervision of the structural information of multiple pixels. This is not surprising. The conventional point-wise paradigm is also common in other machine learning models.

Can we propose a novel training paradigm that can capture the structural information of a group of inputs in ways other than the MSE loss of individual pixels? Yes, we formulate a *multiplex loss* associated with the novel training paradigm as

$$L_{\mathrm{M}}(\Theta) = \frac{1}{\|\mathcal{R}\|} \sum_{\boldsymbol{r} \in \mathcal{R}} l_{\mathrm{MSE}}(\Theta, \boldsymbol{r}) + \lambda L_{\mathrm{S3IM}}(\Theta, \mathcal{R}), \tag{2}$$

where $L_{\mathrm{S3IM}}(\Theta, \mathcal{R})$ is computed over a group of nonlocal pixels and the hyperparameter λ adjusts the importance of S3IM. Unlike $L_{\mathrm{S3IM}}(\theta, \mathcal{R})$, the conventional loss $l_{\mathrm{MSE}}(\theta, (\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{c}, \sigma))$ is computed pixel-wisely. We emphasize that $L_{\mathrm{S3IM}}(\Theta, \mathcal{R})$ cannot be expressed as the sum of any other loss computed over individual pixels from $\mathcal{R}$. We call the proposed L_{S3IM} a multiplex loss because it allows a model to process multiple nonlocal inputs as a whole multiplex input. We will formally define S3IM in Section 3.

Main Contributions. We summarize main contributions as follows. **(1)** We propose the novel Stochastic Structural SIMilarity (S3IM) index, which measures the similarity between two groups of pixels and captures nonlocal structural similarity information from stochastically sampled pixels. To the best of our knowledge, we are the first to formulate a multiplex loss that can process multiple inputs collectively by capturing nonlocal information rather than processing multiple inputs independently by capturing individual-pixel information in neural fields. **(2)** S3IM is model-agnostic and can be generally applied to all types of neural field methods, such as NeRF and neural surface representation, with limited coding costs and computational costs. Our extensive experiments demonstrate the unreasonable effectiveness of the *nonlocal multiplex training* paradigm in improving NeRF and neural surface representation. Particularly, the improvement in quality metrics can be especially significant for those difficult tasks (e.g., Tables 1 and 6 and Figure 1) and is robust even with sparse inputs, corrupted images, and dynamic scenes.

2. Background

In this section, we introduce background knowledge.

2.1. NeRF

Recall that NeRF maps from a single 5D representation (x, y, z, θ, ϕ) to a single volume density σ and view-dependent RGB color $\boldsymbol{c} = (r, g, b)$ with an MLP network: $f_\Theta : (\boldsymbol{x}; \boldsymbol{d}) \rightarrow (\boldsymbol{c}; \sigma)$. For a target view with pose, a camera ray can be parameterized as $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ with the ray origin $\boldsymbol{o}$ and ray unit direction $\boldsymbol{d}$. The expected color $\boldsymbol{C}(\boldsymbol{r})$ of camera ray $\boldsymbol{r}(t)$ with near and far bounds t_n and t_f is

$$\hat{\boldsymbol{C}}(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\boldsymbol{c}(t)dt, \tag{3}$$

where $T = \exp(-\int_{t_n}^{t} \sigma(s)ds)$ denotes the accumulated transmittance along the ray from t_n to t. For simplicity, we have ignored the coarse and fine renderings via different sampling methods.

The rendered image pixel value for camera ray $\boldsymbol{r}$ can then be compared against the corresponding ground truth pixel value $\boldsymbol{C}(\boldsymbol{r})$, for all the camera rays. The conventional NeRF rendering loss is the MSE loss

$$L(\Theta) = \frac{1}{\|\mathcal{R}\|} \sum_{\boldsymbol{r}\in\mathcal{R}} \|\hat{\boldsymbol{C}}(\boldsymbol{r}) - \boldsymbol{C}(\boldsymbol{r})\|^2. \tag{4}$$

Obviously, NeRF is a point-wise machine learning model during both training and inference. The prediction and the loss are both computed pixel-wisely.

2.2. Quality Metrics: PSNR and SSIM

PSNR and SSIM are two most popular metrics of image quality assessment [42, 13]. For simplicity, we take grey-level (8 bits) images as examples. Given a test image I_a and a reference image I_b, both of size $W \times H$, the PSNR can be defined as

$$\mathrm{PSNR}(I_a, I_b) = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}(I_a, I_b)}\right), \tag{5}$$

where $\mathrm{MSE}(I_a, I_b) = \frac{1}{WH}\sum_{i,j,k}(I_{b,ijk} - I_{a,ijk})^2$. It is easy to see that PSNR directly depends on MSE and overlooks the collective information of a group of pixels.

In contrast, SSIM is a well-known quality metric that can capture local structural similarity between images or patches. SSIM is considered to be correlated with the quality perception of the human visual system well and is widely used for evaluating NeRF [42, 13]. Suppose $\boldsymbol{a} = \{a_i | i = 1, 2, 3, \ldots, n\}$ and $\boldsymbol{b} = \{b_i | i = 1, 2, 3, \ldots, n\}$ to be two discrete non-negative signals paired with each other (e.g., two image patches extracted from the same spatial location from paired images). SSIM is expressed by the combination of three terms which are the luminance, contrast, and structure comparison metrics:

$$\mathrm{SSIM}(a, b) = l(\boldsymbol{a}, \boldsymbol{b})c(\boldsymbol{a}, \boldsymbol{b})s(\boldsymbol{a}, \boldsymbol{b}). \tag{6}$$

The luminance $l(\boldsymbol{a}, \boldsymbol{b})$, contrast $c(\boldsymbol{a}, \boldsymbol{b})$, and structure comparison $s(\boldsymbol{a}, \boldsymbol{b})$ are, respectively, written as

$$l(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\mu_a\mu_b + C_1}{\mu_a^2 + \mu_b^2 + C_1}, \tag{7}$$

$$c(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\sigma_a\sigma_b + C_2}{\sigma_a^2 + \sigma_b^2 + C_2}, \tag{8}$$

$$s(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sigma_{ab} + C_3}{\sigma_a\sigma_b + C_3}. \tag{9}$$

where C_1, C_2, and C_3 are small constants given by $C_1 = (K_1 L)^2$, $C_2 = (K_2 L)^2$, and $C_3 = C_2/2$. Following the common setting [42, 20], we have $K_1 = 0.01$, $K_2 = 0.03$, and $L = 1$ for RGB. The range of SSIM lies in $[-1, 1]$.

In practice of image quality assessment, people usually apply the SSIM index locally rather than globally for robust statistics and efficient computation. The local statistics, including mean μ_a, variance σ_a, and covariance σ_{ab} are computed within a local $K \times K$ kernel window, which moves with a stride size s over the entire image. At each step, the local statistics and SSIM index are calculated within the local window. The final SSIM metric for evaluating NeRF is actually the mean SSIM (MSSIM) which is computed by averaging the SSIM indexes over each step. We leave the more details of SSIM in Appendix B.

