Learning a Locally Unified 3D Point Cloud for View Synthesis
Meng You, Mantang Guo, Xianqiang Lyu, Hui Liu, and Junhui Hou, Senior Member, IEEE

Abstract—In this paper, we explore the problem of 3D point cloud representation-based view synthesis from a set of sparse source views. To tackle this challenging problem, we propose a new deep learning-based view synthesis paradigm that learns a locally unified 3D point cloud from source views. Specifically, we first construct sub-point clouds by projecting source views to 3D space based on their depth maps. Then, we learn the locally unified 3D point cloud by adaptively fusing points at a local neighborhood defined on the union of the sub-point clouds. Besides, we also propose a 3D geometry-guided image restoration module to fill the holes and preserve high-frequency details of the rendered novel views. Experimental results on three benchmark datasets demonstrate that our method can improve the average PSNR by more than 4 dB while preserving more accurate visual details, compared with state-of-the-art view synthesis methods. The code will be publicly available at https://github.com/mengyou2/PCVS.

Index Terms—Image-based rendering, view synthesis, 3D point clouds, point cloud fusion, deep learning.

I. INTRODUCTION

Given a collection of posed images observed from source views, view synthesis aims at generating photorealistic images at novel views. As view synthesis can benefit a variety of applications, e.g., robotics [1], 3D modeling [2], [3], virtual reality [4], and so on, a considerable number of view synthesis methods [5], [6], [7], [8], [9], [10] have been proposed over the past decades. Particularly, 3D point cloud representation-based methods, which generally render novel views from 3D scene representations, e.g., 3D point clouds/meshes, have been attracting attention.

Fig. 1 shows paradigms of different 3D point cloud representation-based view synthesis methods. Specifically, some methods [11], [12], [13], [14], [15], [16] illustrated in Fig. 1 (a) require inputting the full 3D point cloud, which may be difficult and labor-intensive to obtain. Conversely, other methods [17], [18], [19] shown in Fig. 1 (b) construct the point cloud from a single source view by predicting its depth map and further projecting the source view to 3D space. In reality, it is easy to obtain multiple source views of a scene, which may greatly improve the quality of the novel view synthesis. Unfortunately, it is not straightforward to generalize the single view-based paradigm shown in Fig. 1 (b) to handle multiple input source views well. To synthesize novel views from more than one source view [20], as shown in Fig. 1 (c), one can simply extend the single view-based paradigm, i.e., separately generating an intermediate novel view from each source view and then fusing the intermediate results via a confidence-based blending module. However, such a manner cannot take advantage of the relationship between different source views well, thus limiting its performance (see Section IV for the results).

To this end, we propose a new deep learning-based paradigm for view synthesis, as illustrated in Fig. 1 (d), which learns a unified 3D point cloud by adaptively fusing the sub-point clouds constructed from different source views. Specifically, we first project pixels of each source view into 3D space with corresponding depth maps, leading to sub-point clouds. We consider two scenarios regarding depth maps: (1) datasets already provide them; (2) a typical self/un-supervised depth estimation method for multi-view images is utilized to estimate them. As the depth maps are not perfect and inevitably contain errors, we adaptively fuse the sub-point clouds to a locally unified 3D point cloud representation with better quality via a point cloud fusion module. Besides, we propose a 3D geometry-guided image restoration module to fill holes and recover high-frequency details in the rendered novel view. Extensive experiments on three benchmark datasets demonstrate the significant superiority of our method over state-of-the-art view synthesis methods both quantitatively and visually. Besides, comprehensive ablation studies validate the effectiveness of the key modules of our framework.

In summary, the main contributions of this paper are three-fold:
• a new learning-based paradigm of point cloud representation-based view synthesis from multiple source views;
• a learnable point cloud fusion module to construct a locally unified point cloud representation of the input scene; and
• a 3D geometry-guided image restoration module to fill holes and recover high-frequency details of the rendered novel view.

Manuscript received 2 June 2023; revised 24 September 2023; accepted 26 September 2023. Date of publication 9 October 2023; date of current version 13 October 2023. This work was supported in part by the Hong Kong Research Grants Council under Grant 11218121 and Grant 21211518, in part by the Basic Research General Program of Shenzhen Municipality under Grant JCYJ20190808183030948, and in part by Hong Kong University Grants Committee under Grant UGC/FDS11/E02/22.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
The rest of this paper is organized as follows. Section II briefly reviews related works. Section III presents the proposed framework for view synthesis, followed by comprehensive experiments and analyses in Section IV. Finally, Section IV concludes the paper and discusses some future directions for improving the proposed view synthesis paradigm.

II. RELATED WORK

View synthesis is a long-standing problem in computer vision/graphics. Traditional methods [21], [22], [23], [24], [25] implement image-based rendering with the idea of blending synthesized images from source views. However, they are usually time-consuming and require dense inputs to achieve high-quality results. Recently, powerful deep learning has been widely used in the view synthesis area. Some methods [5], [6], [7], [8] perform pixel interpolation in the novel view by employing a neural network to estimate appearance flow between viewpoints. These methods lack scene geometry in their models, thus limiting their performance, especially for real scenes. To overcome these limitations, some methods attempt to explicitly utilize scene geometry by learning scene representations from large datasets. These methods first reconstruct 3D geometry, such as volumetric representations, point cloud representations, and neural representations, from images, and then render novel views.

A. Volumetric Representation-Based Methods

Recently, some methods have been learning the volumetric representations, such as voxel-based grid, multi-plane image (MPI), or layered depth image (LDI), from source images. Specifically, the voxel-based grid methods [26], [27], [28], [29], [30], [31], [32], [33] represent objects as a 3D volume in the form of voxel occupancies. Kar et al. [29] utilized the underlying 3D geometry in multi-view images to reconstruct the voxel occupancy grid by unprojecting image features along with viewing rays. Sitzmann et al. [34] proposed DeepVoxels to encode the view-dependent appearance of the scene without explicitly modeling its 3D geometry. MPI [35], [36], [37], [38], [39] represents the scene as a set of fronto-parallel planes at fixed depths, where each plane consists of an RGB image and an α map. Mildenhall et al. [37] proposed to expand each input view into an MPI, and then render the novel view by blending its adjacent MPIs. Li et al. [39] built the DeepMPI representation by adding latent features in MPI layers to render view-dependent lighting effects. Similar to MPI, LDI [40], [41], [42], [43], [44] keeps several depth and color values at every pixel and renders images by a back-to-front forward warping algorithm. Tulsiani et al. [42] used convolutional neural networks (CNN) to infer LDI representation from a single image and forward-splitting pixels to render a novel view. Shih et al. [43] proposed to generate LDI from RGBD images and employ a learning-based inpainting model to synthesize the color and depth information at occluded regions. Choi et al. [45] and Shi et al. [10] constructed the depth probability volume of the novel view to backward warp color images or feature maps from source views. Because of their discrete sampling, these mentioned volumetric representation-based methods could not achieve high-resolution results. To overcome this issue, we opt to use point clouds as the representation of the scene, which has the advantage of accurately representing complex and irregular geometries. Point clouds consist of individual points that can be positioned anywhere in 3D space, enabling them to capture intricate details and subtle features that might be challenging to depict with a voxel grid.

