GENERAL NEURAL GAUGE FIELDS

Anonymous authors
Paper under double-blind review

ABSTRACT

The recent advance of neural implicit representations, such as neural radiance fields, has significantly pushed the boundary of scene representation learning. Aiming to boost the computational efficiency and rendering quality of neural radiance fields of 3D scenes, a popular line of research maps the 3D coordinate system to another measuring system, e.g., 2D manifolds and hash tables, for modeling radiance fields. The conversion of coordinate systems can be typically dubbed as *gauge transformation*, which is usually a pre-defined mapping function, e.g., an orthogonal projection or spatial hash function. This begs a question: can we learn the gauge transformation along with the neural scene representations in an end-to-end manner? In this work, we extend this problem to a general paradigm with a taxonomy of discrete and continuous cases, and develop an end-to-end training framework to jointly optimize the gauge transformation and radiance fields. To counter the problem that the learning of a gauge transformation can collapse easily, we derive a general regularization mechanism from the principle of information conversation during the gauge transformation. On the strength of the derived unified neural gauge field framework, we naturally discover a new type of gauge transformation which achieves a trade-off between learning collapse and computational cost.

1 INTRODUCTION

Representing 3D scenes with high efficiency and quality has been a long-standing target in computer vision and computer graphics research. Recently, the implicit representation of neural radiance fields (Mildenhall et al., 2021) has shown that a 3D scene can be modeled with neural networks, which achieves compelling visual quality and low memory footprint. However, it suffers from long training times. The explicit voxel-based methods (Yu et al., 2021a; Sun et al., 2022) emerged with faster convergence but higher memory requirements, due to the use of the 3D voxel grids as scene representation. To strike a good balance between computational efficiency and rendering quality, EG3D (Chan et al., 2022) proposed to project the 3D coordinate system to a tri-plane system. Along with this line of research, TensoRF (Chen et al., 2022) factorizes 3D space into compact low-rank tensors; Instant-NGP (Müller et al., 2022) models the 3D space with multi-resolution hash grids to enable remarkably fast convergence speed.

These recent works (e.g., EG3D and Instant-NGP) share the same prevailing paradigm by converting the 3D coordinate of radiance fields to another coordinate system. Particularly, a coordinate system of the scene (e.g., 3D coordinate and hash table) can be regarded as a kind of gauge, and the conversion between the coordinate systems can be referred to as **gauge transformation** (Moriyasu, 1983). Notably, existing gauge transformations in radiance fields are usually pre-defined functions (e.g., orthogonal mappings and spatial hash functions (Teschner et al., 2003)), which are sub-optimal for modeling the radiance fields as shown in Fig. 1. In this end, a learnable gauge transformation is more favored as it can be optimized towards the final objective. This raises an essential question: how to learn the gauge transformations along with the radiance fields. Some previous works explore a special case of this problem, e.g., NeuTex (Xiang et al., 2021) and NeP (Ma et al., 2022) aim to transform 3D points into continuous 2D manifolds. However, a general and unified learning paradigm for various gauge transformations has not been established or explored currently.

In this work, we introduce general **Neural Gauge Fields** which unify various gauge transformations in radiance fields with a special focus on how to learn the gauge transformations along with radiance fields. Basically, a gauge is defined by gauge parameters and gauge basis, e.g., codebook indexes

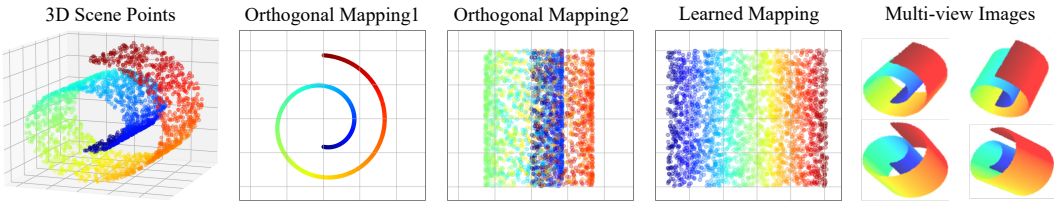


Figure 1: Conceptual illustration of a gauge transformation from 3D point coordinates to a 2D plane. Instead of naively employ a pre-defined orthogonal mapping which incurs overlap, the proposed neural gauge fields aim to learn the mapping along with radiance fields driven by multi-view synthesis loss.

and codebook vectors, which can be continuous or discrete. Thus, the gauge transformations for radiance fields can be duly classified into continuous cases (e.g., triplane space) and discrete cases (e.g., a hash table space). We then develop general learning paradigms for continuous cases and discrete cases, which map a 3D point to a continuous coordinate or a discrete index in the target gauge, respectively.

As typical cases, we study continuous mapping from 3D space to the 2D plane and discrete mapping from 3D space to 256 discrete vectors. As shown in Fig. 2 and 3, we observed that naively optimizing the gauge transformations with the radiance field severely suffers from learning collapse, which means the gauge transformations will collapse to a small region in a continuous case or collapse to a small number of indexes in a discrete case Baevski et al. (2019); Kaiser et al. (2018). To regularize the learning of gauge transformation, a cycle consistency loss has been explored in Xiang et al. (2021); Ma et al. (2022) to avoid many-to-one mapping; a structural regularization is also adopted in Tretschk et al. (2021) to preserve local structure by only predicting a coordinate offset. However, the cycle consistency regularization tends to be heuristic without grounded derivation while the structural regularization is constrained to regularize continuous cases. In this work, we introduce a more intuitive **Info**rmation **Reg**ularization (**InfoReg**) from the principle of information conservation during gauge transformation. By maximizing the mutual information between gauge parameters, we successfully derive general regularization forms for gauge transformations. Notably, a geometric uniform distribution and a discrete uniform distribution are assumed as the prior distribution of continuous and discrete gauge transformations, respectively, where an Earth Mover's distance and a KL divergence are duly applied to measure the distribution discrepancy for regularization.

As InfoReg successfully handles learning collapse by maximizing the mutual information to avoid information loss, we conjecture that a gauge (parameter) with higher information capacity helps to alleviate the learning collapse. Since a discrete gauge transformation maps a 3D point to the most probable (i.e., top-1) index, a gauge parameter with higher information capacity can be achieved by mapping a 3D point to k most probable indexes as the encoding, which is a special case of discrete gauge termed as **Top-k Gauge**. We found that this top-k gauge will gradually obviate the need for regularizations with the increase of k. Notably, with proper selection of used vectors, the top-k gauge can be learned to outperform the strong baseline of spatial hash function. As a tradeoff, the top-k gauge also incurs higher computation cost with the increase of k for encoding.

The contributions of this work are threefold. First, we develop a general framework of neural gauge fields which unifies various gauge transformations in radiance fields, and gives general learning forms for continuous and discrete gauge transformations. Second, we strictly derive a regularization mechanism for gauge transformation learning from the perspective of information conservation during gauge transformation that outperform earlier heuristic approaches. Third, we introduce a new top-k gauge that can achieve a tradeoff between learning collapse and computation cost.

