Deep Novel View Synthesis from Colored 3D Point Clouds

Zhenbo Song[1], Wayne Chen[2], Dylan Campbell[2,3], and Hongdong Li[2,3]

[1] Nanjing University of Science and Technology, China
[2] ANU -Australian National University
[3] Australian Centre for Robotic Vision

Abstract. We propose a new deep neural network which takes a colored 3D point cloud of a scene as input, and synthesizes a photo-realistic image from a novel viewpoint. Key contributions of this work include a deep point feature extraction module, an image synthesis module, and a refinement module. Our PointEncoder network extracts discriminative features from the point cloud that contain both local and global contextual information about the scene. Next, the multi-level point features are aggregated to form multi-layer feature maps, which are subsequently fed into an ImageDecoder network to generate a synthetic RGB image. Finally, the output of the ImageDecoder network is refined using a RefineNet module, providing finer details and suppressing unwanted visual artifacts. W rotate and translate the 3D point cloud in order to synthesize new images from a novel perspective. We conduct numerous experiments on public datasets to validate the method in terms of quality of the synthesized views.

Keywords: Image synthesis · 3D point clouds · Virtual views

1 Introduction

This paper addresses the problem of how to render a dense photo-realistic RGB image of a static 3D scene from a novel viewpoint, only based on a set of sparse colored point clouds depicting the scene. The rendering pipeline is illustrated in Figure 1. Traditional methods are often based on fitting point clouds to a piecewise smooth mesh surface; they however suffer from the need of a strong scene prior and large amount of computation, despite which they can fail when the point clouds are too sparse or contain gross outliers, as is typical for real-world 3D range scans.

Practical uses of novel view synthesis include generating photo-realistic views from a real physical scene for immersive Augmented Reality applications. Structure from Motion (SfM) techniques have been applied to reconstruct 3D models depicting the real scene. This way, the 3D models are represented as sparse set of 3D point clouds which are both computation and memory efficient, but falls short in visual appearance due to the very low density in discrete point samplings. This has motivated this paper to develop an efficient new method of dense novel view synthesis *directly* from sparse set of colored 3D point clouds.

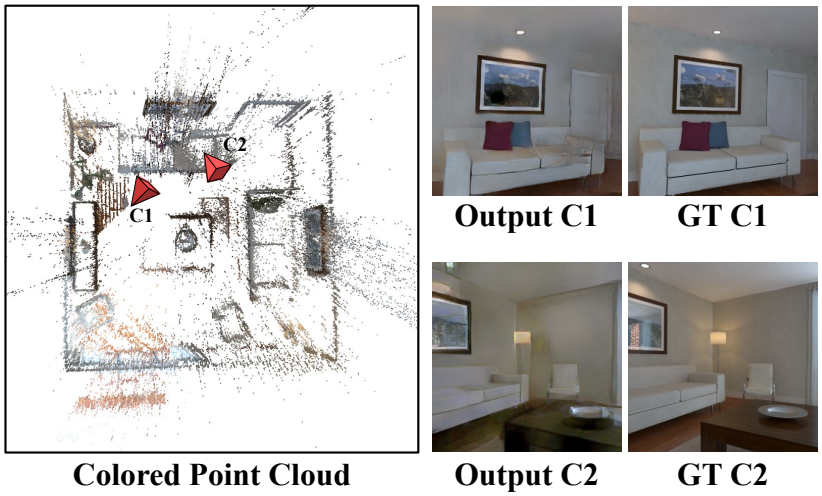

Fig. 1. Synthesizing novel views from colored 3D point clouds. The colored point cloud is generated from key frames of a video sequence using DSO [7]. Given two specific viewpoints C1 and C2 in the point cloud, our method synthesizes RGB images 'Output C1' and 'Output C2'. The corresponding ground-truth RGB images are labeled 'GT C1' and 'GT C2'.