3. Methodology: Multiplex Training via S3IM

In this section, we formulate a novel multiplex loss, S3IM, with a novel model-agnostic multiplex training paradigm for neural fields.

3.1. S3IM

Our motivation is to let the collective and nonlocal structural information contained in the group of data points supervise the learning of neural fields. Therefore, we must first define a multiplex-style loss over a group of pixels that can capture structural information, rather than the conventional point-wise loss (e.g., MSE) defined over individual pixels.

The pixels in a local patch may contain certain positional information. However, due to the stochastic training, the stochastically sampled pixels in a minibatch $\mathcal{R}$ cannot form a local patch and lose the positional relationship completely.

We formulate a stochastic variant of SSIM, namely S3IM, for stochastic training of NeRF. The idea is concise. Suppose we have B (e.g., 1024) pixels per minibatch and choose the kernel size and the stride size of S3IM as $K \times K$ (e.g., 4×4) and $s = K$. For simplicity and efficiency, we choose the stride size to be the same as the kernel size because the stochastic patches from one minibatch are independent and without overlapping pixels in this case.

We summarize S3IM as three steps:

1. We let B rays/pixels from a dataset/minibatch $\mathcal{R}$ randomly form a rendered patch $\mathcal{P}(\hat{\mathcal{C}})$ and the corresponding ground-truth image patch $\mathcal{P}(\mathcal{C})$, where $\hat{\mathcal{C}} = \{\hat{\boldsymbol{C}}(\boldsymbol{r}) | \boldsymbol{r} \in \mathcal{R}\}$ and $\mathcal{C} = \{\boldsymbol{C}(\boldsymbol{r}) | \boldsymbol{r} \in \mathcal{R}\}$.
2. We compute SSIM with the kernel size $K \times K$ and the stride size s over the rendered patch and the corresponding ground-truth patch, which is exactly an estimator of the proposed S3IM over the paired stochastic patches.
3. Due to stochasticity of $\mathcal{P}(\cdot)$, we may repeat steps **(1)** and **(2)** M times and average the M estimated SSIM values to obtain the final S3IM.

In summary, the final S3IM can be written as

$$\mathrm{S3IM}(\hat{\mathcal{R}}, \mathcal{R}) = \frac{1}{M}\sum_{m=1}^{M} \mathrm{SSIM}(\mathcal{P}^{(m)}(\hat{\mathcal{C}}), \mathcal{P}^{(m)}(\mathcal{C})), \tag{10}$$

where SSIM needs to apply the kernel size $K \times K$ and the stride size s.

Algorithm 1: Multiplex Training via S3IM

Let $\mathcal{A}$ be an SGD-like training algorithm;
while *no stopping criterion has been met* **do**
  Sample a data minibatch of rays $\mathcal{R}$ from $\mathcal{D}$;
  Obtain the ground-truth pixels $\mathcal{C} = \{\boldsymbol{C}(\boldsymbol{r})|\boldsymbol{r} \in \mathcal{R}\}$;
  Compute the rendered pixels $\hat{\mathcal{C}} = \{\hat{\boldsymbol{C}}(\boldsymbol{r})|\boldsymbol{r} \in \mathcal{R}\}$;
  for $m = 1$ **to** M **do**
    Initialize the stochastic patch generation function $\mathcal{P}^{(m)}$;
    Transform the rendered pixels into the rendered stochastic patch $\mathcal{P}^{(m)}(\hat{\mathcal{C}})$;
    Transform the ground-truth pixels into the ground-truth stochastic patch $\mathcal{P}^{(m)}(\mathcal{C})$;
    Compute $\mathrm{SSIM}(\mathcal{P}^{(m)}(\hat{\mathcal{C}}), \mathcal{P}^{(m)}(\mathcal{C}))$ with the given kernel $K \times K$ and the stride size K;
  end
  Obtain the S3IM loss $L_{\mathrm{S3IM}}(\Theta) = 1 - \frac{1}{M}\sum_{m=1}^{M} \mathrm{SSIM}(\mathcal{P}^{(m)}(\hat{\boldsymbol{C}}(\mathcal{R})), \mathcal{P}^{(m)}(\boldsymbol{C}(\mathcal{R})))$;
  Obtain the conventional MSE loss $L(\Theta) = \frac{1}{\|\mathcal{R}\|}\sum_{\boldsymbol{r}\in\mathcal{R}} \|\hat{\boldsymbol{C}}(\boldsymbol{r}) - \boldsymbol{C}(\boldsymbol{r})\|^2$;
  $L_M(\Theta) = L(\Theta) + \lambda L_{\mathrm{S3IM}}(\Theta)$;
  Compute the gradient $\nabla L_{\mathrm{M}}(\Theta)$;
  Update the model parameters Θ by $\mathcal{A}$;
end

We note that computing S3IM can be well vectorized and *multiplex training* also only requires back-propagation for once per iteration. Thus, the extra computational cost of *multiplex training* is limited. As S3IM lies in $[-1, 1]$ and positively correlated with image quality, we define the S3IM-based loss L_{S3IM} as

$$\begin{aligned} L_{\mathrm{S3IM}}(\Theta, \mathcal{R}) =& 1 - \mathrm{S3IM}(\hat{\mathcal{R}}, \mathcal{R}), \\ =& 1 - \frac{1}{M}\sum_{m=1}^{M} \mathrm{SSIM}(\mathcal{P}^{(m)}(\hat{\mathcal{C}}), \mathcal{P}^{(m)}(\mathcal{C})). \end{aligned} \tag{11}$$

We present the pseudocode in Algorithm 1; for generality, we focus on the shared machine learning module of various neural field methods and ignore the details of sampling in Equation (3). Fortunately, similar to SSIM, we may directly apply a default setting that $K = 4$ and $S = K$ without fine-tuning. The multiplex training paradigm via S3IM brings in two extra hyperparameters λ and M. If we let M approach $+\infty$, we will eliminate the stochasticity of S3IM and obtained its expected value. According to our experimental results in Section 5, we observe that $M = 1$ usually produce nearly same good results as $M = 10$, while we choose the default value $M = 10$ mainly for reducing the stochasticity of the final results. Thus, only the S3IM loss weight λ requires fine-tuning in practice.

While the original SSIM is also differentiable, directly optimizing it may not work well. Even if we use a pixel data loader that provides a minibatch of pixels per iteration, and an additional local-patch dataloader, which yields a minibatch of local patches per iteration, our ablation study shows that optimizing S3IM (via stochastic patches) significantly outperforms optimizing SSIM (via local patches). This is not surprising. SSIM over local patches can only capture structural information carried by nearby pixels from one image, while S3IM over stochastic patches can capture non-local structural information carried by distant pixels from different images. The additional data loader not only brings in extra coding and computational costs, but also hurts the performance of NeRF.

3.2. S3IM for Neural Surface Representations

In this subsection, we show that it is very easy to apply the proposed *multiplex training* to other neural field methods.

