Image and Vision Computing 156 (2025) 105465

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Feature Field Fusion for few-shot novel view synthesis

Junting Li [a], Yanghong Zhou [a,b], Jintu Fan [a,b], Dahua Shou [a,b], Sa Xu [c], P.Y. Mok [a,b,d,*]

[a] School of Fashion and Textiles, The Hong Kong Polytechnic University, Hong Kong
[b] Research Centre of Textiles for Future Fashion, The Hong Kong Polytechnic University, Hong Kong
[c] Lunaler Healthy Technology Co., Ltd., Guangzhou, 511431, China
[d] Division of Integrated Systems and Design, The Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Keywords:
Neural radiance fields
Novel view synthesis
Few-shot learning
Feature field fusion
Sparse setting
Regularizations

ABSTRACT

Reconstructing neural radiance fields from limited or sparse views has given very promising potential for this field of research. Previous methods usually constrain the reconstruction process with additional priors, e.g. semantic-based or patch-based regularization. Nevertheless, such regularization is given to the synthesis of unseen views, which may not effectively assist the field of learning, in particular when the training views are sparse. Instead, we propose a feature Field Fusion (FFusion) NeRF in this paper that can learn structure and more details from features extracted from pre-trained neural networks for the sparse training views, and use as extra guide for the training of the RGB field. With such extra feature guides, FFusion predicts more accurate color and density when synthesizing novel views. Experimental results have shown that FFusion can effectively improve the quality of the synthesized novel views with only limited or sparse inputs.

1. Introduction

Neural Radiance Field (NeRF) [1–3] and 3D Gaussian Splatting (3DGS) [4], with an impressive ability of novel-view synthesis, have attracted widespread attention in the domain of pattern recognition [5,6] as well as in the vision and graphics community [7–10] in recent years. For NeRF, it is typically trained on *a large number of views*, requiring tens of input images with varied camera poses for a specific 3D scene, which has imposed a crucial restriction on the applications of NeRF. This is because in real-life scenarios, it is very difficult, if not impractical, to collect such a large number of input views of a 3D scene for training purpose.

Learning a radiance field from *limited or a few shots* (e.g., sparse inputs with only three input views), therefore, has attracted a great deal of research attention. A key approach is to design novel regularization for NeRFs, such as DietNeRF [12], RegNeRF [13], FreeNeRF [14], SimpleNeRF [15], SparseNeRF [11], ConsistNeRF [16], ZeroRF [17], and mi-MLP [18]. These methods exploit *additional* information, such as depth, geometry, or semantics, to constrain the optimization process. Nevertheless, such extra information is often applied as a 'weak constraint' for training, meaning that even though it guides the learning process, it is not explicitly incorporated into the model for the view synthesis. Existing methods suffer from this limitation of lacking explicit control in the novel-view synthesis; it is thus still challenging to make accurate predictions of each sample point's properties (e.g., color, depth, or geometry) along the ray, especially in sparse scenarios.

Compared to RGB images, feature maps capture more high-level and abstract information, beyond simple color, and preserve spatial correspondence well. Moreover, features are more robust to variations in lighting, viewpoint, and background, because they focus on meaningful patterns and structures rather than raw pixel values. In this paper, we propose a novel approach called feature field fusion (FFusion) for few-shot novel-view synthesis. As shown in Fig. 2, FFusion is a multi-task learning network that simultaneously learns both a feature map field and an RGB field in order to capture implicit 3D representations of the scene for the feature maps and RGB colors, respectively. We use features both as regularization during training and as prior information incorporated into the model functioning as explicit condition guidance for view synthesis. Learning feature representations in addition to colors can help NeRF capture the main content of the image more effectively and reconstruct finer details. Specifically, we extract feature maps from a pre-trained CNN model and train a NeRF, called a *neural feature radiance field*, which models a 3D scene with features. To visualize the additional information a learned feature field can provide, Fig. 1(b) shows an example of the reconstructed image of rendered feature maps from the feature field using a 2D CNN decoder. It can be seen that the image reconstructed from rendered feature maps accurately captures the shape of the flower stamen even though the image is blurry. This demonstrates that while the feature field may not fully capture all the fine details and textures, it still conveys essential

* Corresponding author.
E-mail addresses: tracy.mok@ust.hk, tracy.mok@polyu.edu.hk (P.Y. Mok).

https://doi.org/10.1016/j.imavis.2025.105465
Received 5 September 2024; Received in revised form 31 December 2024; Accepted 12 February 2025
Available online 22 February 2025

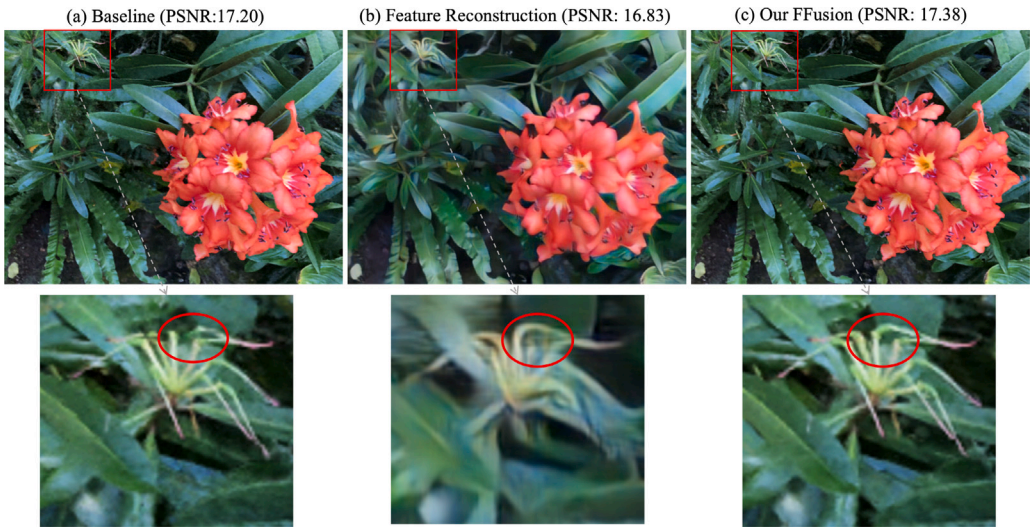

Fig. 1. Novel view synthesized by (a) baseline of SparseNeRF [11], (b) feature field, and (c) our FFusion. By combining knowledge from the feature field, our FFusion reconstructs a better shape of the flower stamen and captures finer details, maintaining high-quality RGB field learning.