2 Related Work

Recent work has demonstrated the potential of neural radiance fields (Mildenhall et al., 2021) and its extensions for multifarious vision and graphics applications, including fast view synthesis (Liu et al., 2020; Yu et al., 2021b; Hedman et al., 2021; Lindell et al., 2021; Neff et al., 2021; Yu et al., 2021a; Reiser et al., 2021; Sun et al., 2022), generative models (Schwarz et al., 2020; Niemeyer & Geiger,

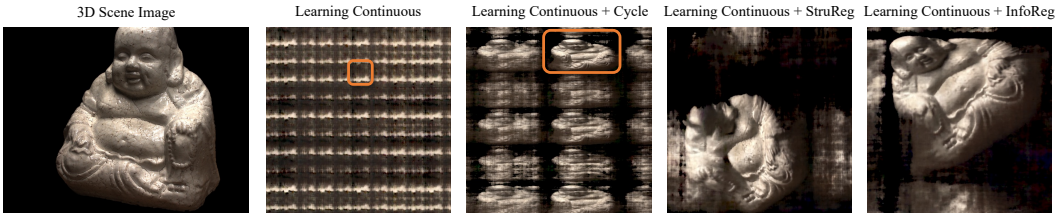


Figure 2: Continuous gauge transformation from 3D scene to 2D square. Without regularization, the learned gauge transformation will collapse to a small region in the 2D square. Including cycle regularization (Xiang et al., 2021) or structural regularization (StruReg) Tretschk et al. (2021) don't fully solve the problem, and our InfoReg achieves a clearly better regularization performance.

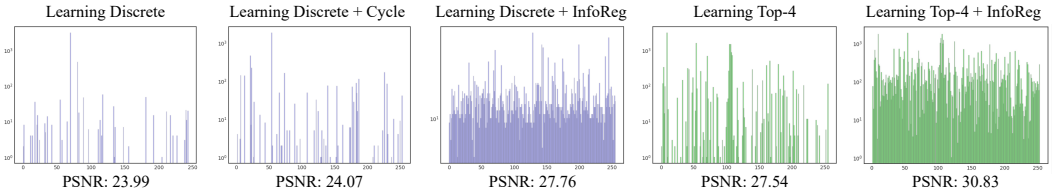


Figure 3: Discrete gauge transformation from 3D scene to 256 discrete vectors. Without regularization, the learned gauge transformation will collapse to a small number of vector indexes (horizontal axis). The learning with cycle regularization (Xiang et al., 2021) is still significantly collapse-prone, while our InfoReg enables to alleviates the collapse substantially.

2021; Gu et al., 2022; Chan et al., 2021; Or-El et al., 2022), surface reconstruction (Wang et al., 2021; Oechsle et al., 2021; Yariv et al., 2021), etc. Under this context, various gauge transformations have been intensely explored in radiance fields for different purposes, e.g., reducing computation cost (Chan et al., 2022), fast convergence (Chen et al., 2022; Müller et al., 2022), editable UV map (Xiang et al., 2021; Ma et al., 2022).

A pre-defined mapping function is usually adopted as gauge transformation. A simple case is orthogonal mapping, which projects 3D space into a continuous low-dimension space. For instance, EG3D (Chan et al., 2022; Peng et al., 2020) orthogonally projects 3D space into single plane or triplane space for scene modelling. TensoRF (Chen et al., 2022) further extends this line of research for fast and efficient optimization by mapping the 3D scene space into a set of low-dimension tensors. With a pre-defined spatial hash function, the recent Instant-NGP (Müller et al., 2022) maps 3D grid points into hash table indexes and achieves a speedup of several orders of magnitude. Besides, PREF (Huang et al., 2022) explores to transform 3D scene points to a set of Fourier basis via Fourier transformation, which helps to reduce the aliasing caused by interpolation.

Some works also explore learning the gauge transformation for downstream tasks in neural radiance fields. For example, NeuTex (Xiang et al., 2021) and NeP (Ma et al., 2022) propose to learn the mapping from 3D points to 2D texture spaces; GRAM (Deng et al., 2022) learns to map the 3D space into a set of implicit 2D surfaces; VQ-AD Takikawa et al. (2022) enables compressive scene representation by mapping coordinates to discrete codebook, which is a discrete case of gauge transformation. However, all above works suffer from learning collapse without regularization or designed initialization. Besides, a popular line of research (Tretschk et al., 2021; Tewari et al., 2022; Park et al., 2020; Pumarola et al., 2020; Liu et al., 2021; Peng et al., 2021) learns the deformation from the actual 3D space into another (canonical) 3D space, where a structural regularization is usually applied by predicting a deformation offset instead of the absolute coordinate.

3 METHODS

This section will introduce the framework of neural gauge fields in detail, followed by the derivation of the learning regularization based on information conservation during gauge transformation. A new top-k gauge and some insights for the tradeoff between the learning collapse and computation cost are drawn in the end.

3.1 NEURAL GAUGE FIELDS

A gauge system is defined by a gauge basis and gauge parameters, where the gauge basis is a measuring space (e.g., triplane in EG3D (Chan et al., 2022) and hash table in Instant-NGP (Müller et al., 2022)). The gauge parameters indicate the specific position in the measuring space (e.g., triplane coordinate in EG3D (Chan et al., 2022) and hash table index in Instant-NGP (Müller et al., 2022)). Based on this intuition, the conversion between gauges in radiance fields can be termed as *gauge transformation*. In this work, we introduce the general framework of **Neural Gauge Fields** which consists of a gauge transformation and a radiance field. A neural gauge field aims to transform the original 3D space to another gauge system which is then used to index a radiance field. This additional transform could introduce certain bonus to the radiance field, e.g., low memory cost, high rendering quality, or explicit texture, depending on the purpose of the model. This work focuses on how to learn a decent gauge transformation for radiance field regardless of the specific purposes.

The gauge transformation could be a pre-defined function e.g., an orthogonal mapping in EG3D and a spatial hash function in Instant-NGP, or directly learned by the neural network along with radiance fields. Intuitively, compared with a pre-defined function, a learnable gauge transformation is more favorable as it can be optimized towards the final use case of the neural radiance field and possibly yields better performance. For learning neural gauge fields, we disambiguate between two cases: continuous (e.g., 2D plane and sphere surface) and discrete (e.g., hash table space) mappings.

Continuous Gauge Transformation. As the target gauge system is continuous, the gauge transformation can be handled by directly regressing the coordinate (i.e., gauge parameters) in the target gauge. For a point $x \in \mathbb{R}^3$ in the 3D space and a target gauge, a gauge network $\mathcal{M}$ (a MLP structure or transformation matrix) can be employed to predict the target coordinate as $x' = \mathcal{M}(x), x' \in \mathbb{R}^N$, where N is the coordinate dimension of the target gauge. Then, the predicted coordinate x' can be fed into an MLP or used to look up feature vectors to model radiance fields.

Discrete Gauge Transformation. Without loss of generality, we assume there is a codebook with N discrete vectors $[v_1, v_2, \cdots, v_N]$ as the gauge basis, and the discrete gauge transformation aims to transform a 3D point to a discrete codebook index (i.e., gauge parameter). In line with the setting of Instant-NGP, we divide the 3D space into a grid of size $M * M * M$. For each grid point x, we apply a gauge network $\mathcal{M}$ to predict its discrete distribution $P_x \in \mathbb{R}^N$ over the N discrete vectors, followed by an Argmax operation to select the most probable index i_x as below:

$$P_x = \mathcal{M}(x), \;\; i_x = Argmax(P_x) \qquad P_x \in \mathbb{R}^N, \;\; i_x \in \mathbb{N}^1 \tag{1}$$

The produced indexes for grid points are usually used to lookup corresponding vectors from the codebook [1], and the features of other 3D points can be acquired by interpolating the nearby grid point vectors. As the Argmax operation is not differentiable, a reparameterization trick (Bengio et al., 2013) is applied to approximate gradient by replacing Argmax with Softmax in back-propagation. The pseudo code of the forward & backward propagation of discrete cases is given in Algorithm 1.