NeuS [39] is a recent powerful neural surface representation method. We choose neural surface representation as another benchmark task in this paper because neural surface representation is another line of research which has received much attention and made great progress recently in computer vision and computer graphics.

NeuS optimize a color loss with a regularizer loss and an optional mask loss, which can be written as

$$L_{\mathrm{NeuS}} = L_{\mathrm{color}} + \lambda_{\mathrm{reg}} L_{\mathrm{reg}}, \tag{12}$$

where the color loss use the L_1 loss form as

$$L_{\mathrm{color}} = \frac{1}{\|\mathcal{R}\|}\sum_{\boldsymbol{r}\in\mathcal{R}} \|\hat{\boldsymbol{C}}(\boldsymbol{r}) - \boldsymbol{C}(\boldsymbol{r})\|_1. \tag{13}$$

Similarly, when we train NeuS in the proposed *multiplex training* paradigm, we replace the original color loss by the multiplex color loss via S3IM as

$$L_{\mathrm{M,color}} = L_{\mathrm{color}} + \lambda L_{\mathrm{S3IM}}. \tag{14}$$

Our neural surface reconstruction experiments demonstrate that S3IM not only improves RGB image quality metrics, but also improves other geometric quality metrics (e.g., Chamfer Distance) even more significantly. This suggests that S3IM is generally useful for neural fields.

4. Related Work

In this section, we review representative related works and discuss their relations to our method.

Neural Fields. Fields can continuously parameterize an underlying physical quantity of an object or scene over

space and time. Since a long time ago, fields have been used to describe physical phenomena [30], compute image gradients [31], simulate collisions [25]. Recent advances showed increased interest in employing coordinate-based neural networks to parameterize some physical quantities over space and time, such as a neural network that maps a 3D spatial coordinate to a flow field in fluid dynamics, or a colour and density field in 3D scene representation. Such networks are often referred to as neural fields [45]. The application of neural fields in visual computing has make remarkable progress on various computer vision problems such as 3D scene reconstruction and generative modelling. In computer vision, people often refer to neural fields as implicit neural representations [34, 19, 23, 48, 39] (e.g., neural surface representation).

Neural Radiance Fields and Neural Surface Representations. Neural radiance field [20] and neural surface representation are two of the most representative neural field methods. This is why we let NeRF and NeuS serve as the base models for evaluating the proposed S3IM and the resulted *multiplex training* paradigm.

The line of NeRF has developed a number of useful NeRF variants. NeRF++ [54] helped resolve the shape-radiance ambiguity of NeRF. Mip-NeRF [1] adopted a multiscale representation method and significantly improved the quality of representating fine details. D-NeRF [28] extends neural radiance fields to modeling dynamical scences. Some works, such as Pixel-NeRF [51] and Reg-NeRF [22], focused on view synthesis from sparse inputs. NeRF−− [44] performs view synthesis by estimating approximate camera poses rather than known camera poses. As the common NeRF methods suffers a lot from slow training and inference, some works, including DVGO [38], TensoRF [3], and Instant NGP[21], aimed at accelerating training and inference of NeRF. Inspired by NeRF, NeuS [39] related the occupancy function of a volume to its volume density, thereby leading to improved rendering results and better geometry reconstruction. The proposed *multiplex training* paradigm via S3IM, orthogonal to the existing point-wise training paradigm, is model-agnostic and orthogonal to these NeRF variants.

5. Empirical Analysis and Discussion

In this section, we empirically demonstrate that *multiplex training* via S3IM significantly outperform the conventional training paradigm for NeRF and its variants.

We let the experimental settings follow original papers to produce the baselines, unless we specify otherwise. The main principle of our experimental setting is to fairly compare *multiplex training* via S3IM and standard training for NeRF and relevant neural field methods. Thus, we keep all hyperparameters same for standard training and *multiplex training* except S3IM. We mainly used five recent representative methods, including, vanilla NeRF [20], DVGO [38],

Table 1. Quantitative results of NeRF methods on Replica Dataset [37]. Model: DVGO.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Scene 1	Standard	13.26	0.506	0.719
	Multiplex	**32.63**	**0.929**	**0.0685**
Scene 2	Standard	14.82	0.653	0.637
	Multiplex	**35.03**	**0.957**	**0.0527**
Scene 3	Standard	15.24	0.644	0.636
	Multiplex	**29.88**	**0.957**	**0.0639**
Scene 4	Standard	17.73	0.691	0.505
	Multiplex	**39.32**	**0.976**	**0.0325**
Scene 5	Standard	16.52	0.659	0.505
	Multiplex	**35.70**	**0.969**	**0.0589**
Scene 6	Standard	20.10	0.843	0.309
	Multiplex	**29.27**	**0.947**	**0.0944**
Scene 7	Standard	23.20	0.845	0.248
	Multiplex	**31.53**	**0.952**	**0.0648**
Scene 8	Standard	15.67	0.729	0.521
	Multiplex	**34.66**	**0.956**	**0.0740**
Mean	Standard	17.07	0.696	0.510
	Multiplex	**33.50**	**0.955**	**0.0637**

Table 2. Quantitative results of NeRF methods on Replica Dataset [37]. Model: TensoRF.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Scene 1	Standard	12.13	0.468	0.7786
	Multiplex	**37.15**	**0.958**	**0.0335**
Scene 2	Standard	14.17	0.605	0.709
	Multiplex	**36.55**	**0.952**	**0.0563**
Scene 3	Standard	15.20	0.642	0.695
	Multiplex	**38.79**	**0.977**	**0.0287**
Scene 4	Standard	19.63	0.625	0.659
	Multiplex	**44.65**	**0.990**	**0.00965**
Scene 5	Standard	19.63	0.638	0.525
	Multiplex	**40.96**	**0.980**	**0.0273**
Scene 6	Standard	10.92	0.507	0.693
	Multiplex	**37.94**	**0.969**	**0.0473**
Scene 7	Standard	11.06	0.475	0.751
	Multiplex	**36.77**	**0.966**	**0.0437**
Scene 8	Standard	11.73	0.630	0.797
	Multiplex	**39.59**	**0.977**	**0.0307**
Mean	Standard	14.30	0.574	0.689
	Multiplex	**39.05**	**0.971**	**0.0454**

TensoRF [3], D-NeRF [28], and NeuS [39]. We present the experimental details in Appendix A.

5.1. Novel View Synthesis Experiments

Table 3. Quantitative results of NeRF methods on T&T. The mean PSNR, SSIM, LPIPS are computed over four scenes of T&T. Model: DVGO, TensoRF, and (vanilla) NeRF.

Model	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
DVGO	Standard	22.42	0.776	0.236
	Multiplex	**23.20**	**0.809**	**0.176**
TensoRF	Standard	19.69	0.650	0.365
	Multiplex	**22.85**	**0.777**	**0.230**
NeRF	Standard	21.02	0.659	0.364
	Multiplex	**22.64**	**0.725**	**0.304**

Static scene synthesis. We first study how *multiplex training* via S3IM improves NeRF on two benchmark datasets, Replica Dataset [37] and Tanks and Temples Advanced (T&T) [15]. Replica Dataset is a relatively difficult dataset which include large-scale scenes with less training images. T&T is a popular benchmark dataset for image-based 3D reconstruction with more training images. We use T&T Advanced as the defaulted T&T in the main text as it contain more complex details, while we leave the experimental results of T&T Intermediate in Appendix C.