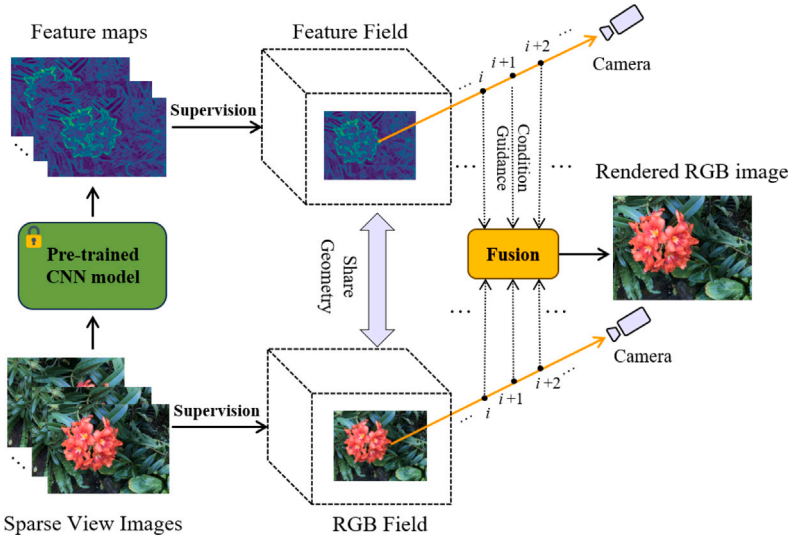

Fig. 2. Concept illustration of FFusion NeRF: We learn a feature field and a RGB radiance field simultaneously and the abstract feature is employed to help the learning of RGB field. The feature radiance field and RGB radiance field share the same geometry.

structure of the image. Fig. 2 shows that the feature field and the RGB field share the same geometry. This is because that, in our formulation, the feature maps correspond to the RGB images at each pixel, the feature field therefore influences the learning of each sample point's properties, including color, depth, and semantics. It serves as a strong constraint for novel-view synthesis at the pixel level. By incorporating the knowledge from the feature field, our FFusion gives more accurate color predictions, as demonstrated in Fig. 1(c) in comparison to the baseline in Fig. 1(a).

The main contributions of the present work are as follows:

- We propose a novel approach that exploits spatial feature maps to enhance the scene representation of NeRF for sparse view inputs. To the best of our knowledge, this is the first work to employ spatial feature maps in NeRF.
- A feature field fusion module is proposed to incorporate the learned feature field to the color field so as to fully utilizing the knowledge contained in the feature field as a condition guidance for novel view synthesis.
- Extensive experiments have shown the effectiveness of our proposed method on LLFF and DTU datasets. Moreover, our method is model-agnostic that integrates seamlessly with various state-of-the-art approaches, achieving superior performance on both datasets.

2. Related work

Novel-view Synthesis with Sparse Inputs. The original NeRF is designed for scenarios with very dense input views, which do not meet the needs of real-world applications because such data capturing process is time consuming and real camera poses are required [19, 20]. Our work focus on 3D scene synthesis from sparse input views (e.g., as few as 3). The existing NeRFs and 3DGS [10] for sparse input views can be broadly classified into transfer learning-based methods and regularization-based methods. The transfer learning-based methods [21–24] attempt to pre-train a NeRF on large-scale curated multi-view datasets and fine-tune the NeRF on the target scene. pixelSplat [25] trains a model to predict feed-forward 3D Gaussians with 2 inputs. MVSplat [26] improves the quality and feed-forward speed of pixelSplat with a cost volume for valuable geometry cues. However, they rely on large-scale datasets and their effectiveness is constrained

by the domain gap between the pre-trained model and the target scene, which can lead to sub-optimal results if the domains differ significantly. In contrast, regularization-based methods [2,11–14] do not depend on large-scale pre-trained models. Instead, they improve the model's ability to generalize to new or diverse scenes by incorporating prior knowledge or constraints, thereby mitigating the effects of domain mismatch.

Regularization-based radiance field. Diet-NeRF [12] regularizes the field by comparing the semantic embedding of unseen viewpoints to that of known viewpoints. RegNeRF [13] regularizes unobserved views using patch-based depth and color constraints. Instead of constraining the field from unobserved views, other methods explore to improve the results from the limited training views. SparseNeRF [11] distills depth information predicted from a prior model to constrain the geometry. Li et al. [27] improved sparse view synthesis with both geometry and appearance regularization. FSGS [28] handles the sparse initialization of 3DGS with proximity-guided gaussian unpooling and online depth augmentation. GeoRGS [29] explores seed-based geometric regularization with depth similarity and consistency for 3DGS in sparse input. GaussianObject [30] trains a Gaussian Repair Model, which is diffusion model, to help the Structured 3D gaussians with a Distance-Aware Sampling. FreeNeRF [14] explores the frequency of position encoding and trains the field in a coarse-to-fine process. ReconFusion [31] exploits a diffusion prior for novel view synthesis. Although regularization from unobserved views is useful for sparse input, existing methods do not fully explore using prior information as guidance for training views. Zhi et al. [32] used semantic labels as additional supervision to learn a field for segmentation. DFF [33] uses semantic features as supervision to learn a semantic field that can decompose different parts of a scene. LeRF [34] distinguishes different parts of a scene through a language model. Furthermore, Latent-NeRF [35] generates 3D scenes represented by latent 3D representations. In these studies, a semantic field can be learned to form 2D supervision. Nevertheless, the detailed information of each 3D point is not known yet, in particular in field learning under sparse inputs.

Feature field. In NeRF, the RGB field is learned to predict the color values for each point in the 3D scene, enabling the generation of photo-realistic images by combining these color predictions with density information. Differently, the feature field is learned to predict feature values and generate feature maps. With feature fields, a variety of applications can be realized, including image editing [36], style transfer [33,37], keypoint transfer [38], segmentation transfer [6,33,39], and label transfer [32]. Typically, these methods involve learning both the feature field and the RGB field simultaneously. The learned feature field can then be utilized directly for various tasks or applied through knowledge distillation. However, the feature field has not been utilized to assist NeRF with sparse input views.