B. 3D Point Cloud Representation-Based Methods

Given the point cloud associated with descriptors, some methods [11], [12], [13], [16], [46] rasterize the point cloud into 2D image space with a learning-based differentiable rendering scheme. Instead of explicitly rendering the point cloud representation, Dai et al. [14] and Song et al. [15] proposed to extract features from the point cloud representation to construct a multi-plane 3D representation, and then render the color image from it via a neural network. Other methods [47], [48] use the point cloud as a base geometry model, which is further fitted to a surface mesh, and generate novel views by blending weights of sources on the mesh surface. Some methods construct the point cloud by estimating the depth maps of source views. Niklaus et al. [17] proposed to predict the depth map of the source view guided by semantic information and then render the novel view from the colored point cloud constructed from the source view based on the estimated depth map. Based on the estimated depth map, Wiles et al. [18] projected the feature map of the source view to the 3D space and then synthesized a novel view by decoding...
the feature map rendered from the point cloud. Le et al. [19] proposed to backward warp the synthesized novel view to the source one to supervise the depth estimation of the source view. Cao et al. [20] forward warped each input view with a differentiable point cloud renderer similar to [18], but extended to multiple inputs by fusing rendered view-dependent features. Alike et al. [13] and Rakhimov et al. [46] also considered the problem of rendering views from point clouds associated with feature descriptors. For a specific scene, Alieve et al. [13] and Rakhimov et al. [46] constructed a complete 3D point cloud of the scene, derived from a large number of source images using SfM or MVS techniques, which is then utilized to render all target views. During each training iteration, they only updated descriptors of points that were visible in the target view. Instead of reconstructing the entire scene, we focus on locally learning partial point clouds guided by the target view, i.e., given a set of source views, we obtain a point cloud from each source view and then fuse the resulting point clouds to produce a unified one that is further used to render in-between novel views. The unified point cloud varies with the set of input source views. Besides, Alike et al. [13] started with zero descriptor values and updated the neural descriptors via backpropagation through the loss derivatives. Rakhimov et al. [46] modeled the neural descriptor for the point as a linear combination of learnable basis functions, where basis functions were learned by MLPs with inputting view directions, and coefficients were obtained by solving multivariate linear regression problems. Our method constructs the neural descriptor for the point by concatenating the RGB value and image features and generates the neural descriptor of the point in the unified point cloud by interpolating the neural descriptors of its KNN points.

C. Neural Scene Representation-Based Methods

More recent methods represent the scene as neural radiance field (NeRF) by learning a continuous volumetric scene function [9]. Martin-Brualia et al. [49] extended NeRF to handle a collection of in-the-wild images. To tackle the problem that NeRF requires to be re-trained before generalizing to other unobserved scenes, some methods [50], [51], [52], [53], [54], [55] focus on NeRF generalization by involving the scene prior at the training phase. Yu et al. [50] attached image features behind the inputs of the NeRF. Wang et al. [51] combined image-based rendering method with NeRF by aggregating input features to estimate visibility and blend colors simultaneously. Chibane et al. [53] introduced the classical multi-view stereo ideas into NeRF, predicting RGB and density for each 3D point from its stereo correspondence in the image feature space. Chen et al. [55] leveraged 3D plane-swept cost volumes to reconstruct a neural encoding volume with per-voxel neural features, which is further regressed to volume density and radiance. Xu et al. [56] introduced scene geometry into NeRF using 3D point clouds with neural descriptors to model a radiance field.

Compared with our method, NeRF-based methods generally require a large number of training images to learn a high-quality radiance field. As NeRF-based methods only supervise the final integral radiance color instead of directly supervising RGB colors and density values of those 3D points, it is highly ill-posed. For predicting RGB colors and density values of sampled 3D points by a simple MLP with few training images. Our method can generate high-quality target views with only a few source views, e.g., two source views. Thus, our method would be more suitable than NeRF-based methods for the scenario where a limited number of source views are available. Since NeRF-based methods model a scene as a continuous function in 3D space, they can produce smooth animations or transitions between synthesized views. For our method, we construct different unified 3D point clouds from different sets of input source views, which leads to slight inconsistency between synthesized views. Thus, NeRF-based methods would be more suitable for our method for the scenario requiring highly smooth animations between synthesized views.

III. PROPOSED METHOD

Overview: As shown in Fig. 2, we consider synthesizing novel views from multiple source views by reconstructing a unified 3D point cloud as the scene representation. Specifically, we first construct a sub-point cloud from each source view by projecting its pixels to 3D space based on the depth map and then fuse the sub-point clouds to build a locally unified one via the point cloud fusion module (Section III-A). After rendering the novel view from the unified point cloud, we restore it by filling holes and recovering high-frequency details via a 3D geometry-guided restoration module (Section III-B). In what follows, we detail each module.
Note that rather than constructing a single 3D point cloud for a specific scene, which is then used for rendering all novel views, we aim to construct different locally unified 3D point clouds from different sets of source views, i.e., given a set of source views, we obtain a sub-point cloud from each source view and then fuse the resulting sub-point clouds to produce a locally unified one that is further used to render in-between novel views. The locally unified point cloud varies with the set of source views. Such a manner is potential memory- and computationally-efficient because a source view far away from the target view has a marginal contribution.

**A. Unified 3D Point Cloud Representation**

Given \( N \) source views of dimensions \( H \times W \) \( \{ I_n \in \mathbb{R}^{W \times H} \}_{n=1}^{N} \) and their depth maps \( \{ D_n \in \mathbb{R}^{W \times H} \}_{n=1}^{N} \), as well as the camera intrinsic and extrinsic parameters, we can project the pixels of \( \{ I_n \}_{n=1}^{N} \) to a common 3D coordinate system, producing \( N \) sub-colored point clouds. Ideally, pixels of \( \{ I_n \}_{n=1}^{N} \) corresponding to the same scene point should be projected to an identical 3D point.