The most related recent work to this paper is *invsfm* [28], where the authors proposed a cascade of three U-Nets [17] to reveal scenes from SfM models. The input to their network is a sparse depth image with optional color and SIFT descriptors, that is, a projection of the SfM point cloud from a specific viewpoint. Their synthesized images are fairly convincing, but their pipeline does not take full advantage of the available 3D information. Projected point clouds lose adjacency information, with convolutions only sharing information between points that project nearby on the image. In contrast, the original point clouds retain this structural information, and convolutions share information between points that are nearby in 3D space. Moreover, a network trained on point cloud data is able to reason more intelligently about occlusion than one that takes a lossy z-buffering approach. Recently, point cloud processing has advanced considerably, with the development of PointNet [29] and PointNet++ [30] stimulating the field and leading to solid improvements in point cloud classification and segmentation. Additionally, generative adversarial networks (GAN) [10] have demonstrated the power of generating realistic images. Synthesizing both developments, *pc2px* [2] trained an image generator conditioned on the feature code of a object-level point cloud to render novel view images of the object. So far, directly generating images from scene-level point clouds remains an under-explored research area.

Inspired by these works, we develop a new type of U-Net, which encodes point clouds directly and decodes to 2D image space. We refer to the encoder as *PointEncoder* and the decoder as *ImageDecoder*. The main motivation for the design of this network is that we intend to make full use of all structural

information in the point clouds, especially in the local regions of each point. Ideally, the 3D point features should help to recover better shapes and sharper edges in images. Meanwhile, we also use the associated RGB values for each point to enrich the 3D features with textural information. Consequently, our network is trained to generate RGB images from sparse colored point clouds. We further propose a network to refine the generated images and remove artifacts, called *RefineNet*. In summary, our contributions are:

1. a new image synthesis pipeline that generates images from novel viewpoints, given a sparse colored point cloud as input;
2. an encoder–decoder architecture that encodes 3D point clouds and decodes 2D images; and
3. a refinement network that improves the visual quality of the synthesized images.

Our approach generalizes effectively to a range of different real-world datasets, with good color consistency and shape recovery. We outperform the state-of-the-art method *invsfm* in two ways. Firstly, our network achieves better quantitative results, even with fewer points as input, for the scene revealing and novel view synthesis tasks. Secondly, our network has better qualitative visual results, with sharper edges and more complete shapes.

2 Related Work

There are two types of approaches for generating images from sparse point clouds: rendering after building a dense 3D model by point cloud upsampling or surface reconstruction; or directly recovering images using deep learning. In this section, we first review existing works on dense 3D model reconstruction and learning based image recovery. Then broader related topics are discussed such as novel view synthesis and image-to-image translation.

Dense 3D Model Reconstruction. Existing methods for building a dense 3D model from a point cloud can be grouped into two categories: point cloud upsampling, and surface reconstruction. PU-Net [38] and PU-GAN [20] are two deep learning based point cloud upsampling techniques. In these works, multi-level features for each point are learnt via deep neural networks. These features are further expanded in feature space and then split into a multitude of features to reconstruct a richer point cloud. Nevertheless, the upsampled point cloud is still not dense enough to enable image rendering. For mesh reconstruction, traditional algorithms often need strong priors including volumetric smoothing, structural repetition, part composition, and polygonal surface fitting [3]. Recently, some deep learning methods have been developed to address this problem. A 3D convolutional network called PointGrid, proposed by Le *et al.* [19], learns local approximation functions that help to reconstruct local shapes with better detail. However, reconstruction and storage of dense 3D models are not computationally efficient for practical applications.

Learning-Based Image Recovery. Instead of reconstructing the entire dense 3D model, some works synthesize images directly from sparse point clouds. A conditional GAN developed by Atienza [2] generates images from a deep point cloud code along with angles of camera viewpoints. Although the result does not outperform the state of the art, it shows more robustness towards downsampling and noise. Similarly, Milz *et al.* [11] adopt a GAN that conditions on an image projection. However, these two methods only work on object-level point clouds. In contrast, Pittaluga *et al.* [28] proposed a three stage neural network which recovers the source images of a SfM point cloud scene. The input to their network is a sparse depth image, that is, the projection of the point cloud onto the image plane with depth, color and a SIFT descriptor associated with each sparse 2D point. In contrast to these approaches, we focus on extracting the structural features of point clouds in 3D space and use them to generate better images.