We now have established the general learning paradigms for continuous and discrete gauge transformations. In our setting, the gauge transformations are optimized along with the radiance field supervised by the per-pixel color loss between the rendering and the ground-truth multi-view images. However, we found the learning of gauge transformations is easily stuck in local minimal. Specifically, the gauge transformation tends to collapse to a small region in continuous cases as in Fig. 2 or collapse to a small number of indexes in discrete cases as in Fig. 3, highlighting the need for an additional regularization to enable an effective learning of continuous and discrete gauge transformations, which we will introduce in the following.

Algorithm 1 Pseudo code of forward & backward propagation in learning discrete gauge transformation.

Input: A 3D point x, predicted distribution $P_x = [p_1, p_2, \cdots, p_N]$, codebook $\mathcal{V}$.

Forward propagation:

1. index = Argmax(P_x)
2. index_hard = One_Hot(index)
3. feature lookup: v_x = Matmul(index_hard, $\mathcal{V}$)

Backward propagation:

1. index_soft = Softmax(P_x)
2. feature lookup: v_x = Matmul(index_soft, $\mathcal{V}$)

Output: feature v_x.

[1] Actually, the predicted indexes can also be fed into an MLP for embedding, but this manner tends to be meaningless in practical applications and is seldomly adopted.

3.2 LEARNING REGULARIZATION

Some previous works (Xiang et al., 2021; Ma et al., 2022) leverage a cycle consistency loss to regularize the mapping from a 3D space to a 2D UV space to prevent many-to-one mapping of 3D points. Alternatively, a structural regularization is also employed in some works (Tretschk et al., 2021; Tewari et al., 2022) to preserve local structure by predicting a residual offset instead of the absolute coordinate.

However, the cycle loss only works for certain continuous cases and shows poor performance for the regularization of discrete cases, while the structural regularization cannot be applied to discrete cases at all. In this work, we strictly derive a general regularization mechanism for gauge transformation from the perspective of information conservation which is superior to the individual regularizers used in prior work. Specifically, although the gauge of the scene has been changed, the intrinsic information of the scene is expected to be preserved during gauge transformation, which can be achieved by maximizing the mutual information between the original 3D points and target gauge parameters.

With the gauge network $\mathcal{M}$ for gauge transformation, the sets of original 3D points and target gauge parameters can be termed as X and $Y = \mathcal{M}(X)$, respectively. Then, the mutual information between X and Y can be formulated as:

$$I(X,Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy = \iint p(y|x)p(x) \log \frac{p(y|x)}{p(y)} dxdy, \tag{2}$$

where $p(x)$ and $p(y)$ are the distributions of original 3D point x and target gauge parameter y, $p(y|x)$ is the conditional distribution of y produced by the gauge transformation of x. To avoid that the distribution of Y collapses into a local region or small number of indices, we assume that Y obeys a prior uniform distribution $q(y)$, and the KL divergence between $p(y)$ and $q(y)$ can be written as $KL(p(y)||q(y)) = \int p(y) \log \frac{p(y)}{q(y)} dy$. Then, the final regularization loss can be formulated as:

$$\mathcal{L}_{reg} = \min_{p(y|x)} \left\{ -\gamma \iint p(y|x)p(x) \log \frac{p(y|x)}{p(y)} dxdy + \epsilon \int p(y) \log \frac{p(y)}{q(y)} dy \right\}, \tag{3}$$

where γ and ϵ denote the weight of regularization and prior distribution discrepancy. This translates into the following term (see derivation in the Appendix):

$$\mathcal{L}_{reg} = \min_{p(y|x)} \Big\{ -(\gamma + \epsilon) \cdot \mathbb{E}[KL(p(y|x)p(x)||p(y)p(x))] + \epsilon \cdot \mathbb{E}[KL(p(y|x)||q(y))] \Big\}. \tag{4}$$

Regarding the first term $KL(p(y|x)p(x)||p(y)p(x))$ in Eq. 4, we can replace the unbounded KL divergence with the upper-bounded JS divergence as: $JS(p(y|x)p(x)||p(y)p(x))$.

Lemma 1 *Jensen-Shannon Mutual Information Estimator (Nowozin et al., 2016):*

$$JS(P,Q) = \max_T \Big(\mathbb{E}_{x \sim p(x)}[-sp(-T(x))] - \mathbb{E}_{x \sim q(x)}[sp(T(x))] \Big), \tag{5}$$

where $sp(z) = log(1 + e^z)$. For $JS(p(y|x)p(x)||p(y)p(x))$, we thus have:

$$\max_T \left(\mathbb{E}_{(x^+,y) \sim p(y|x)p(x)}[-sp(-T(x^+,y))] - \mathbb{E}_{(x^-,y) \sim p(y)p(x)}[sp(T(x^-,y))] \right), \tag{6}$$

where T denotes a network which minimizes the distance of positive pairs and maximizes the distance of negative pairs. Please refer to section 4.1 for the details of T and the selection of positive & negative pairs.

Now the first term of Eq. (4) has been well solved. We then turn to the second term $\mathbb{E}[KL(p(y|x)||q(y))]$ (i.e., the KL divergence between $p(y|x)$ and $q(y)$). This term can be solved according to the types of prior uniform distributions $q(y)$ assumed in continuous and discrete cases.

Continuous Gauge Transformation. To prevent that the gauge transformation collapses to a local region, we expect the prior distribution $q(y)$ is a uniform distribution over the target gauge space. Without loss of generality, we assume the target gauge space is a 2D plane as shown in Fig. 4 (a). Then, we uniformly sampled h points on the 2D plane as a discrete approximation of the prior uniform distribution, which is denoted by $\overline{q}(y) = [u_1, u_2, \cdots, u_h], u_i \in \mathbb{R}^2$). With a set of 3D

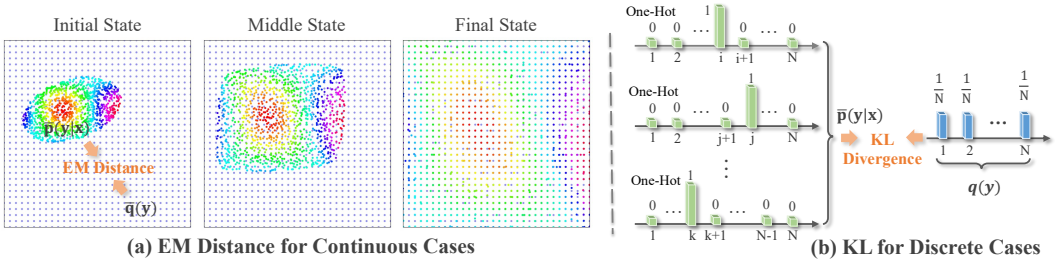


Figure 4: (a) Earth Mover's distance to regularize the discrepancy between $\overline{p}(y|x)$ and $\overline{q}(y)$ in continuous case (2D plane), where $\overline{p}(y|x)$ and $\overline{q}(y)$ are predicted 2D points within a batch and uniformly sampled points in the 2D plane, respectively. (b) KL divergence to regularize the distribution discrepancy in discrete cases (N basis vectors), where $\overline{p}(y|x)$ and $q(y)$ are the average of predicted one-hot distributions and a discrete uniform distribution, respectively.

points transformed into 2D points, the conditional distribution $p(y|x)$ can also be approximated by sampling h points from the predicted 2D points according to the radiance contribution (mentioned previously), which is denoted by $\overline{p}(y|x) = [v_1, v_2, \cdots, v_h], v_i \in \mathbb{R}^2$. Considering $\overline{q}(y)$ and $\overline{p}(y|x)$ are distributions over a geometric space, KL divergence is inapposite to measure their discrepancy as the distribution geometry is not concerned in KL divergence. We thus resort to a geometry-aware metric: Earth Mover's (EM) distance (Rubner et al., 2000).