3. Method

In this paper, we propose a novel framework for neural radiance fields in a sparse setting. Specifically, our method is based on the original NeRF (described in Section 3.1) so that its variants can also serve as backbones in our framework. For sparse views, we propose FFusion NeRF framework (Section 3.2) that not only takes advantage of feature supervision as regularization but also functions as additional information incorporated for novel-view synthesis.

3.1. Preliminary

Different from classical approaches that rely on explicit geometry or point clouds, NeRF represents a 3D scene as a continuous 3D function, called a radiance field. It is a neural network consisting of multiple perception layers (MPLs). It maps any given 3D point $\mathbf{p} = (x, y, z)$ and a view direction $\mathbf{d}$ to the volume density σ representing scene opacity and view-dependent radiance (RGB color) capturing its appearance and color. More specifically, the $\mathbf{p}$ and $\mathbf{d}$ are encoded as $\gamma(\mathbf{p})$ and $\gamma(\mathbf{d})$ by a positional encoding γ, and the $\gamma(\mathbf{p})$ is then fed into a backbone block to learn the backbone feature $f_b = MLP_b(\gamma_L(\mathbf{p}))$, which is fed into an MPL to output the density $\sigma = MLP_\sigma(f_b)$. After that, the feature f_b and the direction $\mathbf{d}$ are both fed into a color prediction block MLP_c to predict the corresponding color c_i conditioned on the view direction $\mathbf{d}$, which is given as

$$c_i = MLP_c(f_b, \gamma(\mathbf{d})) \tag{1}$$

To compute the color radiance at a pixel (u, v) in an image, NeRF typically casts a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ through this pixel where t varies along the ray from a near plane t_n to a far plane t_f, and then evenly partition $[t_n, t_f]$ into M shading points $\{\mathbf{t}_j | i = 1, \ldots, M\}$ along the ray. The expected pixel color radiance is computed by volume render:

$$\hat{C} = \sum_{i=1}^{N} w_i c_i \tag{2}$$

$$w_i = T_i(1 - exp(-\sigma_i \delta_i)), \tag{3}$$

where $\delta_i = t_i - t_{i-1}$, and $T_i = exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$ is the transmittance of the ray which presents the probability that information from the 1st sampled point can pass through to the $i - 1^{th}$ sampled point. For model training, Mean Squared Error (MSE) is exploited to minimize the distance between expected colors and ground truth colors:

$$L_c = \sum_{\mathbf{r} \in R} \|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2 \tag{4}$$

where R represents a batch of sampling rays and $C(\mathbf{r})$ is the ground-truth pixel color.

3.2. FFusion NeRF

3.2.1. Feature field

Based on the original NeRF, a feature radiance field, denoted as MLP_f, is proposed to predict the feature representation of images, which captures more high-level features other than RGB colors, such as edge, shape, and texture. To avoid the problem of overfitting and save computation resources, the feature radiance field and the color radiance field share the backbone, whereas the backbone feature f_b adds two MLPs (denoted as MLP_f in Fig. 3) to predict the feature radiance f as follows:

$$f = MLP_f(f_b) \tag{5}$$

Similar to the volume rendering of pixel colors, the feature $\hat{F}$ is rendered using Eqs. (2) and (3) by replacing the c with f.

To train the feature field, we extract the corresponding feature maps F of input images from a VGG network [40] pretrained on the ImageNet dataset [41] for classification. More specifically, we compare in later experiment section that the use of different layers from the VGG network, including ReLU1-1, ReLU2-1 and ReLU3-1, as the ground truths to supervise feature field training, which correspond to the 3rd, 10th, and 17th layers of the VGG network. The model parameters are optimized by minimizing the distance between the rendered feature representations $\hat{F}$ and the estimated features F with L_2 loss:

$$L_f = \sum_{\mathbf{r} \in R} \|\hat{F}(\mathbf{r}) - F(\mathbf{r})\|_2^2 \tag{6}$$

where R represents a batch of sampling rays and $F(r)$ is the ground-truth pixel representation, namely an extracted feature map.

3.2.2. Feature field fusion

To leverage feature radiance knowledge for RGB field training, a feature field fusion module MLP_{fusion} is designed. As shown in Fig. 3, the feature radiance f and the backbone feature f_b are processed by two linear layers LN_t and LN_b, respectively, and projected to a space that is compatible with that of the view direction $\mathbf{d}$. The channel numbers of LN_t and LN_b are set to match the dimensions of

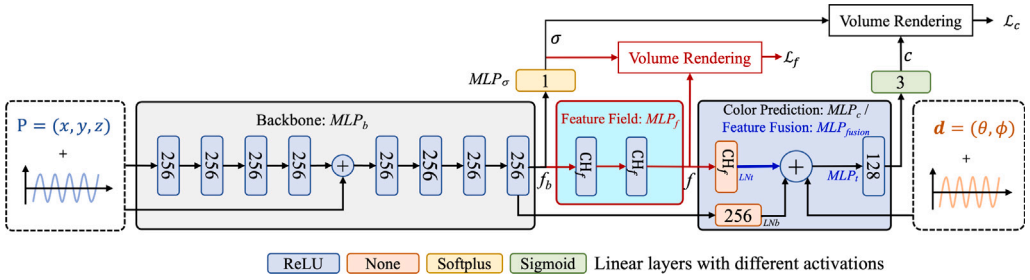

Fig. 3. Network structure of our FFusion, where the proposed new structure, comparing to NeRF, are highlighted in red and blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

their corresponding inputs, i.e., [CH_f] the channel number of f and 256, respectively. Subsequently, the projected feature field $LN_t(f)$, the backbone feature $LN_b(f_b)$, and the view direction **d** are concatenated together. This concatenated vector is then processed by an MLP layer MLP_t to generate a fused feature. The channel number of MLP_t is set to 128 to compress the representation for color prediction. Finally, the fusion feature is fed into another MLP layer MLP_c to produce the output color radiance c. The channel number of MLP_c is set to 3 in order to match the dimension of the RGB image.

$$c_i = MLP_c(MLP_t(LN_t(f) \oplus LN_b(f_b) \oplus \gamma(\mathbf{d}))) \tag{7}$$

After that, the expected color $\hat{C}$ is computed through volume rendering using Eqs. (2) and (3). It should be noted that during volume rendering, the same density values are used in the alpha blending process of both the feature and the color fields, thereby ensuring that the feature and RGB fields share the same geometry. During training, the expected color $\hat{C}$ is optimized by minimizing the loss function defined in Eq. (4). With reference to Fig. 3, the difference between the network structure of NeRF and our FFusion model is the feature field (FF) MLP_f, and the feature fusion module MLP_{fusion} has an additional feature input f compared to the color prediction block MLP_c of the original NeRF model. Our framework is adaptable and can work with different backbones, which we will further discuss later in Section 4.4.