We first choose the accelerated NeRF variants, DVGO, and TensoRF, as representatives of the NeRF family because training the accelerated NeRF methods is more environment-friendly and can significantly reduce the energy costs and carbon emissions of our work.

Our quantitative results on eight Replica scenes and four T&T scenes in Tables 1, 2, and 3 demonstrate that *multiplex training* via S3IM remarkably improves all three common image quality metrics for the NeRF faimily, including vanila NeRF, DVGO, and TensoRF. Particularly, the mean improvement in PSNR over eight Replica scenes can be incredibly up to **16.43** and **24.75** for DVGO and TensoRF, respectively, over eight novel view synthesis tasks. The PSNR gain suggests that test MSE can decrease by 2-3 orders of magnitude due to S3IM. As PSNR directly depends on MSE, it is surprising to see the improvement in PSNR given the fact that S3IM distract the training objective from the original MSE loss. It means that S3IM significantly improve the generalization of NeRF.

Why is S3IM so important on Replica Dataset? We conjecture that this is because Replica contains more complex details but fewer training images, while T&T Dataset contains less complex details but more training images. We will further verify this in the following experiments.

To the best of our knowledge, this is the most significant performance improvement along this line of research. We visualize the qualitative comparisons in Figure 1. We leave the results of each T&T scenes in Appendix C.

Few-shot Learning with sparse inputs. Learning with sparse or very few examples is a hot topic in machine learning [40] as well as neural rendering. In practice, we may not be able to collect many images of a scene. Does S3IM work well even if we have very few images? To answer this question, we train DVGO with the sparse version of a simple truck scene from T&T Intermediate, called Sparse Truck, where we randomly remove some training images.

We visualize the qualitative comparisons in Figure 2. The experimental results in Figure 3 further suggest that the performance improvement of *multiplex training* can be more significant when we have fewer training images. For example, when only 20% of the original training dataset is available, the improvement in PSNR can be surprisingly up to 4.32 for the simple Truck scene of T&T Intermediate. The more significant improvement with sparser inputs may explain why *multiplex training* makes more significant improvements on Replica Dataset than T&T Dataset.

Robustness to image corruption. While common benchmark datasets in NeRF studies are relatively clear, the training images in practice may be corrupted or noisy due to the realistic limitation of data collection. Gaussian image noise in digital images often arise during acquisition due to the inherent noise in the sensors [2]. Robustness to data corruption and noise memorization can be an important performance metric for neural networks [46] in weakly-supervised learning but unfortunately overlooked by most existing NeRF studies.

We again use the Truck scene to make an image-corrupted dataset, called Corrupted Truck, where we inject Gaussian noise with the standard deviation as std into original Truck images (each RGB value lies in $[0, 1]$) and obtain the corrupted version. We visualize the qualitative comparisons in Figure 2. The experimental results in Figure 4 show that *multiplex training* via S3IM can significantly improve the robustness to image corruption.

Table 4. Quantitative results of NeRF methods on dynamic scenes, Mutant and LEGO [28]. Model: D-NeRF.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Lego	Standard	21.29	0.821	0.0634
	Multiplex	**23.36**	**0.900**	**0.0482**
Mutant	Standard	32.14	0.972	0.0181
	Multiplex	**32.76**	**0.976**	**0.0151**

Dynamic scene synthesis. Rendering novel photo-realistic views of dynamic scenes is a more difficult task

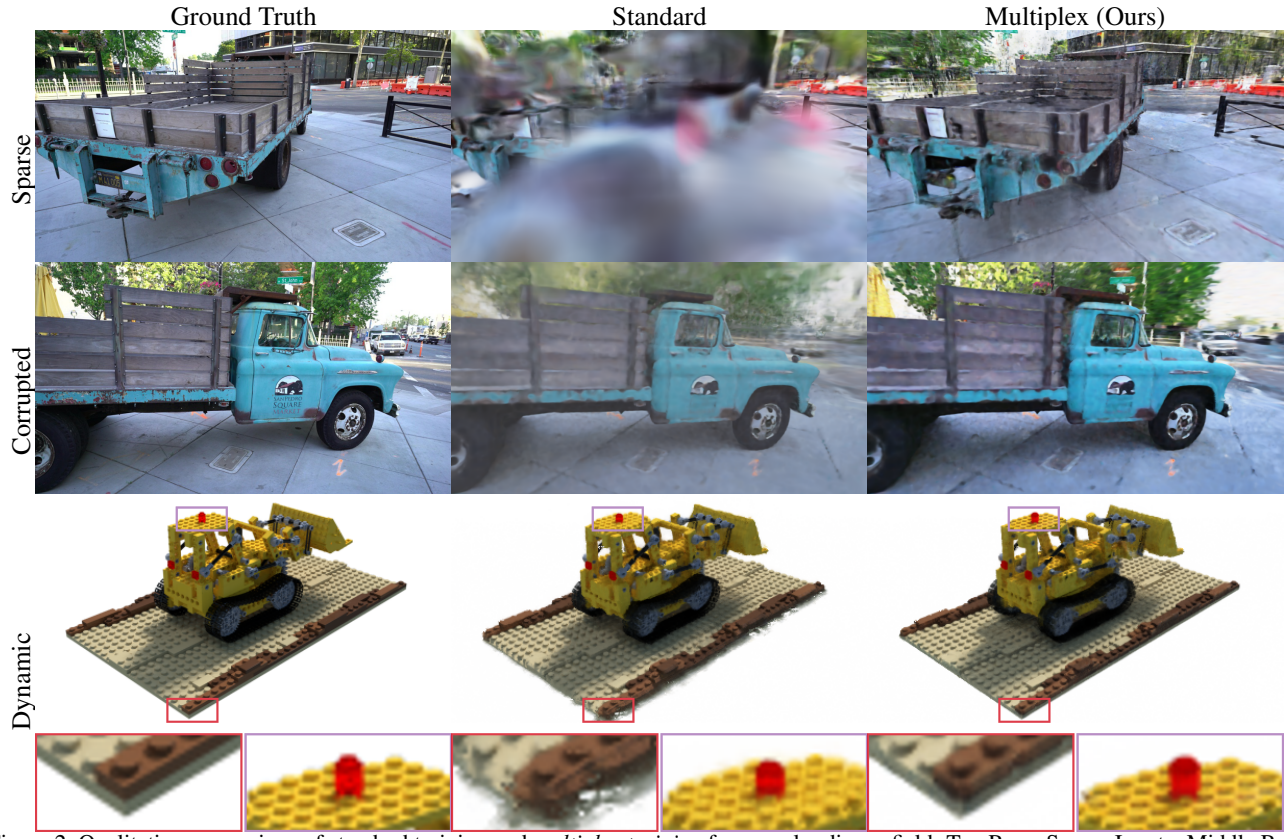

Figure 2. Qualitative comparison of standard training and *multiplex training* for neural radiance field. Top Row: Sparse Inputs. Middle Row: Corrupted Images. Bottom Row: Dynamic Scene.