Here, Earth Mover's distance measures the discrepancy between two distributions by calculating the minimum work required to move one distribution to another on the 2D square, where the work is measured by multiplying the amount of distribution to be moved and the distance to be moved. Thus, a cost matrix C and transport matrix M with size of (h, h) can be defined where each entry C_{ij} in C gives the distance between points v_i and u_j, each entry M_{ij} in M represents the amount of probability moved between point v_i and point u_j. Then, the Earth Mover's distance between $\overline{q}(y)$ and $\overline{p}(y|x)$ can be formulated as: $\mathcal{L}_{EM} = \min_M(\sum_{i=1}^{h}\sum_{j=1}^{h} C_{ij}M_{ij})$. The distribution of predicted points will be regularized to fit the uniform distribution by minimizing $\mathcal{L}_{EM}$ as shown in Fig. 4 (a).

Discrete Gauge Transformation. For discrete gauge transformation, we expect the prior distribution $q(y)$ to be a uniform distribution over the discrete vectors (i.e., the gauge basis), so that the target gauge parameters will not collapse to a small number of vector indexes. Thus, the prior distribution $q(y)$ can be denoted by $q(y) = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$ for N discrete vectors. As the one-hot distribution p_i is predicted from 3D points as described in Algorithm 1, the conditional distribution $p(y|x)$ can be approximated by the average of all predicted one-hot distributions over a training batch, which is denoted by $\overline{p}(y|x) = \sum_{i=1}^{B} p_i/N$ where B is the number of samples within a training batch. Then the KL divergence between $q(y)$ and $\overline{p}(y|x)$ can be duly computed as shown in Fig. 4 (b).

3.3 Top-k Gauge

As the InfoReg maximizes the mutual information between gauge parameters to avoid loss of information, we conjecture that a target gauge with higher information capacity may help to alleviate the loss of information and reduce the need for regularizations, which gives rise to a top-k gauge that is described in the following.

As a concrete example, suppose there is a gauge basis with N discrete vectors $[v_1, v_2, \cdots, v_N]$, and the discrete gauge transformation maps a 3D grid point to one of the vectors. Thus, a gauge parameter with higher information capacity can be achieved by mapping a 3D point to multiple vectors for encoding, which is termed as **Top-k Gauge**. Specifically, the top-k gauge feeds a 3D grid point into a gauge network $\mathcal{M}$ to predict a distribution $[p_1, p_2, \cdots, p_N]$. To select the multiple indexes for encoding, a differentiable top-k operation (please refer to Appendix for more detail) can be applied to the distribution to select k most probable vectors to model radiance fields. Notably, we intriguingly discover the top-k gauge will suffer less from learning collapse with the increase of used vectors for encoding as shown in Fig. 7. Note, a top-k gauge with $k = 1$ is equivalent to the standard discrete gauge. On the other hand, the top-k gauge will lead to slightly higher memory cost and slower training speed with the increase of used vectors. Thus, there is a tradeoff between

Regularizations	Continuous Gauge (2D Plane)				Discrete Gauge (256 Vectors)			
	S-NeRF	S-NSVF	DTU	T&T	S-NeRF	S-NSVF	DTU	T&T
Baseline (w/o Regularization)	23.38	24.52	24.14	22.87	22.31	23.02	22.91	22.04
+StruReg (Tretschk et al., 2021)	28.54	30.82	26.51	25.49	N/A	N/A	N/A	N/A
+CycleReg (Xiang et al., 2021)	29.72	31.67	26.08	25.02	22.46	23.14	23.18	22.12
+InfoReg	**30.74**	**32.51**	**27.06**	**25.95**	**26.56**	**29.13**	**26.21**	**24.79**

Table 1: The PSNR performance with different regularizations for the learning of continuous gauge transformation (from 3D space to 2D square) and discrete gauge transformation (from 3D space to 256 basis vectors).

Gauge Transformations	S-NeRF		S-NVSF		DTU		T&T	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Orthogonal Mapping	18.20	0.816	19.32	0.833	23.31	0.845	19.97	0.829
Learning Continuous Gauge *	**30.74**	**0.919**	**32.51**	**0.949**	**27.06**	**0.873**	**25.95**	**0.867**
Spatial Hash (Teschner et al., 2003)	27.06	0.902	29.81	0.932	27.02	0.876	26.22	0.869
Learning Discrete Gauge *	26.56	0.893	29.13	0.925	26.21	0.863	24.79	0.855
Learning Top-4 Gauge *	**29.39**	**0.917**	**31.39**	**0.942**	**28.77**	**0.887**	**27.14**	**0.892**

Table 2: Performance of different methods (pre-defined or learned) for gauge transformations (from 3D space to a 2D plane and 256 basis vectors). * denotes the InfoReg is used for learning regularization. Orthogonal mapping and spatial hash function are typical methods for continuous and discrete gauge transformation, respectively.

computation cost and learning collapse which can be controlled while designing a learnable gauge transformation. Please refer to section 4.2.2 for the analysis of trade-off in top-k gauge.

4 EXPERIMENTS

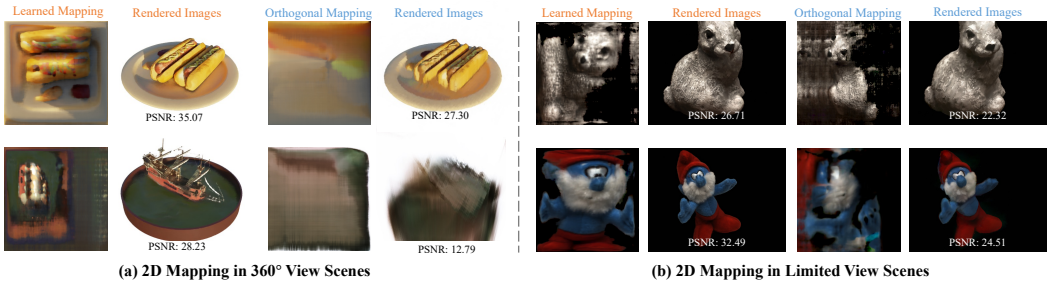
4.1 IMPLEMENTATION

The gauge network $\mathcal{M}$ for learning gauge transformations can be a MLP network or a transformation matrix, depending on the specific downstream application. For the mapping from 3D space to 2D plane to get (view-dependent) textures, the radiance field is modeled by a MLP-based network which takes predicted 2D coordinates, i.e. output of the gauge network, and a certain view to predict color and density. For the mapping from 3D space to discrete codebooks, the radiance field is modeled by looking up features from the codebook, followed by a small MLP of two layers to predict color and density. By default, the codebook has two layers and each layer contains 256 vectors with 128 dimensions. In line with Instant-NGP (Müller et al., 2022), the 3D space is also divided into two-level 3D grids with size $16 \times 16 \times 16$ and $32 \times 32 \times 32$ for discrete gauge transformation.