3.2.3. Loss function

The overall model optimization is guided by the loss function:

$$L = \lambda L_f + L_c \tag{8}$$

where λ balances the learning between features L_f and colors L_c.

4. Experiments

4.1. Datasets, metrics and implementation details

Datasets. We conducted experiments on the LLFF [1] dataset and the DTU [42] dataset to evaluate the effectiveness of our method. **LLFF** dataset consists of 8 forward-facing scenes. Following [1], we kept every 8th image as the hold-out test set and evenly selected the training views from the remaining images to build the training set. **DTU** dataset is an object-level dataset containing various objects in multiple views in a controlled indoor setting. Following previous work [1,13,22], 15 scenes with 49 frames in each scene were selected for training.

Evaluation Metrics. To evaluate view synthesis performance, we used peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) [43] and learning perceptual image patch similarity (LPIPS) [44] as evaluation metrics. For the DTU dataset, all three metrics were computed within the mask of the object in the scene. For a fair comparison, we evaluated these metrics on the test set and calculated the mean value among all scenes in the dataset.

Table 1
Ablation study evaluating the impact of the proposed modules of our FFusion network for sparse setting of 3 input views on the LLFF dataset.

Model	PSNR↑	SSIM↑	LPIPS↓
Vanilla – NeRF [1]	16.81	0.539	0.447
Vanilla + LPIPS	14.17	0.450	0.510
Vanilla + SSIM	14.08	0.447	0.543
Vanilla + FF	16.12	0.506	0.481
Vanilla + Fusion	17.32	0.520	0.504
Ours (FFusion– NeRF [1])	**17.70**	**0.557**	**0.363**

Implementation Details. Our experiment was based on the open source code of RegNeRF. We used the Adam optimizer in training the models, with the exponential learning rate decreasing from 5×10^{-4} to 5×10^{-5}. We set the batch size to 1024 and trained each scene on one 3090 Ti GPU. For the LLFF dataset, we trained 69k/13k/20k iterations for 3/6/9 input views at a resolution of 504 × 378. For the DTU dataset, we trained 43k/87k/131k for 3/6/9 input views at a resolution 400 × 300. For the LLFF and DTU datasets, we used the Relu3-1 and Relu1-1 layer, respectively, of a VGG encoder [40] as the supervision for training our model, namely F in Eq. (6). We loaded the weights from Adain [45] for the VGG model. For the DTU dataset, we masked the background region following previous methods when calculating metrics to ensure a fair comparison with other methods.

4.2. Ablation study

In order to examine the effectiveness of the proposed FFusion in a sparse setting, we conducted an ablation study on the LLFF dataset. To avoid impacts brought by other methods such as the use of depth information or semantics, we chose the vanilla model of NeRF [1] as our baseline for the ablation study to analyze the effects of individual modules of our proposed framework, including the feature field (FF) and the fusion module.

The qualitative and quantitative results are shown in Fig. 4 and Table 1, respectively. The qualitative results in Fig. 4 show that the reconstructed feature field (the depth map in 'vanilla (feature)' column, which is obtained by volume rendering using Eqs. (2) and (3) and replacing color c with feature f) can show clear geometry of the scene. Such feature field is even more accurate than the rgb field (i.e., the depth map in 'vanilla (rgb)' column) in a sparse setting, because feature field remains the same regardless of view directions. Nevertheless, as shown in the column marked with 'vanilla +FF', by simply introducing a feature field cannot result in better quality synthesized views. On the other hand, the result in the column marked with 'vanilla + Fusion' shows that when additional features are input to the color prediction block as a condition guidance, it can contribute to better output, even

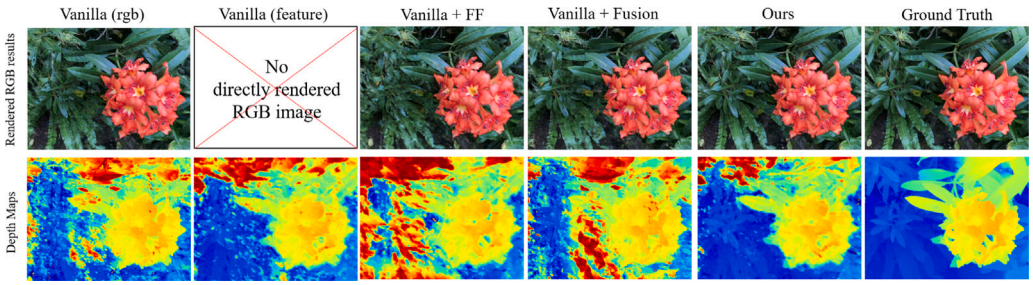

Fig. 4. Ablation study comparing synthesized views and corresponding depth maps utilizing vanilla NeRF model [1] in a very sparse setting of 3 input views based on the flower scene of the LLFF dataset.

though the feature loss L_f is obtained together with color prediction loss, i.e., λ=0 in Eq. (8). Lastly, the column marked with 'Ours' shows the result of our full model of FFusion, in which the feature is tactically used as a condition to guide the novel view synthesis, resulting in better quality of output in a sparse setting.

Moreover, to show that our method improves pixel-level consistency rather than visual perceptual consistency, we compared our FFusion with the vanilla NeRF with LPIPS and SSIM losses, in which SSIM loss is often used in 3DGS related studies. For 3DGS, SSIM loss is computed based on the entire image in every rendering step. In contrast, NeRF samples a random subset of pixels, typically around 1024 pixels, from one or more images during each training iteration. These pixels are not necessarily from neighboring patches. To apply LPIPS or SSIM loss to NeRF, we randomly sampled 4096 pixels from a patch, as LPIPS requires a patch of at least 64 × 64 pixels for computation. During training, we used the same loss formulation as 3DGS, i.e., $0.8L_1+0.2L_x$, where L_x can be either LPIPS or (1-SSIM). However, even with a batch size four times larger than ours, the results of vanilla with LPIPS or SSIM loss are significantly worse than that of the original (vanilla) NeRF. This could be due to the fact that, although LPIPS or SSIM introduces local regularization, the patch-based computation does not capture good global information of the scene.