However, due to inevitable errors in estimated depth maps or occlusions among different source views, there would be point deviations, i.e., the pixels corresponding to the same scene point are projected to different 3D locations in the union of projected sub-point clouds. Thus, we propose a point cloud fusion module to fuse these sub-point clouds into a locally unified one for synthesizing novel views in-between source viewpoints. As shown in Fig. 3, we first sample some anchor points from the union of sub-point clouds randomly. For each anchor point, we then seek its K nearest neighboring (KNN) points, which are further linearly interpolated with learned weights via an MLP to synthesize the point of the unified point cloud. In this way, we can eliminate point deviations effectively by adaptively adjusting the point cloud, and significantly improve the reconstruction quality of novel views.

Specifically, let \( \{ P_n \in \mathbb{R}^{3 \times 1} \}_{j=1}^{N} \) denote the projected sub-point clouds corresponding to \( \{ I_n \}_{n=1}^{N} \), where \( S = H \times W \). We first sample \( M \) points named anchor points from the union of \( \{ P_n \in \mathbb{R}^{3 \times 1} \}_{j=1}^{N} \) in a random manner to construct a base point cloud \( P_b = \{ x_i \in \mathbb{R}^{3} \}_{j=1}^{M} \), i.e.,

\[
P_b = f_s(P_1 \cup \cdots \cup P_n \cdots \cup P_N),
\]

where \( f_s(\cdot) \) denotes the random sampling process.

For a typical anchor point \( x_i \in P_b \), we seek its KNN from \( P_1 \cup \cdots \cup P_N \) in the sense of Euclidean distance, denoted by \( \{ x_i^k \in \mathbb{R}^{3} \}_{k=1}^{K} \). We then employ an MLP to learn the interpolation weight for a typical neighbor \( x_i^k \) by embedding the following information.

1) The Relative Position and Distance Between \( x_i^k \) and \( x_i \): We define the relative position \( \Delta x_i^k \) and distance \( d_i^k \) as the difference and the Euclidean distance between \( x_i^k \) and \( x_i \), respectively, i.e.,

\[
\Delta x_i^k = x_i^k - x_i, \quad \text{and} \quad d_i^k = \| x_i^k - x_i \|_2,
\]

where \( \| \cdot \|_2 \) denotes the \( \ell_2 \) norm of a vector.

2) The Descriptor Similarity Between \( x_i^k \) and \( x_i \): We first separately construct a descriptor for \( x_i^k \) and \( x_i \), i.e.,

\[
\tilde{f}_i^k = CAT(c_i^k, \hat{f}_i), \quad \text{and} \quad \hat{f}_i = CAT(c_i, f_i),
\]

where \( CAT(\cdot) \) is the concatenation operation, and \( f_i^k \in \mathbb{R}^{35} \), \( c_i^k \in \mathbb{R}^{3} \) and \( f_i^k \in \mathbb{R}^{32} \) (resp. \( f_i \in \mathbb{R}^{35} \), \( c_i \in \mathbb{R}^{3} \) and \( f_i \in \mathbb{R}^{32} \)) are the descriptor, the RGB values and the image feature corresponding to \( x_i^k \) (resp. \( x_i \)), respectively, and we learn the image features \( f_i^k \) and \( f_i \) from source views by employing a sub-CNN. Then, we compute the descriptor similarity \( s_i^k \) as the cosine similarity between \( \tilde{f}_i^k \) and \( \hat{f}_i \), i.e.,

\[
s_i^k = \frac{\tilde{f}_i^k \cdot \hat{f}_i}{\| \tilde{f}_i^k \| \| \hat{f}_i \|}.
\]

We finally construct the embedding \( e_i^k \) as

\[
e_i^k = CAT(\Delta x_i^k, d_i^k, s_i^k),
\]

and separately predict the interpolation weight \( w_{ip}^k \) and \( w_{ij}^k \) for the point position and its corresponding descriptor as

\[
w_{ip}^k = f_p(e_i^k; \theta_p), \quad \text{and} \quad w_{ij}^k = f_c(e_i^k; \theta_c),
\]

where \( f_p(\cdot; \cdot) \) and \( f_c(\cdot; \cdot) \) are the learnable MLPs parameterized by \( \theta_p \) and \( \theta_c \), respectively. With the learned weights, we can obtain the point and its corresponding descriptor of the unified 3D point cloud \( \tilde{P} = [\tilde{x}_i \in \mathbb{R}^{3}]_{i=1}^{M} \) as

\[
\tilde{x}_i = \sum_{k=1}^{K} w_{ip}^k x_i^k, \quad \text{and} \quad \tilde{f}_i = \sum_{k=1}^{K} w_{ij}^k \tilde{f}_i^k.
\]

Remark. Random sampling has demonstrated effectiveness in deep learning-based semantic segmentation for large-scale point clouds [57]. When the sampling rate is high, random

Fig. 3. Flowchart of the point cloud fusion module.
sampling can efficiently produce an approximate uniformly-distributed sub-set. Besides, another potential advantage of random sampling is that it can introduce greater diversity in the training data, making our method more robust and generalizing better to new and unseen point clouds. Since the sampled anchor point cloud is not directly used for rendering but rather to find subsets of the union of sub-point clouds by searching KNN, any information loss resulting from random sampling can be compensated by fusing all points in the subset to gather local information and encode it into the interpolated descriptor. We also refer readers to the effectiveness and advantages of random sampling for this task in Section IV-C.

B. 3D Geometry-Guided Image Restoration

With the unified 3D point cloud representation as well as the camera intrinsic and extrinsic parameters of the target view, we can render a coarse novel view image, denoted as $\tilde{I}_t$, via a typical renderer. In this paper, we adopt the differentiable renderer involved in PyTorch3D [58] that splats each point to a circular region in screen-space whose opacity decreases away from the region’s center and computes the value of each pixel by blending information for the neighboring points whose splatted regions overlap the pixel. However, as illustrated in the first column of Fig. 4 (b), $\tilde{I}_t$ still suffers from holes since there are still pixel positions not in the splatted regions of any points. Besides, blending the colors of several points into one pixel also causes the missing of high-frequency details in $\tilde{I}_t$. To deal with these issues, as shown in Fig. 4 (a), we propose a geometry-guided image restoration module to restore $\tilde{I}_t$ by filling holes and propagating high-frequency details from the source views under the guidance of the depth map of the novel view.