For the implementation of our InfoReg, the network T first maps the gauge parameters into features. Specifically, a 5-layer MLP is employed in T to encode the target gauge parameters y as features y', and the encoded feature x' of original 3D points x can be directly taken from the fifth layer of the gauge network. The distance between constructed pairs can be formulated as: $T(x, y) = d(norm(x'), norm(y'))$, where $norm$ denotes normalization, d is a distance metric, e.g., Euclidean distance (used by default) and cosine distance. During training, an input 3D point and the corresponding transformed gauge parameters form a positive pair (x^+, y), the negative sample x^- is randomly sampled from the training batch. Note InfoReg only uses one negative sample which is obtained by shuffling the input 3D points. Besides, as 3D points near the scene surface are of more significance, we leverage the radiance contribution per shading point (Xiang et al., 2021) to weigh the mutual information. Please refer to the Appendix A.3 and A.5 for a more detailed implementation and experimental settings.

4.2 EVALUATION

The introduced neural gauge fields focus on learning gauge transformation, the experiments are therefore performed to evaluate the performance of gauge transformations. Specifically, as a gauge



(a) 2D Mapping in 360° View Scenes **(b) 2D Mapping in Limited View Scenes**

Figure 5: Qualitative results of continuous gauge transformation from 3D space to 2D plane on (a) 360° view scenes and (b) limited view scenes.

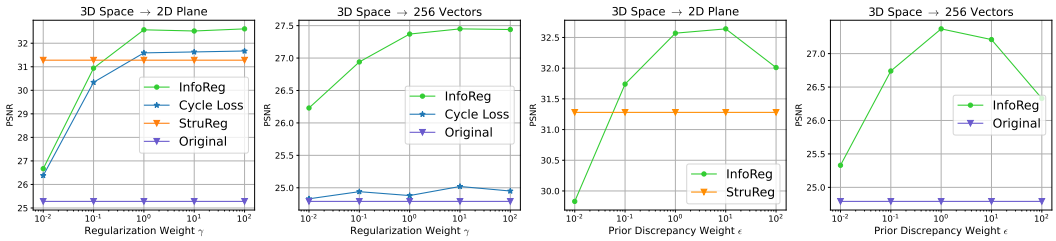


Figure 6: Ablation study of the regularization weights and prior distribution discrepancy weight.

transformation is applied to index a radiance field, the gauge transformation can be evaluated by the novel view synthesis performance of a radiance field. Besides, a good gauge transformation should also well utilize the target gauge space, which can be evaluated by visualizing the mapping results on 2D plane or computing the codebook utilization. In this end, the experiments are mainly conducted on continuous gauge transformation from 3D space to 2D plane and discrete gauge transformation from 3D space to 256 codebook vectors.

4.2.1 Continuous & Discrete Gauge

Regularization. We compare the performance of different regularizations including structural regularization (StruReg), Cycle loss (CycleReg), and our InfoReg. As tabulated in Table 1, the proposed InfoReg outperforms other regularization clearly in both continuous and discrete cases. Particularly, StruReg and CycleReg can bring certain improvements for continuous cases, while StruReg cannot be applied to discrete cases and CycleReg presents limited performance for discrete cases.

Pre-defined vs Learned Mapping. As shown in Table 2, we compare the performance of the pre-defined functions and the learned mapping for gauge transformation. For continuous cases, the learned mapping with InfoReg outperforms the pre-defined orthogonal mapping clearly. For discrete cases, the standard learned discrete mapping (top-1 gauge) is slightly inferior to the pre-defined spatial hash function, while the top-4 variant of the learned discrete mapping surpasses the pre-defined function substantially.

Quantitative Results. To demonstrate the superiority of learned gauge transformation over pre-defined function (i.e., orthogonal mapping), we visualize the mapping results from 3D space to 2D plane in Fig. 5. Fig. 5 (a) illustrates the mapping on the challenging 360° view scenes. The learned mapping can still well capture the scene texture, while the orthogonal mapping presents poor mapping performance and even fails in certain case. Fig. 5 (b) compares the mapping on limited view scenes. The orthogonal mapping is still prone to lose much texture information thanks to the inherent inflexibility.

Parameter Study of InfoReg. Fig. 6 examines the effect of the regularization weight γ and prior discrepancy weight ϵ in Eq. (4) on the Lego scene. With the increase of regularization weight, the regularization performance of InfoReg is improved consistently, although it is saturated with a large weight. The regularization performance presents a positive correlation with a small prior discrepancy weight and a negative correlation with a large prior discrepancy weight.

Models	Model Training Details			Synthetic-NeRF		Synthetic-NSVF	
	Steps↓	Time(s)↓	Size(MB)↓	PSNR↑	SSIM↑	PSNR↑	SSIM↑
TensoRF	30k	708	67.19	32.32	0.958	35.32	0.976
TensoRF+NGF	30k	721	67.47	**32.96**	**0.962**	**35.81**	**0.979**
Instant-NGP	30k	458	96.84	32.43	0.960	35.43	0.977
Instant-NGP+NGF	30k	496	97.25	**32.91**	**0.962**	**36.01**	**0.981**

Table 3: The rendering performance gain by applying our neural gauge fields (NGF) to SOTA methods on Synthetic-NeRF and Synthetic-NSVF. Notably, only very slight computation and memory cost are introduced with the including of neural gauge fields.

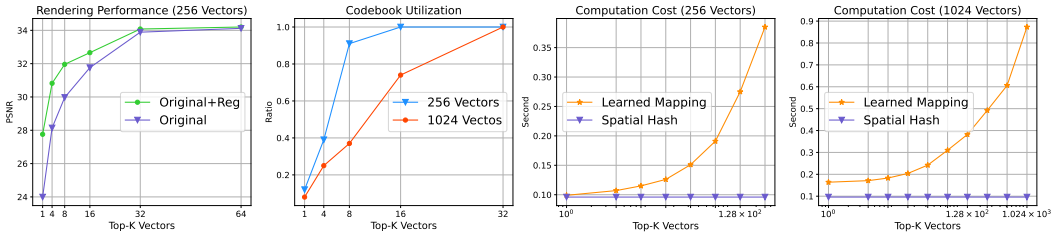


Figure 7: The rendering performance, codebook utilization, and computation cost with different k values in top-k gauges. The experiment is conducted on the Lego scene of synthetic-NeRF dataset.

Combination with SOTA Methods. To demonstrate the power of learning gauge transformation, we apply our neural gauge fields (NGF) to SOTA methods including TensoRF (initial 64×64 planes with 192 dimension which is gradually upsampled to 192×192) and Instant-NGP (10 layers with 2^{14} to 2^{23} entries). To optimizing gauge transformations, three planes of 1 dimension are employed to indicate the coordinate offset of three feature planes in TensoRF+NGF, ten grids of 2^8 dimensions are employed to indicate the codebook distribution in Instant-NGP+NGF. Please refer to the Appendix A.5 for more detailed setting. As shown in Table 3, learning gauge transformation benefits these SOTA methods consistently with ignorable (or very slight) increase of training time and model size.