Table 1 presents the quantitative results of the ablation study in a 3-view setting for a baseline of NeRF, and with incorporation of only feature field (FF), with feature fusion module (without loss guidance) and that of our full model (with loss guidance). As shown, the performance slightly degrades when only the feature field is added to the NeRF. This degradation occurs because both the feature field and the RGB field share the same backbone, leading to difficulties in accurate geometry learning due to noise in the feature maps. The fusion of feature can integrate the knowledge from the feature, resulting in better color prediction, and our full model brings significant improvements to all three metrics, namely, PSNR increases from 16.81 to 17.70, SSIM rises from 0.539 to 0.557, and LPIPS decreases from 0.447 to 0.363.

4.3. Comparison with SOAT methods

We compared our FFusion with 9 state-of-the-art methods, including transfer learning-based and regularization-based methods, on the LLFF and DTU datasets to evaluate the effectiveness of our FFusion method. For a fair comparison, we do not compare to those diffusion-based methods [31,46], because they benefit from large pre-trained diffusion models and can generate high-fidelity images, yet they have completely different structures and principles.

Comparison based on the DTU dataset. For the DTU dataset, our method used FreeNeRF [14] as a baseline and the quantitative comparison results with these SOTA methods are given in Table 2. As shown, our FFusion outperforms all listed state-of-the-art methods in terms of PSNR with 3, 6, and 9 input views. Specifically, compared to the existing regularization methods, our FFusion, based on FreeNeRF, achieves a PSNR improvement of 20.83 versus 19.92 with 3 input views, 23.81 versus 23.25 with 6 input views, and 26.26 versus 25.38 with 9 input views. The training of SparseNeRF was not stable on the DTU dataset and it failed to generate results for some scenes with 6 or 9 input views (Table 2). The qualitative comparison results with five selected state-of-the-art methods based on the DTU dataset are given in Fig. 5. The qualitative comparison shows that our FFusion not only achieves better depth estimation but also reconstructs the scene's structure and shape more accurately.

Comparison based on the LLFF dataset The quantitative and qualitative comparison results with state-of-the-art methods based on the LLFF dataset are shown in Tables 3 and 6, respectively. As shown in Table 3, our FFusion based on SparseNeRF outperforms all the listed methods on the metric of SSIM. Specifically, our FFusion achieves SSIM scores of 0.735, 0.829, and 0.860 with 3, 6, and 9 views, respectively. Additionally, in the 3-view setting, our FFusion outperforms all other methods across all metrics. From Fig. 6, it can be seen that all state-of-the-art methods fail to accurately reconstruct the details of the breastbones, whereas our FFusion successfully captures these details.

4.4. Discussion

Model-Agnostic Study. We demonstrate the model-agnostic characteristics of our method using the LLFF dataset. Table 4 presents the comparison results across various baselines. It is evident that our FFusion method consistently improves performance on almost all metrics compared to each baseline. Specifically, compared to DietNeRF [12], which regularizes the consistency of semantic features, our FFusion achieves improvements of 17.92 versus 17.20 in PSNR, 0.564 versus 0.522 in SSIM, and 0.374 versus 0.395 in LPIPS. When FreeNeRF [14] is used as the baseline, our FFusion achieves higher PSNR and SSIM but underperforms in LPIPS. This is because FreeNeRF's longer frequency curriculum and our feature field lead to smoother scenes, which may adversely affect the LPIPS score, despite the more competitive PSNR score.

The effect of feature layers. In our framework, we extract features from a pretrained network to supervise feature field training, i.e., F in Eq. (6). Since different layers of a pre-trained network contain different levels of information about the scene, this may influence the reconstruction process to different degrees. The quantitative results corresponding to different feature layers in Table 5 confirm the selection of Relu3-1 on NeRF baseline for the LLFF dataset and Relu1-1 on FreeNeRF [14] baseline for the DTU dataset. For the LLFF dataset, the use of Relu3-1 achieves the highest performance because the whole image is reconstructed whereas high-level features give more global information. For object-level DTU dataset, the introduction of feature field fusion, regardless of the layer of feature being adopted, does contribute to performance improvement when objects are being masked.

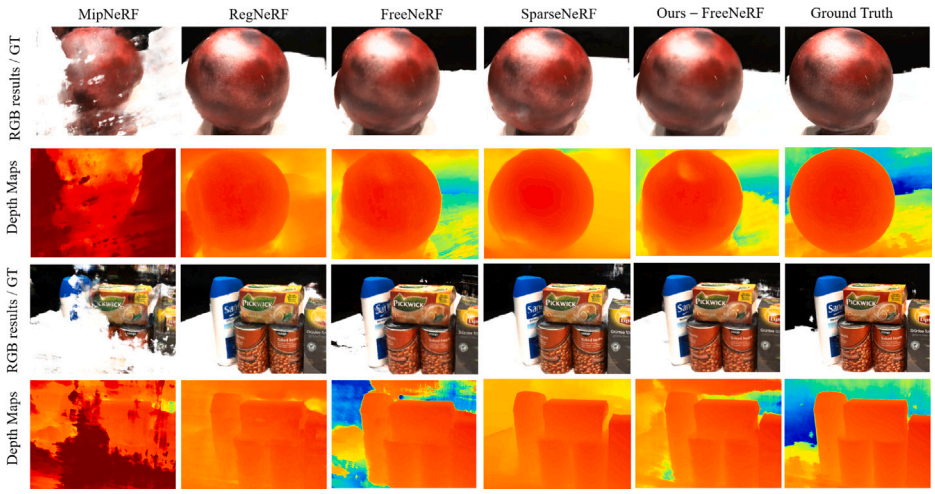

Fig. 5. Qualitative comparison results: with extra feature guides, our method can reduce floaters and reconstruct better quality objects.

Table 2
Quantitative Comparison with SOAT methods on the DTU dataset. *np* means not provided in the original article.