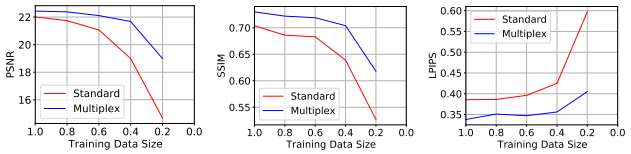

Figure 3. We plot the curves of PSNR, SSIM, and LPIPS with respect to the training data size, namely the portion of training samples kept from the original training dataset. The improvement of *multiplex training* can be even more significant when the training data size decreases. Model: DVGO. Dataset: T&T-Truck.

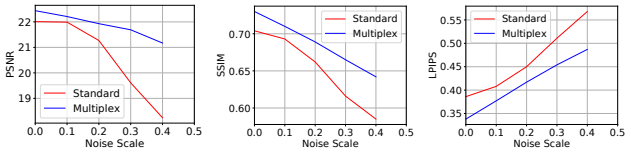

Figure 4. We plot the curves of PSNR, SSIM, and LPIPS with respect to the image noise scale std. The improvement of *multiplex training* can be even more significant when training image is corrupted by random noise. Model: DVGO. Dataset: T&T-Truck.

than neural rendering of static scenes. A NeRF variant, D-NeRF, has been specifically designed to render dynamic scenes. We also study how *multiplex training* via S3IM improves D-NeRF on two representative dynamic scenes, Lego and Mutant [28]. We visualize the qualitative comparisons in Figure 2. The results of Table 4 and Figure 2 both suggest that *multiplex training* via S3IM also significantly improve the performance of D-NeRF for dynamic scenes.

Regularization methods for monocular video. Moreover, we also show that S3IM can further improve other regularized methods specifically designed for monocular video, such as Dynamic NeRF [8]. Monocular video only contains sparse and continuous training views. We reproduce the original Dynamic NeRF's quantitative result on the monocular video (Balloon1) and compare it with S3IM-enhanced results in Table 5.

Table 5. Quantitative results of Dynamic NeRF methods on monocular video [8]. Model: Dynamic NeRF.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Balloon1	Standard	19.80	0.693	0.210
	Multiplex	**22.55**	**0.767**	**0.108**

Table 6. Quantitative results of neural surface reconstruction on Replica Dataset [37]. Model: NeuS.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)	Chamfer-L_1(↓)	Accuracy(↓)	Completeness(↓)	Precision(↑)	Recall(↑)	F-score(↑)	Normal C.(↑)
Scene 1	Standard	28.52	0.843	0.148	18.13	11.60	24.65	29.84	17.79	22.29	72.14
	Multiplex	**30.93**	**0.876**	**0.100**	**6.264**	**6.580**	**5.948**	**58.31**	**81.00**	**59.99**	**80.58**
Scene 2	Standard	29.15	0.834	0.179	19.00	12.73	25.27	28.32	16.29	20.69	**75.16**
	Multiplex	**31.06**	**0.870**	**0.135**	**7.475**	**8.506**	**6.425**	**53.48**	**61.25**	**57.10**	72.84
Scene 3	Standard	28.44	0.877	0.150	40.34	33.80	46.89	9.47	5.68	7.10	67.69
	Multiplex	**33.97**	**0.933**	**0.0372**	**6.188**	**6.462**	**5.914**	**57.66**	**59.95**	**58.72**	**75.96**
Scene 4	Standard	31.84	0.874	0.152	18.90	13.61	24.19	22.25	12.52	16.02	71.89
	Multiplex	**37.28**	**0.940**	**0.0596**	**7.397**	**8.223**	**6.570**	**54.60**	**70.68**	**61.60**	**72.97**
Scene 5	Standard	33.78	0.897	0.121	76.33	11.83	140.83	33.09	0.60	1.18	50.05
	Multiplex	**36.32**	**0.924**	**0.071**	**34.40**	**31.43**	**37.38**	**14.89**	**9.56**	**11.64**	**55.60**
Scene 6	Standard	27.82	0.882	0.141	15.63	11.71	19.54	30.74	21.53	25.32	69.33
	Multiplex	**31.18**	**0.914**	**0.0995**	**6.980**	**7.211**	**6.748**	**60.08**	**59.88**	**59.98**	**73.62**
Scene 7	Standard	28.80	0.898	0.114	11.45	8.88	14.02	43.23	33.39	37.68	**78.53**
	Multiplex	**30.99**	**0.917**	**0.0837**	**7.824**	**7.845**	**7.803**	**51.38**	**51.23**	**51.31**	78.30
Scene 8	Standard	28.29	0.908	0.130	30.82	25.71	35.92	15.34	8.73	11.13	68.86
	Multiplex	**34.89**	**0.954**	**0.0539**	**6.136**	**5.839**	**6.434**	**61.62**	**62.56**	**62.08**	**79.68**
Mean (8 scenes)	Standard	29.58	0.877	0.142	28.83	16.23	41.41	26.54	14.57	17.68	69.21
	Multiplex	**33.33**	**0.916**	**0.0799**	**10.33**	**10.26**	**10.40**	**51.50**	**57.01**	**52.80**	**73.69**

5.2. Surface Reconstruction Experiments

Table 7. Quantitative results of NeuS on T&T.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Scene 1	Standard	20.73	0.636	0.393
	Multiplex	**22.01**	**0.681**	**0.311**
Scene 2	Standard	21.26	0.739	0.434
	Multiplex	**23.23**	**0.812**	**0.270**
Scene 3	Standard	17.58	0.551	0.428
	Multiplex	**19.07**	**0.609**	**0.344**
Scene 4	Standard	20.32	0.554	0.398
	Multiplex	**22.90**	**0.660**	**0.301**
Mean	Standard	19.97	0.620	0.413
	Multiplex	**21.55**	**0.691**	**0.307**

Reconstructing surfaces from images is also a fundamental problem in computer vision. Recent neural surface reconstruction methods [49, 48, 24, 39] belong to another kind of neural field method orthogonal to NeRF. To evaluate the universal effectiveness of the proposed method, we empirically study S3IM for a classical neural surface reconstruction method, called NeuS, which can render both RGB images and surface information. NeuS employs the same training data as NeRF without ground-truth surface information.

We use Replica Dataset and T&T Advanced as two benchmark datasets for surface reconstruction. Replica Dataset has the ground-truth surface information in test data, Advanced Scenes of T&T Dataset has no available ground-truth surface information. We add a group quality metrics which can measure rendering quality of surface information and geometric information when ground-truth surface and geometric information is available. These surface quality metrics, especially such as Chamfer L_1 Distance and F-score, are widely used for evaluating surface reconstruction.