4.2.2 Top-k Gauge

Trade-off between Collapse and Cost. Fig. 7 illustrates the rendering performance, codebook utilization, and computation cost with different k values. The rendering performance and codebook utilization (ratio of used vectors) both are improved consistently with the increase of k, which means the learning collapse is gradually alleviated. The performance gain brought by InfoReg is also narrowed with the increase of k. However, the increase of k incurs the significant increase of computation cost, which may hinder the practical application of the model.

5 Conclusion

This work introduces a general framework of neural gauge fields for unifying various gauge transformations in radiance fields. With a target to optimize the gauge transformation along with radiance fields, we establish the learning paradigm for continuous and discrete gauge transformations and derive a regularization mechanism to solve the issue of learning collapse from the perspective of information preservation. To further explore the learning collapse, a top-k gauge is introduced to achieve a trade-off between learning collapse and computation cost. The extensive experiments and thorough analysis show that the learned gauge transformation outperforms its pre-defined counterpart, which paves the way for learning gauge transformations in rendering pipelines.

6 ETHICS STATEMENT

Our neural gauge fields focus on technically advancing neural scene representations and are not biased towards specific race, gender, or region. As a general framework, neural gauge fields can be applied to various fields, e.g., virtual reality, artistic creation, and game design. On the other hand, the achievements in the image synthesis quality lurk the risk of being misused, e.g., illegal impersonation of actors, or face synthesis for malicious purposes. Therefore, it is also of importance to explore safeguarding technologies that can reduce the potential for misuse. One potential way is to use our approach as well as other image synthesis approaches as a discriminator to identify the synthesized images automatically. This has been actively explored by the community, e.g. Mirsky & Lee (2021). Alternatively, one should explore watermarking technologies to identify synthesized images. Besides, access control should also be carefully considered when deploying the relevant techniques to alleviate the risk of misuse. Finally, we believe creating appropriate regulations and laws is essential to solving the issue of technology misuse while promoting its positive effects on technology development.

7 REPRODUCIBILITY STATEMENT

We attach the source code of neural gauge fields in the supplementary material. Additionally, we provide detailed implementations in the manuscript that enable a full reproduction of the experimental results. For example, we specify the choices of rendering resolution, model architecture, and the codebook setting for discrete gauge transformation. All datasets used in our experiments are publicly accessible. Finally, all pre-trained models will be made publicly available upon the publication of this work. The implementations will enable researchers to reproduce the results as well as extend this work to future research.

REFERENCES

Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *International Conference on Learning Representations*, 2019.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5799–5809, 2021.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022.

Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. Gram: Generative radiance manifolds for 3d-aware image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10673–10683, 2022.

Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations*, 2022.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5875–5884, 2021.

Binbin Huang, Xinhao Yan, Anpei Chen, Shenghua Gao, and Jingyi Yu. Pref: Phasorial embedding fields for compact neural representations. *arXiv preprint arXiv:2205.13524*, 2022.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pp. 2390–2399. PMLR, 2018.

David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14556–14565, 2021.

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.

Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Trans. Graph.(ACM SIGGRAPH Asia)*, 2021.

Li Ma, Xiaoyu Li, Jing Liao, Xuan Wang, Qi Zhang, Jue Wang, and Pedro Sander. Neural parameterization for dynamic human head editing. *arXiv preprint arXiv:2207.00210*, 2022.

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.

Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, 54(1):1–41, 2021.

Keihachiro Moriyasu. *An elementary primer for gauge theory*. World Scientific, 1983.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022.

Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, 2021.

Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11453–11464, 2021.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

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*, pp. 5589–5599, 2021.

Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13503–13513, June 2022.

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14314–14323, 2021.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pp. 523–540. Springer, 2020.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes, 2020.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *International Conference on Computer Vision (ICCV)*, 2021.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020.

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*, pp. 5459–5469, 2022.

Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–9, 2022.

Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, volume 3, pp. 47–54, 2003.

Ayush Tewari, Xingang Pan, Ohad Fried, Maneesh Agrawala, Christian Theobalt, et al. Disentangled3d: Learning a 3d generative model with disentangled geometry and appearance from monocular images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1516–1525, 2022.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12959–12970, 2021.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Robert M Wald. General relativity. 2010.

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.

Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7119–7128, 2021.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021a.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5752–5761, 2021b.

A Appendix

A.1 GAUGE TRANSFORMATION BACKGROUND

In normal usage, gauge means a measuring instrument or measuring system, e.g., pressure gauge and temperature gauge. The transformation between different gauges (e.g., Fahrenheit and Celsius temperature) can be termed as **Gauge Transformation**. In physics, a gauge is a particular choice or specification of vector and scalar potentials which will generate a physical fields. A transformation from one physical field configuration to another is called a gauge transformation. The gauge transformations for classical general relativity Wald (2010) are arbitrary coordinate transformations. Fig. 8 illustrates a simple gauge transformations of Cartesian coordinate.

Under the context of neural fields, the measuring systems (i.e. gauge) of the 3D scene can be extended to more general definition, e.g., 3D Cartesian coordinate system, 2D coordinate system, hash table space, and frequency space. The transformation between different scene measuring system is dubbed as **Gauge Transformation**, and the neural fields with certain gauge transformation is termed as **Neural Gauge Fields**.

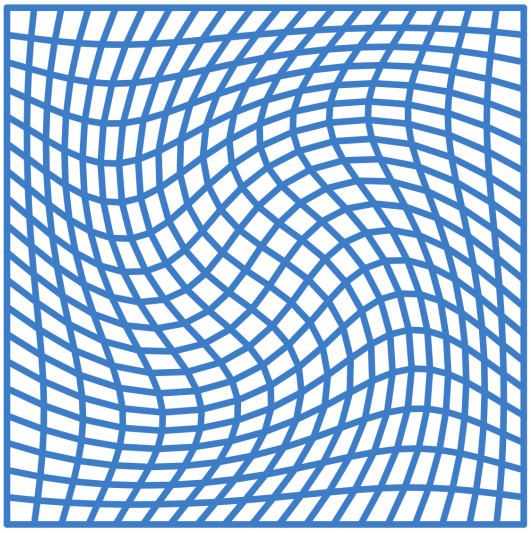


Figure 8: The Cartesian coordinate grid is distorted by a coordinate transformation. The laws of general relativity Wald (2010) are still valid in the new coordinate system. Such changes of coordinate system are the gauge transformations of general relativity in physics.