Specifically, we first project $\hat{P}$ associated with descriptors to render a feature map of the novel view, denoted by $\tilde{F}_t$. To fill the holes in $\tilde{F}_t$, we learn an additive map by using a sub-CNN $f_h(\cdot; \cdot)$ parameterized by $\theta_h$ with $\tilde{F}_t$ as input, i.e.,

$$\tilde{I}^r_t = f_h(\tilde{F}_t; \theta_h) + \tilde{I}_t.$$  

(8)

Furthermore, we transform $\hat{P}$ to the camera coordinate system of the novel view, and then project the points associated with z-coordinates to render the depth map of the novel view, based on which, we separately warp each source view in $\{I_n\}_{n=1}^N$ to the novel view, generating warped source images $\{\tilde{I}_n\}_{n=1}^N$. To propagate the high-frequency details from the source views to $\tilde{I}^r_t$, we also learn an additive map by using another sub-CNN $f_r(\cdot; \cdot)$ parameterized by $\theta_r$ with the concatenation of $\tilde{I}^r_t$ and warped source images as input, i.e.,

$$\tilde{I}_t = f_r([\text{CAT}(\tilde{I}^r_t, \tilde{I}_1, \cdots, \tilde{I}_n, \cdots, \tilde{I}_N); \theta_r] + \tilde{I}^r_t.$$  

(9)

where $\tilde{I}_t$ is the finally synthesized novel view. As an example, Fig. 4 (b) visually illustrates the effect of the proposed 3D geometry-guided image restoration module.

Remark. From a technical point of view, the order of warped images (i.e., the orders of warped images are inconsistent during training and inference) could affect the performance of our method because the warped source images could differ from each other to a large extent due to different relative positions between source views and the target view, resulting in different contributions to the synthesized view (see the experimental verification in Section IV-C.6). Thus, to eliminate the order effect, we should keep the order of source views consistent during training and inference.

C. Loss Function

We supervise both the intermediate and final rendered results by calculating the photometric loss with the ground-truth novel view $I_t$. We define the photometric loss function $\ell_r(\cdot, \cdot)$ as

$$\ell_r(\tilde{I}^r_t, I_t) = \|\tilde{I}^r_t - I_t\|_1 + \sum_{l=1}^L \lambda_l \|\phi_l(\tilde{I}^r_t) - \phi_l(I_t)\|_1,$$  

(10)

where $\|\cdot\|_1$ denotes the $\ell_1$ norm, $\tilde{I}^r_t \in [\tilde{I}_t, \tilde{I}^r_t, \tilde{I}_t]$, $\{\phi_l\}_{l=1}^L$ is a set of layers in a pre-trained VGG-19 network [59], the weights $\lambda_l$ are set to the inverse of the number of neurons in each layer, and $L = 5$. Thus, the total photometric loss is calculated as

$$\tilde{\ell}_r = \ell_r(\tilde{I}^r, I_t) + \ell_r(\tilde{I}^r, I_t) + \ell_r(\tilde{I}_t, I_t).$$  

(11)

D. Practical Extension to the Scenario Without Depth Maps

As depth maps are not always available in practice, we also extend our method by plugging a self-supervised depth estimation module, which is jointly trained with our framework. Such an extension can demonstrate the generalization ability of our method. We employ the network of MVSNet [60], a multi-view-based depth estimation network, to separately estimate the depth maps of input source views. However, MVSNet [60] requires ground-truth depth maps as supervision during
training. To this end, we adopt a self-supervised loss term to regularize the learning of this network.

Specifically, to regularize MVSNet [60] generating the depth map of a typical source view, we warp other source views and the ground-truth target view to the current source view based on the estimated depth map, and then minimize the errors between the current source view and the warped images. Specifically, to estimate the depth map \( D_n \) (\( n \in [1, N] \)) of a typical source view \( I_n \), we inversely warp other views \( \{I_i\}_{i=1, \neq n} \) and the ground-truth novel view \( I_0 \) to \( I_n \) based on \( D_n \), leading to the warped images \( \{I_i \rightarrow n\}_{i=1, \neq n} \) and \( I_0 \rightarrow n \), respectively. Formally, we write the self-supervised loss term for estimating \( D_n \) as

\[
\ell^\text{self}_{dn} = \| (I_n - I_0) \odot M_{i \rightarrow n} \|_1 + \sum_{i=1}^{N} \| (I_n - I_i \rightarrow n) \odot M_{i \rightarrow n} \|_1, \quad i \neq n,
\]

where \( M_{i \rightarrow n} \) is the binary mask corresponding to \( I_i \rightarrow n \), where 0 (resp. 1) indicates the projected pixel is out of (resp. in) the range of the warped image. To further regularize the depth estimation module, apart from the L1 loss between the warped image and the source view, i.e., Eq. (12), we also add the SSIM loss between them. For a typical pair of warped image and source view, we define the SSIM loss function \( \ell^\text{sim}_d(X, Y, M) \) as

\[
\ell^\text{sim}_d(X, Y, M) = \frac{1 - SSIM(M \odot X, M \odot Y)}{2},
\]

where \( X, Y, M \) denote the warped image, the source view, and the corresponding mask, respectively. Thus, we calculate the SSIM loss \( \ell^\text{sim}_d \) for estimating \( D_n \) as

\[
\ell^\text{sim}_d = \sum_{i=1}^{N} \ell^\text{sim}_d(I_n, I_i \rightarrow n, M_{i \rightarrow n}), \quad i \neq n.
\]

Moreover, to promote smoothness of \( D_n \), we penalize the \( \ell_1 \) norm of the gradient, denoted as \( \ell^\text{smooth}_{dn} \):

\[
\ell^\text{smooth}_{dn} = \| \nabla_x D_n \|_1 + \| \nabla_y D_n \|_1,
\]

where \( \nabla_x \) and \( \nabla_y \) are the gradient operators for the spatial domain. Thus, the total depth estimation loss is calculated as

\[
\ell_d = \sum_{n=1}^{N} (\lambda_d \ell^\text{self}_{dn} + \lambda_d \ell^\text{sim}_{dn} + \lambda_d \ell^\text{smooth}_{dn}),
\]

where we empirically set \( \lambda_d = 12 \), \( \lambda_d = 6 \), and \( \lambda_d = 0.18 \). If we employ the self-supervised depth estimation module, we calculate the total training loss as

\[
\ell = \hat{\ell} + \ell_d.
\]

Considering that estimated depth maps have deviations, we measure the depth estimation quality, namely probability map, proposed in MVSNet [60], which takes the probability sum over the four nearest depth hypotheses based on the estimated depth value to measure the estimation quality, as an added embedding term in the point cloud fusion module. We denote the depth estimation quality of \( x^k_i \) by \( p_i^k \) and rewrite Eq. (5) as

\[
p_i^k = \text{CAT}(\Delta x_i^k, d_i, p_i^k, s_i^k).
\]

IV. EXPERIMENTS

A. EXPERIMENT SETTINGS

1) Datasets: We conducted extensive experiments on three challenging datasets, including RealEstate10K [35], Tanks and Temples [61], and DTU [62]. All three datasets contain rotations and translations in camera movement, and particularly the first one has minor movement and the latter two are more significant.