Warping based Novel View Synthesis. Novel view synthesis from single or multiple images often requires a warping process to obtain a candidate image. Depth prediction is a typical strategy for warping. Liu *et al.* [21] regress pixel-wise depth and surface normal, then obtain the candidate image by warping with multiple surface homographies. Niklaus *et al.* [27] introduce a framework that inpaints the RGB image and depth map from a warped image so as to maintain space consistency. To achieve better depth estimation, multi-view images are applied in many methods, such as the use of multi-plane images (MPI) by Zhou *et al.* [39], and estimated depth volumes by Choi *et al.* [5] that leverage estimates of depth uncertainty from multiple views. The warped images using predicted depth maps often have only a few holes and missing pixels, which can be estimated using image completion networks. In comparison, our problem has much sparser inputs with significantly more missing data.

Image-to-Image Translation. Various methods [13, 40, 22] have succeeded in generating images from structural edges, changing the appearance style of existing images and synthesizing images from sketches. In our work, similar elements to these methods are used, such as an encoder–decoder architecture and adversarial training, for the task of pointset-to-image translation.

3 Method

Given a colored point cloud $\mathbf{P} \in \mathbb{R}^{N\times 6}$, where N is the number of points and each point has x, y, z coordinates and r, g, b color intensities, our goal is to generate an RGB image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$ captured by a virtual camera from a specific viewpoint. The viewpoint is defined by the camera extrinsic parameters $\mathbf{T} \in SE(3)$ and the intrinsic parameters of typical pinhole camera model $\mathbf{K} \in \mathbb{R}^{3\times 3}$. As shown in Figure 2, our proposed view synthesis network has three main components: a PointEncoder, an ImageDecoder and a RefineNet. The first two networks together are the coarse image generator G_c and the RefineNet is the refined image generator G_r. We train a cascade of these two generators for pointset-to-image reconstruction and refinement, with an adversarial training strategy [10] using

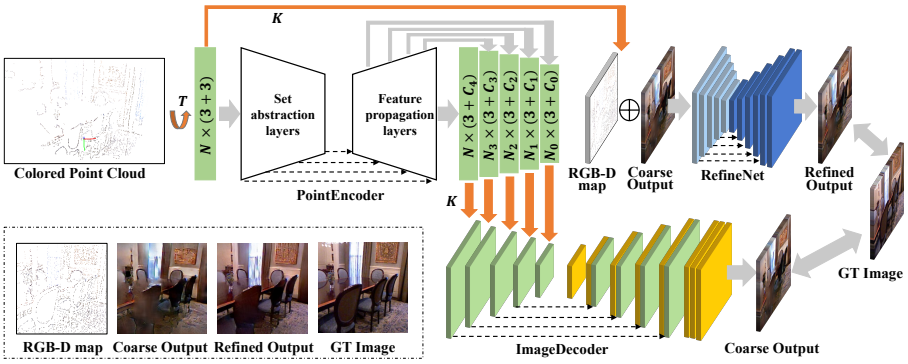

Fig. 2. Network architecture. The network has three modules with learnable parameters: a PointEncoder, an ImageDecoder, and a RefineNet. The PointEncoder has a PointNet++ structure [30] with set abstraction layers and feature propagation layers with skip connections. The ImageDecoder has a U-Net structure [17] but directly uses the projection maps from the PointEncoder. The RefineNet is a standard U-Net with an encoder–decoder structure which takes the coarse output from the ImageDecoder as input, alongside an additional RGB-D map. Visualizations of the different intermediate outputs are included at the bottom left.

the discriminators D_c and D_r respectively. These discriminators use the PatchGAN [14] architecture and instance normalization [35] across all layers.

For the forward pass, the point cloud $\mathbf{P}$ is first rigidly transformed to $\mathbf{P}'$ by applying $\mathbf{T}$. The PointEncoder takes $\mathbf{P}'$ as an input to extract a set of point features in 3D space. These features are then associated onto feature map planes by projecting corresponding 3D points with the camera intrinsics $\mathbf{K}$. The ImageDecoder translates these feature maps into the image domain and produces a coarse RGB image of the final output size. Finally, the RefineNet produces a refined image using an encoder-decoder scheme, given the coarse image and an additional sparse RGB-D map.

3.1 Architecture

PointEncoder. Since point clouds are often sparse and the geometry and topology of the complete scene is unknown, it is difficult to generate photo-realistic images by rendering such point clouds. Thus to synthesize high-quality images, as much implicit structural information should be extracted from the point cloud as possible, such as surface normals, local connectivity, and color distribution. Intending to capture these structures and context, we use the PointNet++ [30] architecture to learn features for each point in 3D space. Consequently, our PointEncoder is composed of four set abstraction levels and four feature propagation levels to learn both local and global point features. Set abstraction layers generate local features by progressively downsampling and grouping the point cloud.