A.2 DERIVATION OF INFOREG

According to Eq. (2), the final optimization loss can be formulated as:

$$\mathcal{L}_{reg} = \min_{p(y|x)} \left\{ -\gamma \iint p(y|x)p(x) \log \frac{p(y|x)}{p(y)} dxdy + \epsilon \int p(y) \log \frac{p(y)}{q(y)} dy \right\}, \tag{7}$$

where γ and ϵ denote the weight of regularization and prior distribution discrepancy. Then, we have:

$$\begin{aligned} L_{reg} &= \min_{p(y|x)} \left\{ -\gamma \iint p(y|x)p(x) \log \frac{p(y|x)}{p(y)} dxdy + \epsilon \iint p(x|y)p(y) \log \frac{p(y)p(y|x)}{q(y)p(y|x)} dxdy \right\} \\ &= \min_{p(y|x)} \left\{ \iint p(y|x)p(x) \Big[-(\gamma+\epsilon) \log \frac{p(y|x)}{p(y)} + \epsilon \log \frac{p(y|x)}{q(y)} \Big] dxdy \right\} \\ &= \min_{p(y|x)} \left\{ -(\gamma+\epsilon) \cdot I(X,Y) + \epsilon \cdot \mathbb{E}[KL(p(y|x)||q(y))] \right\} \end{aligned} \tag{8}$$

For $I(X,Y)$, we have:

$$I(X,Y) = \iint p(y|x)p(x) \log \frac{p(y|x)p(x)}{p(y)p(x)} = KL(p(y|x)p(x)||p(y)p(x)) \tag{9}$$

Thus, we can get:

$$\mathcal{L}_{reg} = \min_{p(y|x)} \Big\{ -(\gamma + \epsilon) \cdot \mathbb{E}[KL(p(y|x)p(x)||p(y)p(x))] + \epsilon \cdot \mathbb{E}[KL(p(y|x)||q(y))] \Big\}, \tag{10}$$

which can be further solved as described in the main paper.

A.3 Radiance Contribution as InfoReg Weight

As 3D points near the scene surface are of more significance in mutual information, we leverage the radiance contribution weights per shading point to weight the mutual information term as in Xiang et al. (2021). Specifically, with volume density σ and radiance c of a 3D scene, the RGB color I can be computed by aggregating the radiance values of shading points on the ray as below:

$$I = \sum_i T_i(1 - \exp(1 - \sigma_i \delta_i))c_i, \quad T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \tag{11}$$

where i denotes the index of a shading point on the ray, σ_i and c_i denote the density and radiance at shading point i, δ_i denotes the distance between two consecutive points, T_i is the transmittance. Thus, the radiance contribution weight of a shading point can be formulated as $w_i = T_i(1 - \exp(\sigma_i \delta_i))$. The w_i is applied to weight the positive pairs in Eq. (6).

A.4 Top-k Gauge Implementation

Differentiable Top-k. With a predicted discrete distribution $P_x = [p_1, p_2, \cdots, p_N]$, the top-k operation aims to select the k most probable index in a differentiable way so that the gradient can be back-propagated. For forward-propagation, we use the efficient top-k operation implemented in PyTorch to select the top-k values and indexes. For back-propagation, we apply the reparameterization trick (Bengio et al., 2013) to approximate the gradient by replacing the top-k operation with Softmax. The detailed pseudo code of the differentiable top-k operation is given in Algorithm 2.

Algorithm 2 Pseudo code of differentiable top-k operation.

Input: k values in top-k, a 3D point x, predicted distribution $P_x = [p_1, p_2, \cdots, p_N]$, codebook $\mathcal{V}$.

Forward propagation:

1. topk_init=[0, 0, $\cdots$, 0] $\in \mathbb{R}^N$
2. index, value = Top-k(P_x)
3. topk_hard = topk_init.scatter(index, value)
4. value_sum = value.sum()
5. topk_hard = topk_hard $\div$ value_sum
6. feature lookup: v_x = Matmul(topk_hard, $\mathcal{V}$)

Backward propagation:

1. index_soft = Softmax(P_x)
2. feature lookup: v_x = Matmul(index_soft, $\mathcal{V}$)

Output: feature v_x.

Grid Upsampling. The information capacity of grid points is gradually improved with the increase of k value in top-k. To better utilize the information capacity for modeling, the 3D grid resolution is also increased accordingly by $\log(k)$ times, e.g., the two-level 3D grids of top-4 gauge are of sizes $16 \cdot \log(k) \times 16 \cdot \log(k) \times 16 \cdot \log(k)$ and $32 \cdot \log(k) \times 32 \cdot \log(k) \times 32 \cdot \log(k)$.

A.5 Implementation and Experimental Setting

We implemented our neural gauge fields and re-implemented all compared methods without customized CUDA kernels. All models are optimized for 150k steps with a batch size of 1024 pixel rays.

Efficient Gauge Transformation. The proposed neural gauge fields can be applied to benefit existing models without increasing their computational cost and training speed significantly. Here,

we provide several strategies to achieve it: First, instead of using MLPs to learn the gauge transformation, we can directly optimize tensors for transformation, e.g., 2D offset grids to indicate the coordinate transformation in TensoRF, 1D vectors for grid points to indicate codebook index distribution in Instant-NGP. Thus, the convergence speed of gauge transformation can be aligned with grid-based models like TensoRF and Instant-NGP. Second, gauge transformation learning can be performed in the middle training stage instead of the full training process, then the learned transformation is frozen for ensuing model training. Thus, the computation cost introduced by learning gauge transformation (and regularization) can be ignorable compared with the total training time. Third, we empirically find that the prior distribution regularization term in InfoReg has been enough to avoid learning collapse in most cases, which is much more efficient than the complete InfoReg.

A.6 Limitations

This work focuses the fundamental problem of learning gauge transformation along with radiance fields and is orthogonal to the SOTA methods for scene representation, e.g., TensoRF and Instant-NGP. Although we demonstrate that the learned gauge transformation can outperform its pre-defined counterpart (e.g., orthogonal mapping and spatial hash function), there are still some challenges to apply this method to the SOTA models.

For instance, the original Instant-NGP employs a 16-layer codebook with a size ranging from 2^{14} to 2^{24}. When replacing the spatial hash function in Instant-NGP with the learned counterpart, the gauge network size will be extremely huge with an output layer dimension ranging from 2^{14} to 2^{24}. One possible solution is firstly grouping the codebook into several sub-blocks with smaller size, followed by smaller gauge network to learn the mapping to each block. Specifically, a large codebook (e.g., 2^{20}) can be divided into a set of sub-codebooks (e.g., 2^8 sub-codebooks with of size 2^{12}). Similarly, a grid space $256 \times 256 \times 256$ can be divided into a set of sub-grids (512 sub-grids of size $32 \times 32 \times 32$). Then, with our neural gauge field framework, the mapping between the sub-grids and sub-codebook can be learned, where a **very small output layer** can be employed for mapping (i.e., 2^8=256). For grid points within the sub-grid, the spatial hash function can be applied to achieve the mapping. Overall, the output layer size can be adjusted freely according to the selection of sub-codebook size. This strategy is adopted for the experiments in Table 3.

A.7 Discussion & Future Work

As a general framework, the proposed neural gauge fields can be extended in several directions. Firstly, this work focuses on radiance fields for novel view synthesis, which can be easily extended to other tasks, e.g., surface reconstruction with SDF, dynamic scenes modelling. Secondly, our InfoReg solves the learning collapse in discrete gauge transformation, and can be potentially applied to the field of discrete representation learning (e.g., VQ-VAE (Van Den Oord et al., 2017)) which also suffers from codebook collapse. Thirdly, this work mainly explores the pure pre-defined or learned mapping, and how to combine pre-defined mapping and learned mapping to yield better performance is also of great potential. Please refer to Appendix A.6 for more discussion of limitations.