Method	3 views			6 views			9 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRF [21]	15.32	0.671	0.304	17.54	0.730	0.250	18.35	0.752	0.232
SRF*fine-tuned* [21]	15.68	0.698	0.281	18.87	0.757	0.225	20.75	0.785	0.205
PixelNeRF [22]	16.82	0.695	0.270	19.11	0.745	0.232	20.40	0.768	0.220
PixelNeRF *fine-tuned* [22]	18.95	0.710	0.269	20.56	0.753	0.223	21.83	0.781	0.203
MVSNeRF [23]	18.63	0.769	0.197	20.70	0.823	0.156	22.40	0.853	0.135
MVSNeRF *fine-tuned* [23]	18.54	0.769	0.197	20.49	0.822	0.155	22.22	0.853	0.135
Mip-NeRF [2]	8.68	0.571	0.353	16.54	0.741	0.198	23.58	0.879	0.135
DietNeRF [12]	11.85	0.633	0.314	20.63	0.778	0.201	23.83	0.823	0.173
RegNeRF [13]	18.89	0.745	**0.190**	23.10	0.760	0.206	24.86	0.820	0.161
SimpleNeRF [15]	16.25	0.751	0.249	20.60	0.828	0.190	22.75	0.856	0.176
FreeNeRF [14]	19.92	0.787	*np*	23.25	0.844	*np*	25.38	0.888	*np*
SparseNeRF [11]	19.55	0.769	0.201	error	error	error	error	error	error
Ours–FreeNeRF	**20.83**	**0.804**	0.201	**23.81**	**0.861**	**0.153**	**26.26**	**0.895**	**0.126**

Table 3
Quantitative Comparison with SOAT methods on the LLFF dataset.

Method	3 views			6 views			9 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRF [21]	12.34	0.250	0.591	13.10	0.293	0.594	13.00	0.297	0.605
SRF *fine-tuned* [21]	17.07	0.436	0.529	17.39	0.438	0.521	17.39	0.465	0.503
PixelNeRF [22]	7.93	0.272	0.682	8.74	0.280	0.676	8.61	0.274	0.665
PixelNeRF *fine-tuned* [22]	16.17	0.438	0.512	17.03	0.473	0.477	18.92	0.535	0.430
MVSNeRF [23]	17.25	0.557	0.356	19.79	0.656	0.269	20.47	0.689	0.242
MVSNeRF *fine-tuned* [23]	17.88	0.584	0.327	19.99	0.660	0.264	20.47	0.695	0.244
Mip-NeRF [2]	14.62	0.351	0.495	20.87	0.692	0.255	24.26	0.805	0.172
DietNeRF [12]	14.94	0.370	0.496	21.75	0.717	0.248	24.28	0.801	0.183
RegNeRF [13]	19.08	0.587	0.336	23.10	0.760	0.206	24.86	0.820	0.161
SimpleNeRF [15]	19.24	0.623	0.375	23.05	0.737	0.296	23.98	0.762	0.286
FreeNeRF [14]	19.63	0.612	0.302	**23.73**	0.779	**0.195**	**25.13**	0.827	**0.160**
SparseNeRF [11](our imp)	19.84	0.620	0.325	23.10	0.749	0.233	24.37	0.795	0.198
Ours–SparseNeRF	**20.09**	**0.735**	**0.296**	23.48	**0.829**	0.218	24.63	**0.860**	0.189

The FFusion with Relu1-1 achieves the largest improvement, because lower-level features provide more detailed information, such as edges and texture.

The effect of loss weight λ. We also examine hyperparameter setting in our method, namely the balancing weight λ in Eq. (8). The precision of geometry of the reconstructed field can be affected by the balancing weight λ between the feature and the rgb color. As evidence in Table 6, a balancing weight of $\lambda = 0.01$ is suggested for a FreeNeRF [14] baseline to balance between feature loss and color loss for better quality of outputs.

The effect of feature extractors. We compare different feature extractors, including VGG16 [40], ResNet18 [47] and Lseg [48], in our framework. Specifically, we extract feature maps from Relu1-1 of VGG16 network, and from Conv1 of ResNet18 network. For Lseg [48], we compute the word–pixel correlation tensor from the Lseg(ViT-L/16) as the feature maps. The comparison results are shown in Table 7,

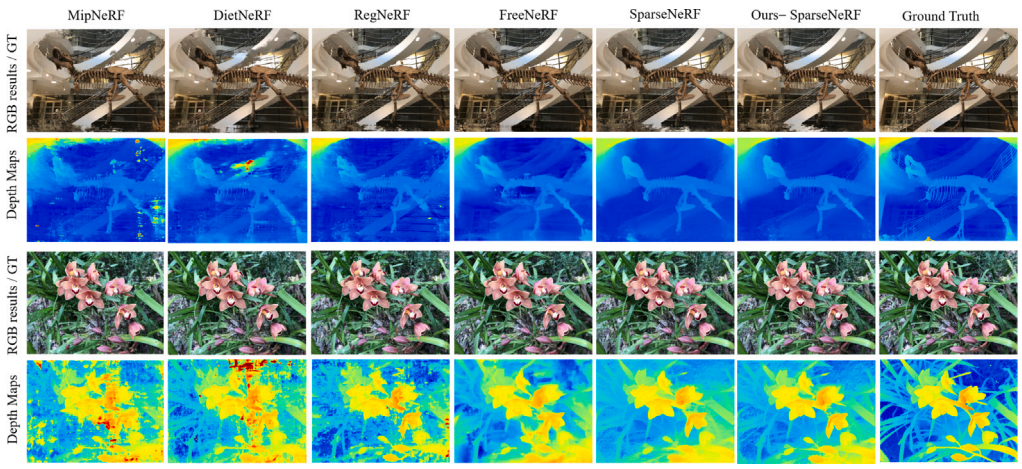

Fig. 6. The qualitative results of our method based on SparseNeRF baseline in comparison to other SOTA methods.

Table 4
Comparison results of our methods with different baselines.

Method	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [2]	16.54	0.561	0.448
FFusion–Mip-NeRF	**17.21**	**0.616**	**0.396**
DietNeRF [12]	17.20	0.522	0.395
FFusion–DietNeRF	**17.92**	**0.564**	**0.374**
RegNeRF [13]	19.14	0.681	0.353
FFusion–RegNeRF	**19.64**	**0.712**	**0.314**
FreeNeRF [14]	19.59	0.618	**0.302**
FFusion–FreeNeRF	**19.97**	**0.620**	0.320
SparseNeRF [11]	19.84	0.620	0.325
FFusion–SparseNeRF	**20.09**	**0.735**	**0.296**

Table 5
Performance and efficiency comparisons for different layers of feature maps on both datasets. For a fair efficiency comparison, the memory consumption and runtime are calculated using the same chunk size (1024 × 16), netchunk size (1024 × 128) and image size (1008 × 756) on a GeForce RTX 3090 GPU.