Then feature propagation layers apply a distance-based feature interpolation and skip connection strategy to obtain point features for all original points.

The input to the PointEncoder is an $N \times (3+3)$ dimensional tensor consisting of 3D coordinates and RGB color intensities. After passing through the PointEncoder, each point has a C-dimensional feature vector. In order to use more point features, we save the features after each propagation level to construct a set of sub-pointsets with associated multi-scale point features. Specifically, after the i-th propagation level, we extract the point features $\mathbf{F}_i \in \mathbb{R}^{N_i \times (3+C_i)}$ of N_i subsampled points with 3D coordinates and C_i-dimensional feature channels. The final point feature set we adopt is denoted as $\mathbf{F} = \{\mathbf{F}_0, ..., \mathbf{F}_k\}$, where $N_k = N$ and $C_k = C$, (N_i, N_{i+1}) and (C_i, C_{i+1}) follow the rules of $N_i <= N_{i+1}$ and $C_i >= C_{i+1}$ respectively. Afterwards, $\mathbf{F}$ is projected and associated onto feature maps for the next step.

ImageDecoder. To decode the point features into an image, a bridge must be built between features in 3D space and features in image space. Considering the extraction process of point feature set $\mathbf{F}$, we observe that each 3D point in a sub-pointset represents a larger region in the original point cloud as the number of points in the subset gets smaller. As a result, feature vectors from smaller subsets contain richer contextual information than features from larger subsets. This is similar to how feature maps with less resolution but more channels in a convolutional neural network (CNN) encode information from a larger number of pixels. For the purpose of maintaining the scale consistency between the 3D space and image space, we project the point features to feature map planes with different resolutions according to their sub-pointset size. The ImageDecoder employs these feature maps and performs an upsampling and skip connection scheme like U-Net [31] until getting an image of the final output size.

More concretely, we project the point feature set $\mathbf{F}$ onto feature map planes $\mathbf{M} = \{\mathbf{M}_0, ..., \mathbf{M}_k\}$, where $\mathbf{M}_i \in \mathbb{R}^{H_i \times W_i \times C_i}$ corresponds to $\mathbf{F}_i$ and $\mathbf{M}_k$ has size $H \times W \times C$. Here, the generated feature maps $\mathbf{M}$ are regarded as a feature pyramid with the spatial dimension of its feature maps increasing by a scale of 2, as $H_{i+1} = 2H_i$ and $W_{i+1} = 2W_i$. In order to get the feature map $\mathbf{M}_i$, pixel coordinates in a map with size $H \times W$ are first calculated for all 3D points in $\mathbf{F}_i$ by perspective projection with camera intrinsics $\mathbf{K}$. After that, these pixel coordinates are rescaled in line with the size of $\mathbf{M}_i$ to associate the point features with it. If multiple points project to the same pixel, we retain the point closest to the camera. The ImageDecoder takes the feature pyramid $\mathbf{M}$ and decodes it into an RGB image.

RefineNet. By this stage in the network, we have generated an image from point features. However this is a coarse result and some problems remain unsolved. One issue is that many 3D points are occluded by foreground surfaces in reality but still project onto the image plane even with z-buffering due to the sparsity of the point cloud. This brings deleterious features from non-visible regions of the point cloud onto the image plane. In addition, the PointEncoder predominately learns to reason about local shapes and structures, and so the

color information is weakened. Accordingly, we propose the RefineNet module to estimate visibility implicitly and re-introduce the sparse color information.

As a standard U-Net architecture, the RefineNet receives a feature map of size $H \times W \times 7$, a concatenation of the coarse decoded image, the sparse RGB map, and the sparse depth map. The latter is used to analyse visibility in many geometric methods [4, 1]. The sparse RGB-D image is obtained by associating the RGB values and z-value of the original point cloud onto a map of size $H \times W \times 4$ using the same projecting rules in the ImageDecoder. The output of the RefineNet is an RGB image with higher quality than the coarse image.