Specifically, RealEstate10K is a huge dataset derived from 80k video clips and we chose a subset of 85 scenes for training and 17 scenes for testing. Tanks and Temples contains more complex indoor and outdoor scenes with irregular camera trajectories. Following the settings in [47], we used 17 of 21 scenes for training and the remaining four (i.e., Truck, Train, M60, and Playground) for testing. Reference [47] also provides depth maps for Tanks and Temples, which were derived from a 3D surface mesh reconstructed from all views. DTU consists of 124 different scenes, where each was captured by 49 cameras located regularly on a sphere. We used DTU only for testing to verify the generalization ability. Particularly, we adopted the evaluation scene set of 18 scenes provided by Yao et al. [60]. Reference [60] also provides depth maps for DTU by rendering the 3D mesh reconstructed from the ground-truth point cloud to each viewpoint.

For RealEstate10K, Tanks and Temples, and DTU datasets, we tested all views of a typical test scene, i.e., all views in a test scene are test views. For each view of the test scene, we first designated it as the target view, and then selected a specific number (2, 3, or 4) of source views for it. Specifically, for the 2-input setting, we choose the left and right adjacent frames of the target view as source views. For the setting with more than two source views, in addition to selecting two adjacent frames of the target view, we adopted the view selection result by [47] to add more source views.

2) Implementation Details: We implemented the encoder using Res-UNet [63] for image feature extraction and set the feature dimensions as 32. For cost volume construction in our self-supervised depth estimation module, we sampled 128 depth plane layers uniformly within the depth range of the scene. We implemented the inpainting and refinement components contained in the restoration module using ResNet with 6 blocks and 4 blocks, respectively. During training, we first trained the network without the refinement component \( f_r(\cdot) \). Then, we trained the refinement component \( f_r(\cdot) \) with prior network parameters fixed. We used the Adam optimizer with the learning rate equal to \( 1e^{-5} \).

B. COMPARISON WITH STATE-OF-THE-ART METHODS

We compared our method with the three most recent view synthesis methods under the setting of synthesizing novel views from two input source views, including two image-based rendering methods, i.e., FVS [47] and SVNVS [10], and one...
TABLE I

QUANTITATIVE COMPARISON OF DIFFERENT METHODS ON Tanks and Temples, DTU AND RealEstate10K. WE EXCLUDED THE FVS METHOD, POST-FUSION-D, AND OURS-D FOR RealEstate10K DUE TO THE LACK OF DEPTH MAPS. ↑ (RESP. ↓) MEANS THE LARGER (RESP. SMALLER), THE BETTER. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

| Method          | Tanks and Temples | DTU | RealEstate10K |
|-----------------|-------------------|-----|---------------|
|                 | LPIPS↑ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↓ | SSIM↓ | LPIPS↑ | PSNR↑ | SSIM↑ |
| IBRNet [51]     | 0.321  | 19.35  | 0.649  | 0.288  | 19.94  | 0.713  | 0.080  | 20.81  | 0.919  |
| SVNVS [10]      | 0.218  | 20.80  | 0.739  | 0.541  | 15.50  | 0.394  | 0.116  | 27.11  | 0.897  |
| Ours-W          | 0.189  | 21.28  | 0.751  | 0.140  | 24.24  | 0.859  | 0.028  | 36.88  | 0.975  |
| FVS [47]        | 0.302  | 18.91  | 0.661  | 0.236  | 20.14  | 0.723  |        |        |        |
| Post-Fusion-D   | 0.155  | 23.47  | 0.848  | 0.193  | 22.50  | 0.823  |        |        |        |
| Ours-D          | 0.128  | 24.77  | 0.869  | 0.138  | 24.59  | 0.874  |        |        |        |

TABLE II

QUANTITATIVE COMPARISON OF DIFFERENT METHODS ON THE FOUR SCENES OF Tanks and Temples. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

| Method          | Train | Playground | M60 | Truck |
|-----------------|-------|------------|-----|-------|
|                 | LPIPS↑ | PSNR↑ | SSIM↑ | LPIPS↑ | PSNR↑ | SSIM↑ | LPIPS↑ | PSNR↑ | SSIM↑ |
| IBRNet [51]     | 0.378  | 17.37  | 0.538  | 0.291  | 23.01  | 0.726  | 0.287  | 18.82  | 0.705  | 0.311  | 19.92  | 0.660  |
| SVNVS [10]      | 0.232  | 19.73  | 0.674  | 0.204  | 22.98  | 0.770  | 0.210  | 20.29  | 0.773  | 0.222  | 20.58  | 0.715  |
| Ours-W          | 0.233  | 20.04  | 0.692  | 0.158  | 23.29  | 0.780  | 0.167  | 21.10  | 0.808  | 0.183  | 20.97  | 0.732  |
| FVS [47]        | 0.341  | 18.22  | 0.597  | 0.285  | 20.91  | 0.696  | 0.354  | 16.28  | 0.633  | 0.180  | 21.32  | 0.754  |
| Post-Fusion-D   | 0.167  | 21.82  | 0.812  | 0.156  | 26.19  | 0.876  | 0.162  | 22.42  | 0.861  | 0.123  | 24.36  | 0.856  |
| Ours-D          | 0.145  | 22.63  | 0.835  | 0.119  | 27.50  | 0.887  | 0.121  | 24.78  | 0.897  | 0.118  | 24.92  | 0.865  |

NeRF-based method, i.e., IBRNet [51]. For fair comparisons, we retrained and evaluated these methods with the same data and settings as ours and well-tuned hyperparameters. We named our method Ours-D when inputting pre-processed depth maps and Ours-W when using self-supervised depth estimation. To directly demonstrate the effectiveness of the proposed unified 3D point cloud representation, we also constructed a baseline shown in Fig. 1(c) called Post-Fusion-D by replacing our point cloud fusion module with the confidence-based blending of rendered results. Specifically, we projected each source view with its features to 3D space and rendered it to the target viewpoint separately. The rendered image was inpainted with the corresponding rendered feature map. Then we added a CNN to learn confidence maps for these inpainted images and fused them into one result further refined by warped source views. The encoding, inpainting, and refinement networks are the same as our method.