Layers	LLFF dataset			DTU dataset			Memory	Time
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	(MB)	(s)
Baseline	16.81	0.539	0.447	19.49	0.761	0.223	2602	14.7
Relu1-1	17.45	0.551	0.443	20.82	0.803	0.201	2766	19.9
Relu2-1	17.61	0.570	0.427	20.60	0.805	0.201	2798	20.3
Relu3-1	17.70	0.557	0.363	19.77	0.786	0.216	2992	21.3

Table 6
Comparison of different loss weights on DTU dataset.

λ	PSNR↑	SSIM↑	LPIPS↓
Baseline [14]	19.49	0.761	0.223
0.1	20.35	0.794	0.210
0.01	20.83	0.803	0.201
0.001	20.39	0.800	0.201

where our FFusion brings performance improvements with all three feature extractors. Comparatively, our FFusion with VGG feature maps performs better than those with ResNet18 and Lseg feature maps. This is because VGG feature maps provide more color and texture information, which are good for image reconstruction. Compared to ResNet, the pre-trained VGG network has a better ability to capture the visual style of an image [49]. For the same reason, Lseg [48] mainly focuses on semantic segmentation, and the learned features contain more high-level semantic information, which are not conducive to supplementing and restoring details. The VGG network pre-trained on ImageNet is, therefore, suggested for extracting feature maps for the supervision of the feature field in our framework.

Table 7
Comparison of different feature extractors for feature supervision on the LLFF dataset.

Feature	PSNR↑	SSIM↑	LPIPS↓
Baseline (Mip-NeRF [2])	16.54	0.561	0.448
VGG [40] (Relu1-1)	17.21	0.616	0.396
ResNet18 [47] (Conv1)	16.57	0.568	0.441
Lseg [48] (ViT-L/16)	16.67	0.576	0.434

Efficiency analysis. We compared the memory consumption and runtime of the baseline method with our FFusion, using different layers of feature maps. The comparison results are shown in Table 5. As shown, our FFusion has a small increase in memory consumption and runtime compared to the baseline. This is because the feature maps do not need to be rendered during the inference stage. Specifically, FFusion with ReLU1-1 increases memory consumption by 164 MB (6.3%) and runtime by 5.2 s. FFusion with ReLU2-1 and ReLU3-1 has higher memory consumption and takes a longer runtime than that with ReLU1-1, because the feature maps being used have larger channel numbers.

5. Conclusions and future work

In this paper, we propose a novel framework FFusion to synthesize novel views with sparse view inputs. To tackle the under-constrained few-shot NeRF problem, our proposed FFusion learns a feature field to help the learning of the RGB color field. The experimental results have also shown that our method significantly improves the synthesis performance in a sparse setting and is complementary to various methods.

Limitations and future work. Although our work has demonstrated that the training of a feature field can benefit the training of the target RGB field, it is challenging to balance the learning of both fields in a multi-task learning manner, especially when combined with other regularization losses, thereby leading to lower LPIPS scores. Fig. 7 shows an example failure case of our method on the DTU dataset. Even though our method improves the quality of synthesized novel views, our method will still fail if the testing view is very different

Fig. 7. Failure case on the DTU dataset. When the test view is very different from the training views, our method will still generate error result, suffering from the same limitation as that of SparseNeRF and FreeNeRF.

from the training view. As a future work, we will explore using the state-of-the-art loss weighting methods to address this issue. We will also explore integrating with diffusion models [30,31] to improve these views. Moreover, we will also study applying FFusion to 3D Gaussian splatting [4].

CRediT authorship contribution statement

Junting Li: Software, Methodology, Data curation, Conceptualization. **Yanghong Zhou:** Writing – original draft, Visualization, Methodology, Investigation. **Jintu Fan:** Supervision, Funding acquisition. **Dahua Shou:** Supervision. **Sa Xu:** Resources. **P.Y. Mok:** Writing – review & editing, Supervision, Investigation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jin-tu Fan reports financial support was provided by The Hong Kong Polytechnic University. Sa Xu reports financial support was provided by Lunaler Healthy Technology Co., Ltd. P.Y. Mok reports financial support was provided by General research grant of Hong Kong Special Administrative Region. P.Y. Mok reports financial support was provided by Research Centre of Textiles for Future Fashion. Yanghong Zhou reports financial support was provided by Research Centre of Textiles for Future Fashion. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work described in this paper is supported, in part, by The Hong Kong Polytechnic University (Grant Numbers Q-8883; CD95/P0049355; BDVH/P0051330; BBFL/P0052601), and by the General Research Grant of the Hong Kong Special Administrative Region (Grant Number 15602323).

Data availability

The data used is publicly available, and has been explained in the paper.

References

[1] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, R. Ng, NeRF: Representing scenes as neural radiance fields for view synthesis, in: ECCV, 2020.

[2] J.T. Barron, B. Mildenhall, D. Verbin, P.P. Srinivasan, P. Hedman, Mip-nerf 360: Unbounded anti-aliased neural radiance fields, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 5470–5479.

[3] J.T. Barron, B. Mildenhall, D. Verbin, P.P. Srinivasan, P. Hedman, Zip-nerf: Anti-aliased grid-based neural radiance fields, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 19697–19705.

[4] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, 3D gaussian splatting for real-time radiance field rendering, ACM Trans. Graph. 42 (4) (2023) 139–1.

[5] A. Mirzaei, T. Aumentado-Armstrong, K.G. Derpanis, J. Kelly, M.A. Brubaker, I. Gilitschenski, A. Levinshtein, SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 20669–20679.

[6] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, A. Kadambi, Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 21676–21685.

[7] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, et al., State of the art on neural rendering, in: Computer Graphics Forum, 39, (2) Wiley Online Library, 2020, pp. 701–727.

[8] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, et al., Advances in neural rendering, in: Computer Graphics Forum, 41, (2) Wiley Online Library, 2022, pp. 703–735.

[9] M. Gu, J. Li, Y. Wu, H. Luo, J. Zheng, X. Bai, 3D human avatar reconstruction with neural fields: A recent survey, Image Vis. Comput. (2024) 105341.

[10] J. Luo, T. Huang, W. Wang, W. Feng, A review of recent advances in 3D Gaussian Splatting for optimization and reconstruction, Image Vis. Comput. (2024) 105304.

[11] G. Wang, Z. Chen, C.C. Loy, Z. Liu, Sparsenerf: Distilling depth ranking for few-shot novel view synthesis, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 9065–9076.