3.2 Training Loss

We employ Xavier initialization and then separately train the network in two independent adversarial steps. Firstly, the coarse generator G_c (the PointEncoder and ImageDecoder) is trained to generate coarse RGB images using ground-truth image supervision. After that, the parameters of G_c are fixed and the refined generator G_r (the RefineNet) is trained to refine the coarse images. Since the same loss function and ground truth supervision are utilized for both steps, we represent G_c and G_r together as G and discriminators D_c and D_r as D to simplify notation in next paragraphs. We notate the input for each step as x, which is a colored point cloud of size $N \times 6$ for G_c and a feature map of size $H \times W \times 7$ for G_r. The generator and discriminator G and D represent the functions that for G map from $\mathbb{R}^{N\times 6} \to \mathbb{R}^{H\times W\times 3}$ (or $\mathbb{R}^{H\times W\times 7} \to \mathbb{R}^{H\times W\times 3}$), and for D map from $\mathbb{R}^{H\times W\times 3} \to \mathbb{R}$.

For each step, the network is trained over a joint objective comprised of an ℓ_1 loss, an adversarial loss and a perceptual loss. Given the ground-truth image $\mathbf{I}_{\text{gt}} \in \mathbb{R}^{H\times W\times 3}$, the ℓ_1 loss and the adversarial loss are defined as

$$\mathcal{L}_{\ell_1} = ||\mathbf{I}_{\text{gt}} - G(x)||_1 \tag{1}$$

$$\mathcal{L}_{\text{adv}} = \log[D(\mathbf{I}_{\text{gt}})] + \log[1 - D(G(x))]. \tag{2}$$

A perceptual loss [8, 15] is also used, which measures high-level perceptual and semantic distances between images. In our experiments, we use a feature reconstruction loss $\mathcal{L}_{\text{feat}}$ and a style loss $\mathcal{L}_{\text{style}}$ computed over different activation maps of the VGG-19 network [34] pre-trained on the ImageNet dataset [6]. The VGG-19 model is denoted as ϕ and the perceptual loss is computed using

$$\mathcal{L}_{\text{feat}} = \sum_{i=1}^{5} ||\phi_i(\mathbf{I}_{\text{gt}}) - \phi_i(G(x))||_1 \tag{3}$$

$$\mathcal{L}_{\text{style}} = \sum_{j=1}^{4} ||G_j^{\phi}(\mathbf{I}_{\text{gt}}) - G_j^{\phi}(G(x))||_1 \tag{4}$$

where ϕ_i generates the feature map after layers `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1`, `relu5_1`; G_j^{ϕ} is a Gram matrix constructed from the feature map that

is generated by ϕ_j, and ϕ_j corresponds to layers `relu2_2`, `relu3_4`, `relu4_4`, `relu5_2`. The Gram matrix treats each grid location of a feature map independently and captures information about relations between features themselves. While $\mathcal{L}_{\text{feat}}$ helps to preserve image content and overall spatial structure, $\mathcal{L}_{\text{style}}$ preserves stylistic features from the target image.

We manually set four hyperparameters as the coefficients of each loss term, and thus our overall loss function is given as follows

$$\mathcal{L}_G = \lambda_{\ell_1}\mathcal{L}_{\ell_1} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} + \lambda_{\text{style}}\mathcal{L}_{\text{style}}. \tag{5}$$

During training, the generator and discriminator are optimized together by applying alternating gradient updates.

4 Experiments

We evaluate our approach on several different datasets, including indoor and outdoor scenes, and on several different sources of 3D data. Specifically, we train our model on the SUN3D [37] dataset and then test it on two other indoor datasets, NYU-V2 [25] and ICL-NUIM [12], as well as the outdoor KITTI odometry dataset [9]. We also explore point clouds generated from different sources: depth measurements, COLMAP [32] and DSO [7]. We first compare the cascaded outputs of our proposed network, from the coarse and fine generators. Then we compare our approach with the state-of-the-art inverse SfM method [28], denoted *invsfm*, in terms of the synthesized image quality. We refer to the task of recovering views that were used to generate the input point clouds as *scene revealing*, and the task of recovering new views as *novel view synthesis*. Furthermore, to demonstrate the generalizability of our method, results on the KITTI dataset are reported, using point clouds generated by LiDAR sensors.