1) Quantitative Comparisons: Table I lists the average PSNR, SSIM, and LPIPS [64] of different methods on each of the three datasets. Besides, Table II lists the results of the four scenes contained in Tanks and Temples. From Tables I and II, it can be observed that

- both Ours-W and Ours-D consistently outperform other methods on all datasets, including both major and significant movement, and especially Ours-W improves the PSNR of the second best method by more than 7 dB on RealEstate10K, demonstrating its significant superiority;
- we used the models trained on Tanks and Temples to perform testing on DTU. The superiority of Ours-W and Ours-D over other compared methods on DTU is more prominent than that on Tanks and Temples, demonstrating its stronger generalization ability;
- compared with the results on Tanks and Temples, the performance of all methods improves significantly on RealEstate10K. The reason may be that the camera movement of RealEstate10K is minor, conducive to obtaining more accurate geometry information;
- Ours-D improves the reconstruction quality by 1.3 dB and 2.0 dB on Tanks and Temples and DTU, respectively, compared with the baseline Post-Fusion-D, demonstrating the superiority of unifying sub-point clouds by our 3D point cloud fusion strategy over 2D confidence-based image blending;
- Ours-D performs better than Ours-W on Tanks and Temples by more than 3dB. The reason is that self-supervised depth estimation can be challenging for datasets with large and irregular camera movements, especially when given only a few input images.

2) Visual Comparisons: We also visually compared the results of different methods on Tanks and Temples, DTU, and RealEstate10K in Fig. 6, Fig. 7, and Fig. 5, respectively, where it can be observed that our method can produce better details.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 6. Visual comparison of different methods on Tanks and Temples. (a) FVS [47], (b) IBRNet [51], (c) SVNVS [10], (d) Post-Fusion-D, (e) Ours-W, (f) Ours-D, (g) Ground Truth. Zoom in to see details.

TABLE III
COMPARISONS OF INFERENCE TIME (IN SECONDS PER VIEW) OF DIFFERENT METHODS ON THE DTU DATASET

| Time (in seconds) | FVS | SVNVS | IBRNet | Ours-W | Ours-D |
|-------------------|-----|-------|--------|--------|--------|

than all the compared methods for all datasets under the same experimental configuration. Using two source views and depth maps, the FVS can not build an accurate 3D mesh, resulting in noticeable blurred artifacts, especially on the boundaries of the synthesized image. For the IBRNet, the blurred artifacts and ghost effects appear at high-frequency regions and occlusion boundaries. As a NeRF-based method, IBRNet needs more source views to achieve satisfactory results. For SVNVS, there are blurred artifacts and slight color distortions compared with the ground truth. For the Post-Fusion-D, the edges of results show severe color distortions and artifacts. Additionally, Fig. 7 shows the generalization ability advantages of Ours-W and Ours-D, and both of them can get sharp and clear results.

3) Efficiency Comparisons: Table III compares the inference time of different methods on DTU. Note that we implemented all methods on a Linux server with Intel CPU Xeon Gold 6226R @ 2.90GHz, 512GB RAM, and NVIDIA GeForce RTX 3090 GPU. Ours-W takes more time to synthesize novel views than Ours-D due to the self-supervised depth estimation module. Ours-D is slightly slower than FVS but faster than SVNVS and IBRNet. Taking the reconstruction quality and efficiency together, we believe our method is the best.

C. Ablation Study and Analysis

1) Network Modules: We carried out comprehensive ablation studies on Tanks and Temples with pre-processed depth maps as input to validate the effectiveness of the three key components of our framework, i.e., the image inpainting component, the point cloud fusion module, and the refinement component. We constructed a baseline by excluding these three components. We sequentially added each component to the base model until all three components were included to form the complete model. When removing the point cloud fusion module, we simply merged the sub-point clouds to generate a unified point cloud to enable the method.

TABLE IV
QUANTITATIVE RESULTS OF THE ABLATION STUDIES.

| Inpainting | Fusion | Refinement | LPIPS↓ | PSNR↑ | SSIM↑ |
|------------|--------|------------|--------|-------|-------|
| (1)    | x      | x          | 0.561  | 13.79 | 0.362 |
| (2)    | ✓      | x          | 0.157  | 22.97 | 0.819 |
| (3)    | ✓      | ✓          | 0.138  | 24.18 | 0.853 |
| (4)    | ✓      | ✓          | 0.159  | 23.34 | 0.836 |
| (5)    | ✓      | ✓          | 0.128  | 24.77 | 0.869 |
Quantitative Results of the Ablation Studies for the Embedded Information in Point Cloud Fusion.

| Relative position & distance | Depth estimation quality | Descriptor similarity | LPIPS | PSNR | SSIM |
|-----------------------------|-------------------------|-----------------------|-------|------|------|
| ✓                           | ×                       | ×                     | 0.212 | 20.67 | 0.724 |
| ✓                           | ✓                       | ✓                     | 0.211 | 20.95 | 0.733 |
| ✓                           | ✓                       | ✓                     | 0.189 | 21.28 | 0.751 |

Fig. 8. Visual results of the ablation studies on network modules. (a) Baseline. (b) Baseline + Inpainting. (c) Baseline + Inpainting + Point Cloud Fusion. (d) Baseline + Inpainting + Refinement. (e) Complete model = Baseline + Inpainting + Point Cloud Fusion + Refinement. (f) Ground truth. Zoom in to see details.

Fig. 9. Visual comparison of the 3D point clouds. The left images in (a) and (b) visualize the point clouds by directly projecting depth maps of input views into 3D space, and the right images show the point clouds learned by the proposed fusion method.

Table V, it can be seen that each embedded information is helpful in improving the overall performance, verifying the rationality of our design.

3) Performance of Our Method Under Various Numbers of Input Source Views: Due to the limited GPU memory, the proposed unified point cloud representation cannot directly process four sub-point clouds. Thus, we adopted a progressive fusion strategy, i.e., fusing three sub-point clouds as an intermediate unified point cloud further fused with the fourth sub-point cloud. Such a progressive manner can be straightforwardly extended to the case with more input views. As shown in Table VI, the 3-input setting performs better than the 2-input because more information is available within the third input view. However, when given one more input view, there is no noticeable improvement. The main reason is that the camera motion in the dataset is very large, and the viewpoint of the fourth view is too far from the target view to provide useful information.