[12] A. Jain, M. Tancik, P. Abbeel, Putting nerf on a diet: Semantically consistent few-shot view synthesis, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 5885–5894.

[13] M. Niemeyer, J.T. Barron, B. Mildenhall, M.S. Sajjadi, A. Geiger, N. 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, 2022, pp. 5480–5490.

[14] J. Yang, M. Pavone, Y. Wang, Freenerf: Improving few-shot neural rendering with free frequency regularization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 8254–8263.

[15] N. Somraj, A. Karanayil, R. Soundararajan, Simplenerf: Regularizing sparse input neural radiance fields with simpler solutions, in: SIGGRAPH Asia 2023 Conference Papers, 2023, pp. 1–11.

[16] S. Hu, K. Zhou, K. Li, L. Yu, L. Hong, T. Hu, Z. Li, G.H. Lee, Z. Liu, Consistentnerf: Enhancing neural radiance fields with 3d consistency for sparse view synthesis, 2023, arXiv preprint arXiv:2305.11031.

[17] R. Shi, X. Wei, C. Wang, H. Su, ZeroRF: Fast sparse view 360deg reconstruction with zero pretraining, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 21114–21124.

[18] H. Zhu, T. He, X. Li, B. Li, Z. Chen, Is vanilla MLP in neural radiance field enough for few-shot view synthesis? in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 20288–20298.

[19] Z. Fan, W. Cong, K. Wen, K. Wang, J. Zhang, X. Ding, D. Xu, B. Ivanovic, M. Pavone, G. Pavlakos, et al., Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds, 2024, arXiv preprint arXiv:2403.20309. 2 (3) 4.

[20] H. Li, Y. Gao, C. Wu, D. Zhang, Y. Dai, C. Zhao, H. Feng, E. Ding, J. Wang, J. Han, Ggrt: Towards pose-free generalizable 3d gaussian splatting in real-time, in: European Conference on Computer Vision, Springer, 2025, pp. 325–341.

[21] J. Chibane, A. Bansal, V. Lazova, G. Pons-Moll, Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7911–7920.

[22] A. Yu, V. Ye, M. Tancik, A. Kanazawa, Pixelnerf: Neural radiance fields from one or few images, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4578–4587.
[23] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, H. Su, Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 14124–14133.
[24] H. Lin, S. Peng, Z. Xu, Y. Yan, Q. Shuai, H. Bao, X. Zhou, Efficient neural radiance fields for interactive free-viewpoint video, in: SIGGRAPH Asia 2022 Conference Papers, 2022, pp. 1–9.
[25] D. Charatan, S.L. Li, A. Tagliasacchi, V. Sitzmann, Pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 19457–19467.
[26] Y. Chen, H. Xu, C. Zheng, B. Zhuang, M. Pollefeys, A. Geiger, T.-J. Cham, J. Cai, Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images, in: European Conference on Computer Vision, Springer, 2025, pp. 370–386.
[27] Q. Li, R. Fu, F. Tang, Depth assisted novel view synthesis using few images, Image Vis. Comput. 147 (2024) 105079.
[28] Z. Zhu, Z. Fan, Y. Jiang, Z. Wang, Fsgs: Real-time few-shot view synthesis using gaussian splatting, in: European Conference on Computer Vision, Springer, 2025, pp. 145–163.
[29] Z. Liu, J. Su, G. Cai, Y. Chen, B. Zeng, Z. Wang, GeoRGS: Geometric regularization for real-time novel view synthesis from sparse inputs, IEEE Trans. Circuits Syst. Video Technol. (2024).
[30] C. Yang, S. Li, J. Fang, R. Liang, L. Xie, X. Zhang, W. Shen, Q. Tian, Gaussianobject: High-quality 3d object reconstruction from four views with gaussian splatting, ACM Trans. Graph. 43 (6) (2024) 1–13.
[31] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P.P. Srinivasan, D. Verbin, J.T. Barron, B. Poole, et al., Reconfusion: 3d reconstruction with diffusion priors, 2023, arXiv preprint arXiv:2312.02981.
[32] S. Zhi, T. Laidlow, S. Leutenegger, A.J. Davison, In-place scene labelling and understanding with implicit scene representation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15838–15847.
[33] S. Kobayashi, E. Matsumoto, V. Sitzmann, Decomposing nerf for editing via feature field distillation, Adv. Neural Inf. Process. Syst. 35 (2022) 23311–23330.
[34] J. Kerr, C.M. Kim, K. Goldberg, A. Kanazawa, M. Tancik, Lerf: Language embedded radiance fields, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 19729–19739.
[35] G. Metzer, E. Richardson, O. Patashnik, R. Giryes, D. Cohen-Or, Latent-nerf for shape-guided generation of 3d shapes and textures, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 12663–12673.
[36] N. Müller, A. Simonelli, L. Porzi, S.R. Bulò, M. Nießner, P. Kontschieder, Autorf: Learning 3d object radiance fields from single view observations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3971–3980.
[37] K. Liu, F. Zhan, Y. Chen, J. Zhang, Y. Yu, A. El Saddik, S. Lu, E.P. Xing, Stylerf: Zero-shot 3d style transfer of neural radiance fields, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 8338–8348.
[38] J. Ye, N. Wang, X. Wang, Featurenerf: Learning generalizable nerfs by distilling foundation models, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 8962–8973.
[39] G. Liao, K. Zhou, Z. Bao, K. Liu, Q. Li, Ov-nerf: Open-vocabulary neural radiance fields with vision and language foundation models for 3d semantic understanding, 2024, arXiv preprint arXiv:2402.04648.
[40] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, ICLR 2015, Computational and Biological Learning Society, 2015.
[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[42] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, H. Aanæs, Large scale multi-view stereopsis evaluation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 406–413.
[43] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[44] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 586–595.
[45] X. Huang, S. Belongie, Arbitrary style transfer in real-time with adaptive instance normalization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1501–1510.
[46] J. Wynn, D. Turmukhambetov, Diffusionerf: Regularizing neural radiance fields with denoising diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4180–4189.
[47] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[48] B. Li, K.Q. Weinberger, S. Belongie, V. Koltun, R. Ranftl, Language-driven semantic segmentation, 2022, arXiv preprint arXiv:2201.03546.
[49] P. Wang, Y. Li, N. Vasconcelos, Rethinking and improving the robustness of image style transfer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 124–133.