Our qualitative results in Figure 5 show that S3IM improve both RGB rendering and depth rendering. Our quantitative results in Tables 6 and 7 suggest that *multiplex training* via S3IM can significantly improve neural surface reconstruction methods in terms of all three image quality metrics and all seven surface quality metrics. For example, in terms if two very popular surface quality metrics, we obtain **a 64% Chamfer $\mathbf{L_1}$ distance reduction and a 198% F-score gain over eight surface reconstruction tasks**. Most surface quality metrics have been improved by more than 10 points. This again demonstrate the unreasonable effectiveness and universality of S3IM in neural field methods.

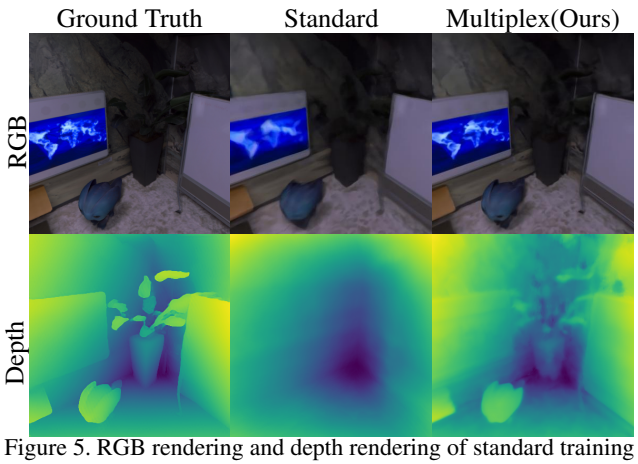

Figure 5. RGB rendering and depth rendering of standard training and *multiplex training* via S3IM for neural surface representation. Model: NeuS. Dataset: Replica Scene 1 (Room 0).

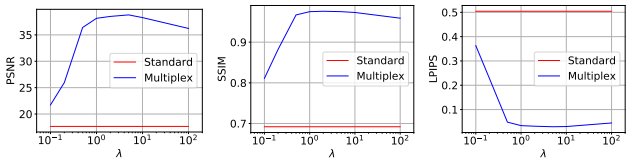

Figure 6. We plot the curves of PSNR, SSIM, and LPIPS with respect to the hyperparameter λ. S3IM is robust to a wide range of λ. Model: DVGO. Dataset: Replica Scene 4.

5.3. Discussion

S3IM hyperparameters $\{\lambda, M, K\}$. We plot the curves of PSNR, SSIM, and LPIPS with respect to the hyperparameter λ in Figure 6. The improvement of S3IM is robust to a wide range of λ choices. We also study how the improvement depends on the hyperparameter M and report that $M = 1$ can also make significant improvements, while the default value is $M = 10$, shown in Table 8.

Moreover, we also empirically verify that choosing a relatively small kernel size for S3IM provide robust statistics as we discuss. We empirically evaluated S3IM with the kernel size K and stride size S as 4, 16, and 64, respectively. Note that the computational complexity is approximately invariant to the kernel size, when we choose $K = S$. The setting $K = S$ can provide fair comparisons. In Table 9, we clearly observe that increasing the size K can significantly degrade the performance.

Table 8. Quantitative results and the training time (A100 GPU hours) with respect to the hyperparameter M. Dataset: Replica

Scene	Model	M	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)	Training Time
Room 0	TensoRF	0	Standard	12.03	0.464	0.773	0.369
		1	Multiplex	36.65	0.954	0.0387	0.374
		10	Multiplex	37.15	0.958	0.0335	0.432
Office 0	NeuS	0	Standard	31.84	0.874	0.152	2.95
		1	Multiplex	37.02	0.937	0.0666	2.95
		10	Multiplex	37.28	0.940	0.0596	2.98

Table 9. Quantitative results of S3IM with various kernel sizes. Model: TensoRF. Dataset: Replica Scene 1 (Room 0).

Kernel Size	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
-	Standard	12.13	0.468	0.7786
4	Multiplex	**37.15**	**0.958**	**0.0335**
16	Multiplex	36.79	0.954	0.0375
64	Multiplex	12.10	0.4989	0.8124

Computational costs. We also present the training time (GPU hours) of standard training and multiplex training with respect to M in Table 8. The results show that the extra computational cost of S3IM is very limited compared with the significant quality improvement. For example, the extra computational costs are only 8% for TensoRF and 1% for NeuS with $M = 10$; only 1% for TensoRF and nearly free for NeuS with $M = 1$.

Table 10. Ablation study of nonlocal S3IM and local SSIM.

Scene	Model	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Truck	DVGO	MSE	22.01	0.704	0.386
		MSE + SSIM	15.90	0.569	0.566
		MSE + S3IM (Ours)	**22.44**	**0.730**	**0.338**

S3IM verus SSIM. We conduct ablation study on the proposed S3IM over stochastic patches and the conventional image quality metric SSIM over local patches. The patch sizes of S3IM and SSIM are both 64×64, while the kernel/stride sizes are both 4.. We present the ablation study of stochastic-patch S3IM and local-patch SSIM in Table 10. The results suggest that S3IM with the non-local and stochastic structural information is very helpful, while SSIM with the local structural information is marginal and sometimes even harmful due to lack of stochasticity. The ablation study again verifies the novel contribution of our method.

Future Work There are at least three promising future research directions. First, we may directly introduce S3IM into non-RGB losses, such as depth losses. Second, we can develop better multiplex losses than S3IM for other machine learning tasks, including Graph Neural Networks and Physics-Informed Neural Network [29], as long as the point-wise losses are optimized in these tasks. Third, it will be very valuable to theoretically understand minima's flatness [53, 47] and generalization learned by S3IM.

6. Conclusion

Recently neural fields have achieved great empirical success and are receiving considerable attention in computer vision and computer graphics. However, the current training paradigm usually uses only point-wise supervision information and overlooks the rich structural information contained in the group of pixels. In this work, we proposed S3IM to extend the performance limit of neural fields by exploiting the nonlocal structural information of groups of pixels. We have demonstrated the unreasonable and robust effectiveness of S3IM for all employed models and scenes in terms of 10 quality metrics, while the extra costs are nearly free. Our extensive experiments strongly support important values of S3IM and nonlocal information. We believe that S3IM will serve as a default method for training neural fields in future.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[2] Ajay Kumar Boyat and Brijendra Kumar Joshi. A review paper: Noise models in digital image processing. *Signal & Image Processing*, 6(2):63, 2015.

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022.

[4] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, 1993.

[5] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[6] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996.

[7] Ahmet M Eskicioglu and Paul S Fisher. Image quality measures and their performance. *IEEE Transactions on communications*, 43(12):2959–2965, 1995.

[8] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[10] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[13] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[15] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

[16] Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984.

[17] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[18] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[19] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019.

[20] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[22] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022.

[23] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[24] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021.

[25] Stanley Osher, Ronald Fedkiw, and K Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004.

[26] Thrasyvoulos N Pappas, Robert J Safranek, and Junqing Chen. Perceptual criteria for image quality evaluation. *Handbook of image and video processing*, 110, 2000.