4) Performance of Our Method Under Different Sampling Strategies: We experimented with different sampling strategies to evaluate our method. We replaced random sampling used in Ours-D with farthest point sampling (FPS) or voxel-based downsampling when performing inference. Given source views and their depth maps, we project the pixels of source views to a common 3D coordinate system using the pinhole camera model, producing sub-colored point clouds that are contained within frustum-shaped viewing volumes as shown in Fig. 10 (a). Since foreground objects of the scene typically account for a relatively larger image proportion and a smaller depth range, while background objects account for a smaller
image proportion and a larger depth range, the points of foreground objects are denser than those of the background objects in the viewing frustum. Thus, the points in the union of these sub-point clouds are also non-uniformly distributed in the 3D space, i.e., points of foreground objects are denser than those of background objects. Given such a non-uniform point cloud, we give a schematic diagram of their sampling methods in Fig. 10 (b). Random sampling chooses each point in the union point cloud with the same probability, which results in sampling more points in foreground objects and fewer points in background objects. FPS selects a subset of points from a larger point cloud by iteratively selecting the point that is farthest from the previously selected points, resulting in sampled points being spread over the shape uniformly. When points of foreground objects are denser than those of background objects, FPS tends to sample a smaller percentage of points in the foreground objects while a larger percentage of points in the background objects, leading to less preservation of foreground information than random sampling. Voxel sampling divides the 3D space into small voxels and only retains one point per voxel. Since a typical voxel in foreground objects usually contains more points than the voxel in background objects, it leads to less preservation of foreground points after voxel sampling.

We provided visual results of the unified point cloud projections from various sampling strategies in Fig. 11. It can be observed that the random sampling method preserves most foreground information from the original point cloud, while FPS and Voxel methods are prone to significant foreground information loss. Additionally, it is crucial that the sampled points cover the main geometry of the object. However, FPS and Voxel methods can compromise the structural integrity, especially when sampling a relatively small number of points, such as in the case of a broken wooden pile. We also show the quantitative results in Table VII, where it is evident that random sampling outperforms the other techniques.

5) Performance of Our Method Under Different Hyper-parameter Settings: We investigated how the numbers of sampled points $M$ and neighboring point number $K$ affect the performance of Ours-D under the 2-input setting. Specifically, we tested different hyperparameter settings by changing the values of $K$ and $R$, where $M = R \cdot S$, $S = H \cdot W$. We compared the average PSNR across different settings on Tanks and Temples, as shown in Fig. 12. The small fluctuation in PSNR values across the different settings indicates the robustness of our model. In our experiments, we adopt $K = 12$ and $R = 0.8$, as a trade-off among the quality, the computation cost, and the memory cost. For the scenario with more input source views, we proportionally scale up the values of $K$ and $R$.

6) Performance of Our Method Under Different Orders of Source Views: We conducted experiments to study how the order of source views affects Ours-D. Specifically, we examined the previously trained network under two scenarios: (i) only changing the order of warped source images in the concatenation operation in Eq. (9), and (ii) changing the order of input source images throughout the entire pipeline. For the 2-input setting, we swapped the order of the two input

| # Input Views | Train | Playground | M60 | Truck |
|---------------|-------|------------|-----|-------|
|               | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ |
| 2             | 0.145 | 22.63 | 0.835 | 0.119 | 27.50 | 0.887 | 0.121 | 24.78 | 0.897 | 0.145 | 22.63 | 0.835 |
| 3             | 0.144 | 23.13 | 0.828 | 0.125 | 27.51 | 0.887 | 0.119 | 24.94 | 0.891 | 0.110 | 25.26 | 0.868 |
| 4             | 0.145 | 23.23 | 0.831 | 0.128 | 27.49 | 0.887 | 0.115 | 25.02 | 0.893 | 0.107 | 25.23 | 0.870 |

| # Sampling Strategy | Train | Playground | M60 | Truck |
|---------------------|-------|------------|-----|-------|
|                    | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ |
| Random              | 0.145 | 22.63 | 0.835 | 0.119 | 27.50 | 0.887 | 0.121 | 24.78 | 0.897 | 0.118 | 24.92 | 0.865 |
| FPS                 | 0.146 | 22.96 | 0.830 | 0.118 | 27.51 | 0.886 | 0.120 | 24.78 | 0.892 | 0.137 | 23.78 | 0.845 |
| Voxel               | 0.174 | 21.89 | 0.811 | 0.219 | 24.18 | 0.822 | 0.179 | 21.59 | 0.849 | 0.195 | 22.29 | 0.806 |

Fig. 11. Visual comparison of projected unified point cloud using different sampling strategies.

Fig. 12. The average PSNR on Tanks and Temples on different settings of hyper-parameters $K$ and $R$. 
source views, and for the 3-input setting, we swapped the order of the first and third views. The changed order differs from the source view selection described in Section IV-A.1 used for training. Table VIII lists quantitative results, showing whereas the added third view often has a more significant performance for the 2-input setting but results in a significant performance decrease for the 3-input setting. The possible reason is that, for the 2-input setting, both input views are usually near the target, whereas the added third view often has a more significant distant viewpoint from the target view in the 3-input setting. Thus, we should keep the source view selection consistent during training and inference to eliminate the order effect.

V. CONCLUSION AND DISCUSSION

We have presented a new learning-based paradigm for view synthesis, which learns a locally unified 3D point cloud representation from multiple source views. Precisely, we constructed the unified point cloud by adaptively fusing points at a local neighborhood defined on the union of the sub-point clouds projected from source views. Owing to the learning of the unified scene representation, as well as a 3D geometry-guided image restoration module to fill the holes and recover high-frequency details of the rendered novel view, our view synthesis paradigm reconstructions novel views with much higher quantitative and visual quality, compared with state-of-the-art methods.

In the future, the following directions could be considered for improving the proposed paradigm. First, random sampling used in our point cloud fusion module to generate the base point clouds, may not be the optimal choice, and our paradigm will benefit from more advanced and efficient sampling methods. Second, efforts can be made to address the slight inconsistency between sub-point clouds, resulting from the progressive fusion strategy to balance the GPU memory cost when handling multiple source views. Third, relative position or confidence map-aware algorithms could be investigated to mitigate the impact of the order of source views. Fourth, it is valuable to investigate inward-facing or long trajectory datasets to explore the potential of our framework in handling diverse scenarios. Finally, it would be interesting to investigate joint camera pose estimation and novel view synthesis for a self-contained paradigm.