A.8 More Comparison & Study

Comparison with SOTA Methods. To demonstrate the power of the learned gauge transformation, we compare our top-256 gauge (best performance but also highest computation cost) with SOTA methods including EG3D (initial 64×64 plane with 32 dimension which is gradually upsampled to 192×192), TensoRF (initial 64×64 planes with 27 dimension which is gradually upsampled to 192×192) and Instant-NGP (8 layers with 2^{14} to 2^21 entries). As shown in Table 4, the learned top-256 gauge outperforms all SOTA methods with a much smaller model size, which demonstrates the potential of the learned gauge transformation for high-quality rendering.

Gaussian Distribution as Prior Distribution. Except for uniform distribution, other distributions like Gaussian distribution can also serve as the prior distribution for effective regularization as shown in Fig. 9. Compared with the rendering performance (30.74 as reported in Table 1) of uniform

Models	Model Details		Synthetic-NeRF		Synthetic-NSVF	
	Steps↓	Size(MB)↓	PSNR↑	SSIM↑	PSNR↑	SSIM↑
EG3D(Chan et al., 2022)	150k	16.70	31.63	0.950	34.52	0.972
TensoRF(Chen et al., 2022)	150k	11.80	32.94	0.959	35.41	0.976
Instant-NGP(Müller et al., 2022)	150k	21.30	33.10	0.963	35.65	0.978
Learning Top-256 Gauge	150k	**3.360**	**33.22**	**0.965**	**36.58**	**0.981**

Table 4: Comparison with SOTA methods on the Lego and Chair scenes.

distribution prior, Gaussian distribution prior achieves slightly inferior result (30.51) on Synthetic-NeRF but also avoids learning collapse well.

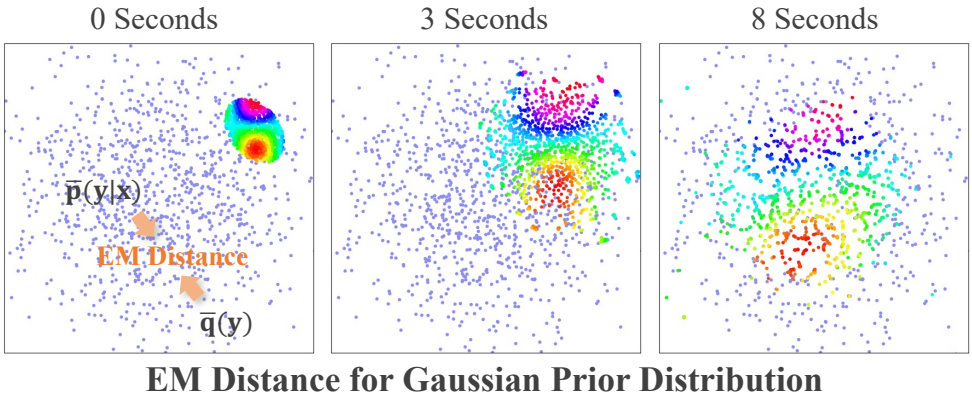


Figure 9: The gauge transformation (3D space to 2D plane) with Gaussian distribution as the prior distribution $\bar{q}(y)$ for regularization. Notably, it only takes 8 seconds to complete the gauge transformation & regularization for the full training process.

A.9 More Qualitative Results

Fig. 10 compares the performance of InfoReg with Euclidean distance and cosine distance, the Euclidean distance is prone to preserve better texture structure compared with Cosine distance although they both cover the 2D space and achieve similar rendering performance.

Fig. 11 shows additional samples of leaning continuous gauge transformation with different regularizations.

Except for novel view synthesis, we also apply our neural gauge fields to the downstream task of texture editing as shown in Fig. 12. Specifically, we learn the mapping from 3D space (points on object surface) to sphere surface which is visualized by cubemap for texture editing.

Figure 10: The results (3D space to 2D plane) of different distance metrics (Euclidean distance or Cosine distance) used in network T in the proposed InfoReg. Euclidean distance leads to better preservation of local texture structure compared with Cosine distance.

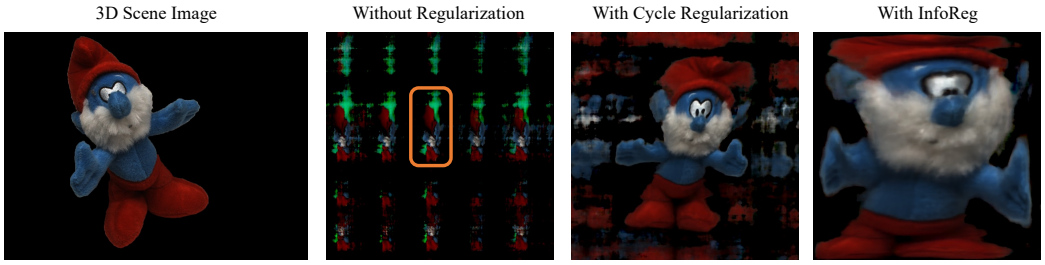


Figure 11: The results (3D space to 2D plane) of different learning regularizations. Our InfoReg leads to the best mapping results by utilizing the full plane.

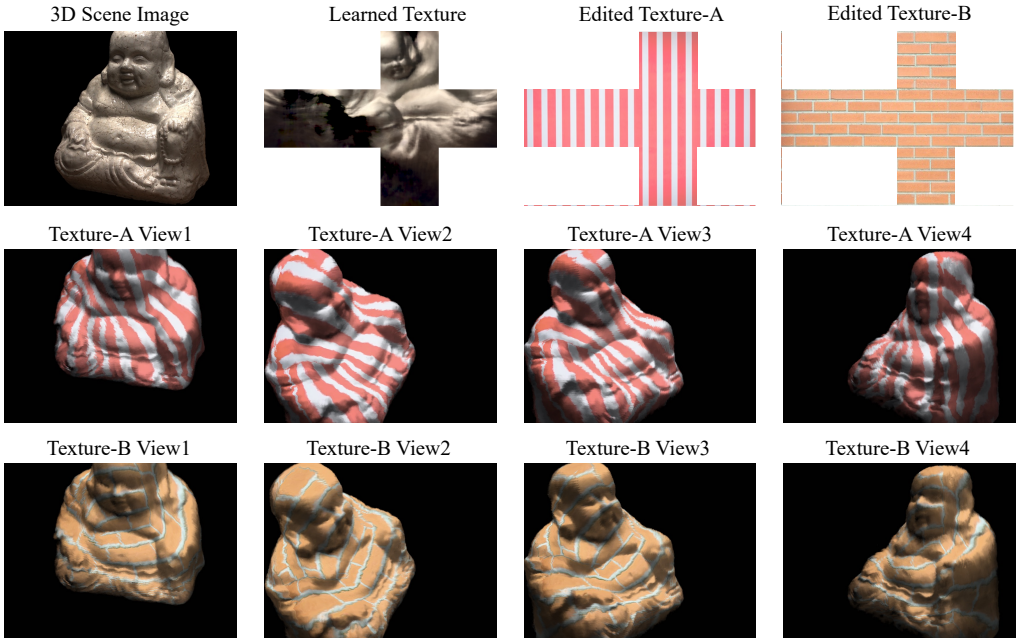


Figure 12: The application of neural gauge fields for texture editing. A gauge transformation is learned from 3D space to sphere surface (visualized by cubemap). By editing the texture of cubemap, the model is able to render multi-view images of the corresponding texture.