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[28] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[29] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[30] Paolo Sabella. A rendering algorithm for visualizing 3d scalar fields. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 51–58, 1988.

[31] Karsten Schlüns and Reinhard Klette. *Local and global integration of discrete vector fields*. Springer, 1997.

[32] Hamid R Sheikh and Alan C Bovik. Image information and visual quality. *IEEE Transactions on image processing*, 15(2):430–444, 2006.

[33] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, volume 4067, pages 2–13. SPIE, 2000.

[34] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[35] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[36] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[37] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[38] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[39] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34:27171–27183, 2021.

[40] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[41] Zhou Wang and Alan C Bovik. A universal image quality index. *IEEE signal processing letters*, 9(3):81–84, 2002.

[42] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[43] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[44] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[45] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[46] Zeke Xie, Fengxiang He, Shaopeng Fu, Issei Sato, Dacheng Tao, and Masashi Sugiyama. Artificial neural variability for deep learning: On overfitting, noise memorization, and catastrophic forgetting. *Neural Computation*, 33(8), 2021.

[47] Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021.

[48] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

[49] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020.

[50] Lin Yen-Chen. Nerf-pytorch. `https://github.com/yenchenlin/nerf-pytorch/`, 2020.

[51] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[52] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022.

[53] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Machine Learning*, 2017.

[54] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[55] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A. Experimental Settings and Details

In this section, we present the experimental settings and details for reproducing the results. The main principle of our experimental setting is to fairly compare *multiplex training* and standard training for NeRF and the variants. Our experimental settings follows original papers to produce the baselines, unless we specify otherwise.

S3IM Setting. We always choose the kernel size $K = 4$, the stride size $S = 4$, and the number of stochastic patches $M = 10$ without fine-tuning. We fine-tune the S3IM loss weight λ from $\{0.05, 0.1, 0.2, 0.5, 1, 2, 5\}$ for the NeRF family and $\{1, 2, 5, 10, 20, 50, 100\}$ for the NeuS family.

A.1. Models and Optimization

DVGO Setting We employ the sourcecode of DVGO (Version 2) in the original project [38] without modifying training hyperparameters. So we train DVGO via Adam [14] with the batch size $B = 8192$. The learning rate of density voxel grids and color/feature voxel grids is 0.1, and the learning rate of the RGB net (MLP) is 0.001. The total number of iterations is 5000. We multiply the learning rate by 0.1 per 1000 iterations.

TensoRF Setting We employ the sourcecode of the original project [3] without modifying training hyperparameters. The total number of iterations is 30000. The batch size is 4096. The initial number of voxels is 128^3, while the final number of voxels is 300^3. The upsamping iterations for voxels are 2000,3000,4000,5500 and 7000, respectively. Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ is used.

NeRF Setting We employ a popular open-source implementation [50] of the original NeRF. Again, we follow its defaulted training setting. The learning rate is 0.0005, and the learning rate scheduler is $0.1^{iters/500000}$.

D-NeRF Setting We directly employ the sourcecode of D-NeRF in the original project [28]. We only slightly change the original batch size $B = 500$ to $B = 512$ for generating the squared stochastic patch. The learning rate is 0.0005. The total number of iterations is 800k. The learning rate decay follows the original paper.

NeuS Setting We employ the NeuS implementation of SDFStudio [52] and follow its default hyperparameters. The difference of the hyperparameters between SDFStudio and the original paper [39] is that SDFStudio trains 100k iterations, while the original paper trains 300k iterations.

A.2. Datasets

Replica Dataset Replica Dataset has no splitted training dataset and test dataset. In the experiments on Replica, if one image index is divisible by 10, we move the image to the test dataset; if not, we move the image to the training dataset. Thus, we have 90% images for training and 10% images for evaluation.

T&T Dataset Advanced T&T Dataset Advanced has no splitted training data and test data. We follow the original splitted way in the standard setting. In the experiments on Replica, if one image index is divisible by 10, we move the image to the test T&T Dataset Advanced; if not, we move the image to the training dataset. Similarly, we again have 90% images for training and 10% images for evaluation.

T&T Dataset Intermediate T&T Dataset has splitted training data and test data. We follow the original splitted way in the standard setting. In the experiments of sparse inputs, we randomly remove the training images. In the experiments of corrupted images, we inject Gaussian noise with the scale std into RGB values of the training images, and clip the corrupted RGB values into $[0, 1]$.

Dynamic Scenes: LEGO and Mutant The two dynamics scenes are used in the original D-NeRF paper. We use them in the same way without any modification.

B. Image Quality Metrics

As we mentioned above, PSNR and SSIM are two most popular image quality metrics. Beyond them, we have also seen other useful metrics. Multiscale SSIM [43] is a variant of SSIM, which incorporates image details at different resolutions. However, Multiscale SSIM still can only capture local structural information carried by nearby pixels.

Deep features learned by DNNs has unreasonable effectiveness as a perceptual metric for measuring the similarity between two sets of perceptual features [55]. Thus, the Learned Perceptual Image Patch Similarity (LPIPS) metric [55] serves as the third rendering quality metric in NeRF and related studies, because it agrees surprisingly well with humans. The Fréchet inception distance (FID) score [11] is another metric which measure the distance-based similarity of perceptive features, but it is more widely used for evaluating generative models, such as Generative Adversarial Networks [5, 9] (GAN) and Diffusion Models [35, 36, 12]. Both LPIPS and FID metrics inevitably have certain stochasticity, because deep features are learned from stochastic training of DNNs. However, the stochasticity does not affect the usefulness and popularity of LPIPS and FID as image quality metrics.

S3IM can also be considered as a image quality metric, while we only use S3IM as a differentiable training objective in this paper. More specifically, S3IM measures the structural similarity of two paired sets of pixels(/signals), which may or may not form images. By analyzing the stochastic patch consists of random pixels, S3IM can capture non-local structural information carried by nearby/distant pixels. S3IM also inevitably have certain stochasticity like LPIPS and FID, while S3IM does not depend on deep features given by DNNs.

SSIM is a well-known quality metric that can capture local structural similarity between images or patches. SSIM is

considered to be correlated with the quality perception of the human visual system well and is widely used for evaluating NeRF [42, 13]. Suppose $\boldsymbol{a} = \{a_i | i = 1, 2, 3, \ldots, n\}$ and $\boldsymbol{b} = \{b_i | i = 1, 2, 3, \ldots, n\}$ to be two discrete non-negative signals paired with each other (e.g. two image patches extracted from the same spatial location from paired images). We denote the mean intensity of a signal as μ (e.g. $\mu_a = \frac{1}{n}\sum_{i=1}^{n} a_i$), the standard deviation of a signal as σ^2 (e.g. $\sigma_a^2 = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \mu_a)^2$), and the covariance between two signals as σ_{ab}^2 (e.g. $\sigma_{ab}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \mu_a)(b_i - \mu_b)$).