REFERENCES

[1] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, “KPAM: Key-Point affordances for category-level robotic manipulation,” 2019, arXiv:1903.06854.

[2] R. Anderson et al., “Jump: Virtual reality video,” ACM Trans. Graph., vol. 35, no. 6, pp. 1–13, Nov. 2016.

[3] A. Collet et al., “High-quality streamable free-viewpoint video,” ACM Trans. Graph., vol. 34, no. 4, pp. 1–13, Jul. 2015.

[4] S.-E. Wei et al., “VR facial animation via multiview image translation,” ACM Trans. Graph., vol. 38, no. 4, pp. 1–16, Aug. 2019.

[5] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, “View synthesis by appearance flow,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 286–301.

[6] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg, “Transformation-grounded image generation network for novel 3D view synthesis,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 702–711.

[7] S.-H. Sun, M. Huh, Y.-H. Liao, N. Zhang, and J. J. Lim, “Multi-view to novel view: Synthesizing novel views with self-learned confidence,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 155–171.

[8] J. Song, X. Chen, and O. Hilliges, “Monocular neural image based rendering with continuous view control,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 4089–4099.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 405–421.

[10] Y. Shi, H. Li, and X. Yu, “Self-supervised visibility learning for novel view synthesis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 9670–9679.

[11] M. Meshry et al., “Neural rerendering in the wild,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 6871–6880.

[12] F. Pittaluga, J. Koppal, S. B. Kang, and S. N. Sinha, “Revealing scenes by inverting structure from motion reconstructions,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 145–154.

[13] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky, “Neural point-based graphics,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 696–712.

[14] P. Dai, Y. Zhang, Z. Li, S. Liu, and B. Zeng, “Neural point cloud rendering via multi-plane projection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 7827–7836.

[15] Z. Song, W. Chen, D. Campbell, and H. Li, “Deep novel view synthesis from colored 3D point clouds,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 1–17.

[16] D. Rückerl, L. Franke, and M. Stamminger, “ADOP: Approximate differentiable one-pixel point rendering,” ACM Trans. Graph., vol. 41, no. 4, pp. 1–14, Jul. 2022.

[17] S. Niklaus, L. Mai, J. Yang, and F. Liu, “3D ken burns effect from a single image,” ACM Trans. Graph., vol. 38, no. 6, pp. 1–15, Dec. 2019.

[18] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, “SynSin: End-to-end view synthesis from a single image,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 145–154.

[19] H. Le, T. Mensink, P. Das, and T. Gevers, “Novel view synthesis from a single image,” in Proc. Brit. Mach. Vis. Conf. Comput. Vis. Pattern Recognit. (BMVC), 2020, pp. 1–19.

[20] A. Cao, C. Rockwell, and J. Johnson, “FWD: Real-time novel view synthesis with forward warping and depth,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 15692–15703.

[21] C. Buehler, M. Bosse, L. Mcmillan, S. Gortler, and R. Tedrake, “Unstructured lumigraph rendering,” in Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn., Aug. 2001, pp. 425–432.

[22] P. E. Debevec, C. J. Taylor, and J. Malik, “Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach,” in Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn., 1996, pp. 11–20.

---

| TABLE VIII | QUANTITATIVE RESULTS OF THE EFFECT OF THE ORDER OF SOURCE VIEWS ON THE PERFORMANCE OF Ours-D | Source View | Train | LPIPS\textsuperscript{p} | PSNR\textsuperscript{r} | SSIM\textsuperscript{t} | LPIPS\textsuperscript{p} | PSNR\textsuperscript{r} | SSIM\textsuperscript{t} | LPIPS\textsuperscript{p} | PSNR\textsuperscript{r} | SSIM\textsuperscript{t} | LPIPS\textsuperscript{p} | PSNR\textsuperscript{r} | SSIM\textsuperscript{t} |
|-----------|---------------------------------|------------|------|----------------|----------------|------|----------------|----------------|------|----------------|----------------|------|----------------|----------------|------|----------------|----------------|
| 2-input   | Change order of warped images   | 0.145      | 22.63 | 0.835         | 0.119          | 27.30 | 0.887         | 0.121          | 24.78 | 0.897         | 0.117          | 22.83 | 0.835         | 0.105          | 25.00 | 0.867         |
| 2-input   | Change order of source images   | 0.146      | 22.65 | 0.832         | 0.121          | 27.47 | 0.885         | 0.120          | 24.86 | 0.900         | 0.116          | 25.01 | 0.867         |
| 3-input   | Change order of warped images   | 0.144      | 23.13 | 0.828         | 0.125          | 27.51 | 0.887         | 0.119          | 24.94 | 0.891         | 0.110          | 25.26 | 0.868         |
| 3-input   | Change order of source images   | 0.157      | 22.60 | 0.812         | 0.143          | 26.88 | 0.875         | 0.134          | 24.29 | 0.878         | 0.113          | 24.89 | 0.866         |
Mantang Guo received the B.Eng. and M.Eng. degrees from the School of Computer Science, Northwestern Polytechnical University, China, in 2016 and 2019, respectively, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, in 2023. His current research interests include computational photography, light field reconstruction, and deep learning.

Hui Liu received the B.Sc. degree in communication engineering from Central South University, Changsha, China, the M.Eng. degree in computer science from Nanyang Technological University, Singapore, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong. From 2014 to 2017, she was a Research Associate with the Maritime Institute, Nanyang Technological University. She is currently an Assistant Professor with the School of Computing Information Sciences, Caritas Institute of Higher Education, Hong Kong. Her current research interests include image processing and machine learning.

Xianqiang Lyu received the B.Eng. and M.Eng. degrees from the School of Computer Science, Northwestern Polytechnical University, Xi’an, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong. His current research interests include computational photography, light field image processing, neural rendering, and deep learning.

Junhui Hou (Senior Member, IEEE) received the B.Eng. degree in information engineering (talented students program) from the South China University of Technology, Guangzhou, China, in 2009, the M.Eng. degree in signal and information processing from Northwestern Polytechnical University, Xi’an, China, in 2012, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2016. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong. His current research interests include multi-dimensional visual computing. He is an elected member of IEEE MSA-TC, VSPC-TC, and MMSP-TC. He received the Early Career Award (3/381) from the Hong Kong Research Grants Council in 2018. He is currently serving as an Associate Editor for IEEE Transactions on Visualization and Computer Graphics, IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Image Processing, Signal Processing: Image Communication, and The Visual Computer.