Training Data Preprocessing. SUN3D is a dataset of reconstructed spaces and provides RGB-D images and the ground-truth pose of each frame. By sampling from an RGB-D image, we can obtain a colored point cloud, which can be transformed to a novel view. Accordingly, we prepare the training data as a pair of RGB-D images and their relative pose, and then train our network with both current view inputs and novel view inputs. We use re-organized pairs of SUN3D data [36] to form a current-pointset–current-image pair and a current-pointset–novel-image pair for augmentation. In order to sample a sparse point cloud, we first sample 4096 pixels on each RGB image including feature points (ORB [24] or SIFT [23]), image edges and randomly sampled points. Then these pixels are inversely projected as a 3D point cloud, using the depth map and camera intrinsics, resulting in a colored point cloud.

Testing Data Preprocessing. We prepare two different types of point clouds for the evaluation of the scene revealing and novel view synthesis tasks. These two tasks have a significant difference: the scene revealing task intends to recover source images that participated in the generation of input pointsets, while the novel view synthesis task requires input pointsets generated from new views. As

invsfm is the closest work to ours, for the scene revealing task we test our trained model on the SfM dataset they provide, which is processed from the NYU-V2 dataset[33] using COLMAP. As is typical for visual odometry or SLAM systems, 3D points are only triangulated from key frames. Therefore, we can evaluate the quality of novel view synthesis by using the remaining frames and the SfM pointset. In our experiment, we utilized DSO on the ICL-NUIM dataset to obtain pointsets and estimate the results for novel view synthesis. For unifying the size of input pointsets to $n \times 6$, we apply a sampling technique: randomly sampling when more than n points are in the field of view; and using nearest neighbor upsampling when there are fewer than n points.

Implementation Details. Our network is implemented using PyTorch and is trained with point clouds of size 4096×6 and images of size 256×256 using the Adam optimizer [16]. Since RefineNet is designed to perform image inpainting given a coarse input, we use the same empirical hyperparameter settings as EdgeConnect [26]: $\lambda_{\ell_1} = 1$, $\lambda_{\text{adv}} = \lambda_{\text{feat}} = 0.1$, and $\lambda_{\text{style}} = 250$. The learning rate of each generator starts at 10^{-4} and decreases to 10^{-6} during training until the objective converges. Discriminators are trained with a learning rate one tenth of the generators' rate.

Metrics. We measure the quality of the synthesized images using the following metrics: mean absolute error (MAE); structural similarity index (SSIM) with a window size of 11; and peak signal-to-noise ratio (PSNR). Here, a lower MAE and a higher SSIM or PSNR value indicates better results.

Runtime. The runtime for training on a single GTX 1080Ti GPU is 3d 21h 50min for 30K training examples and 50 epochs. For inference on a single TITAN XP GPU, the inference time is 0.038s to synthesize a 256×256 image from an $N = 4096$ point cloud. In comparison, *invsfm* takes 0.068s, almost double our inference time. For the PointEncoder/ImageDecoder/RefineNet, the inference time is divided up as 0.015/0.018/0.005s.

4.1 Cascaded Outputs Comparison

In Figure 3 we qualitatively compare the coarse and refined outputs of our two step generators where the size of input point clouds are all sampled to 4096. While the coarse results have good shape and patch reconstruction fidelity, the refined results recover colors and edges better. In addition, numerical comparison (Ours-coarse and Ours-refined) in Table 1 indicates that the RefineNet improves the results significantly. However, the performance of our coarse and refined outputs do not improve as the number of sampled points increases. The main reason is that there may not be that number of points in the field of view for many scenes and our upsampling strategy just replicates the points. Another reason is that we trained our model using 4096 points, thus the best performance is achieved when sampling the same number of points during testing. This reflects the capacity of our model for generating realistic images from very sparse pointsets. In our case, a 256×256 image is synthesized from only 4096 points, which is less than 6.25% of the pixels.

Fig. 3. Comparison of coarse and refined outputs. (Left to right) Input pointset, coarse output, refined output and ground-truth image. The input point clouds are sampled to a size of 4096. The coarse outputs reconstruct region shapes and patches while the refined outputs improve the color consistency and repair regions with artifacts.