SSIM is expressed by the combination of three terms which are the luminance, contrast, and structure comparison metrics:

$$\mathrm{SSIM}(a, b) = l(\boldsymbol{a}, \boldsymbol{b}) c(\boldsymbol{a}, \boldsymbol{b}) s(\boldsymbol{a}, \boldsymbol{b}). \tag{15}$$

The luminance $l(\boldsymbol{a}, \boldsymbol{b})$, contrast $c(\boldsymbol{a}, \boldsymbol{b})$, and structure comparison $s(\boldsymbol{a}, \boldsymbol{b})$ are, respectively, written as

$$l(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\mu_a\mu_b + C_1}{\mu_a^2 + \mu_b^2 + C_1}, \tag{16}$$

$$c(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\sigma_a\sigma_b + C_2}{\sigma_a^2 + \sigma_b^2 + C_2}, \tag{17}$$

$$s(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sigma_{ab} + C_3}{\sigma_a\sigma_b + C_3}, \tag{18}$$

where C_1, C_2, and C_3 are small constants given by

$$C_1 = (K_1 L)^2, C_2 = (K_2 L)^2, \text{and } C_3 = C_2/2. \tag{19}$$

Following the common setting [42, 20], we set $K_1 = 0.01$ and $K_2 = 0.03$ in this paper. The data range L is 1 for pixel RGB values. The range of SSIM lies in $[-1, 1]$.

In practice of image quality assessment, people usually apply the SSIM index locally rather than globally. The local statistics μ_a, σ_a, and σ_{ab} are computed within a local $K \times K$ kernel window, which moves with a stride size s over the entire image. For example, for evaluating NeRF, people often use the kernel size 11×11, the stride size 1, and the circular symmetric Gaussian weighting function $\boldsymbol{w} = \{w_i | i = 1, 2, \ldots, n\}$, with standard deviation of 1.5 samples, normalized to unit sum ($\sum_i w_i = 1$). The local statistics are then written as

$$\mu_a = \sum_{i=1}^{n} w_i a_i, \tag{20}$$

$$\sigma_a = \left(\sum_{i=1}^{n} w_i (a_i - \mu_a)^2\right)^{\frac{1}{2}}, \tag{21}$$

$$\sigma_{ab} = \left(\sum_{i=1}^{n} w_i (a_i - \mu_a)(b_i - \mu_b)\right)^{\frac{1}{2}}. \tag{22}$$

At each step, the local statistics and SSIM index are calculated within the local window. The final SSIM metric for evaluating NeRF is actually the mean SSIM (MSSIM) which is computed by averaging the SSIM indexes over each step.

Table 11. Quantitative results of neural rendering of scenes in T&T Intermediate. Model: DVGO.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
M60	Standard	17.49	0.647	0.501
	Multiplex	**17.71**	**0.652**	**0.483**
Playground	Standard	22.74	0.669	0.467
	Multiplex	**22.75**	**0.681**	**0.444**
Train	Standard	17.19	0.566	0.533
	Multiplex	**18.24**	**0.581**	**0.491**
Truck	Standard	22.01	0.704	0.386
	Multiplex	**22.44**	**0.730**	**0.338**

Table 12. Quantitative results of NeRF methods on T&T. Model: DVGO.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Scene 1	Standard	21.80	0.759	0.243
	Multiplex	**22.16**	**0.783**	**0.191**
Scene 2	Standard	23.96	0.847	0.250
	Multiplex	**25.37**	**0.872**	**0.171**
Scene 3	Standard	18.64	0.697	0.272
	Multiplex	**19.71**	**0.757**	**0.192**
Scene 4	Standard	25.27	0.800	0.179
	Multiplex	**25.55**	**0.823**	**0.151**
Mean	Standard	22.42	0.776	0.236
	Multiplex	**23.20**	**0.809**	**0.176**

Table 13. Quantitative results of NeRF methods on T&T. Model: NeRF.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Scene 1	Standard	20.11	0.647	0.351
	Multiplex	**22.34**	**0.714**	**0.302**
Scene 2	Standard	23.09	0.774	0.349
	Multiplex	**25.20**	**0.848**	**0.247**
Scene 3	Standard	17.22	0.523	0.486
	Multiplex	**18.13**	**0.571**	**0.467**
Scene 4	Standard	23.65	0.692	0.269
	Multiplex	**24.88**	**0.767**	**0.198**
Mean	Standard	21.02	0.659	0.364
	Multiplex	**22.64**	**0.725**	**0.304**

Table 14. Quantitative results of NeRF methods on T&T. Model: TensoRF.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Scene 1	Standard	17.33	0.643	0.413
	Multiplex	**22.67**	**0.779**	**0.216**
Scene 2	Standard	24.44	0.847	0.240
	Multiplex	**25.14**	**0.859**	**0.214**
Scene 3	Standard	12.15	0.342	0.761
	Multiplex	**18.64**	**0.699**	**0.286**
Scene 4	Standard	24.84	0.766	0.207
	Multiplex	**24.93**	**0.770**	**0.205**
Mean	Standard	19.69	0.650	0.365
	Multiplex	**22.85**	**0.777**	**0.230**

Table 15. Quantitative results of few-shot learning. We only keep eight training images from the Truck Scene, T&T Intermediate. Model: DVGO.

Scene	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Truck	Standard	11.37	0.343	0.704
	Multiplex	**13.43**	**0.372**	**0.610**

Table 16. Quantitative results of neural rendering from sparse training images. Size indicates the portion of training samples kept from the original training dataset.

Size	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
20%	Standard	14.67	0.527	0.597
	Multiplex	**18.99**	**0.618**	**0.405**
40%	Standard	18.99	0.639	0.425
	Multiplex	**21.68**	**0.704**	**0.356**
60%	Standard	21.07	0.683	0.396
	Multiplex	**22.11**	**0.719**	**0.347**
80%	Standard	21.74	0.686	0.386
	Multiplex	**22.38**	**0.722**	**0.351**

C. Supplementary Experimental Results

In this section, we present supplementary experimental results.

We first present the quantitative results of DVGO on 4 scenes of T&T Intermediate in Table 11.

We present the quantitative results of DVGO, NeRF, and TensoRF on each T&T scenes in Tables 12, 13, and 14, respectively.

Few-shot learning We evaluate *multiplex training* on a few-shot learning task, where we only keep eight training images. The few-shot learning quantitative results in Table 15 support the significant advantage of *multiplex training* via S3IM.

Table 17. Quantitative results of neural rendering from corrupted training images.

Noise Scale	Training	PSNR(↑)	SSIM(↑)	LPIPS(↓)
0.2	Standard	21.36	0.663	0.453
	Multiplex	**21.94**	**0.686**	**0.420**
0.4	Standard	18.20	0.584	0.569
	Multiplex	**20.89**	**0.636**	**0.494**
0.6	Standard	16.16	0.542	0.684
	Multiplex	**18.06**	**0.571**	**0.599**

Sparse inputs We present the quantitative results of neural rendering from sparse training images in Table 16.

Robustness to corrupted images We present the quantitative results of neural rendering from Corrupted Truck in Table 17.