4.2 Scene Revealing Evaluation

To evaluate scene revealing performance, we utilize pointsets obtained from SFM on the NYU-V2 dataset. We make qualitative comparison of our approach with *invsfm* in Figure 4 (first four columns), and additional results (last four columns) are reported using pointsets generated from RGB-D images. The results demonstrate that our work recovers sharper image edges and maintains better color consistency. With the 3D point features learnt by the PointEncoder, the network is able to generate more complete shapes, including small objects. In Table 1, quantitative results are given for comparison where our refined outputs achieve a notable improvement over *invsfm*. Even when using fewer input points, our approach has higher SSIM and PSNR scores as well as a lower MAE. It is also remarkable that our coarse results correspond closely to the results of *invsfm*, which reflects the effectiveness of our combination of the PointEncoder and ImageDecoder. Finally, the performance of our refined outputs remains stable with respect to the size of input pointsets, which indicates that our approach is robust to pointset density.

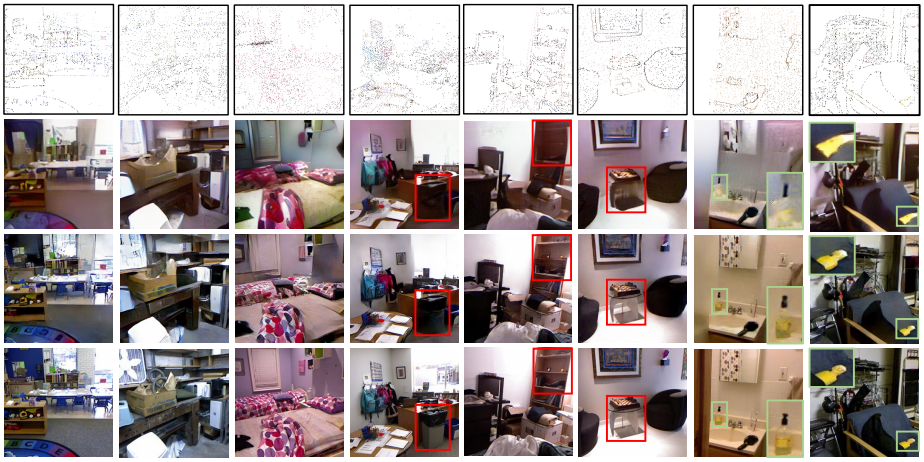

Fig. 4. Qualitative results for the scene revealing task on NYU-V2. (Top to bottom) Input pointset, *invsfm* results, our results, ground-truth images. Here our method uses 4096 sampled points while *invsfm* uses all points. The scenes are diverse and the point cloud sources differ: the first four are captured using SfM while the last four are sampled from RGB-D images. The first three columns show that our method generates sharper edges and better colors. Moreover, our results give better shape completion (red boxes) and finer small object reconstruction (green boxes).

Table 1. Quantitative results for the scene revealing task on NYU-V2. The second column 'Max Points' refers to the size of the input point clouds, where 4096, 8192 and 12288 mean that the point clouds were sampled to this size using the sampling strategy outlined in Section 4, and >20000 means that all points in the field of view were used. ↑ means that higher is better and ↓ means that lower is better.

	Max Points	**MAE ↓**	**PSNR ↑**	**SSIM ↑**
invsfm [28]	4096	0.156	14.178	0.513
	8192	0.151	14.459	0.538
	>20000	0.150	14.507	0.544
Ours-coarse	4096	0.154	14.275	0.414
	8192	0.155	14.211	0.435
	12288	0.164	13.670	0.408
Ours-refined	4096	**0.117**	**16.439**	0.566
	8192	0.119	16.266	**0.577**
	12288	0.125	16.011	0.567

4.3 Novel View Synthesis Evaluation

Since the input of the network is a 3D pointset, synthesizing novel views of scenes can easily be achieved. As mentioned, we tested our proposed method on the non-keyframes of the DSO output. Note that non-keyframes are all aligned to specific poses in the pointsets, which thus can be seen as novel viewpoints

Fig. 5. Qualitative results for the novel view synthesis task on ICL-NUIM. (Top to bottom) Input pointset, *invsfm* results, our results, ground truth images. Here 4096 points are sampled for our method while *invsfm* takes all points. Our method constructs images with better color consistency (first two columns), sharper edges (red box), and finer-detail for small objects (green box).

with respect to the keyframes. We report the results of our model along with *invsfm*. Neither model is trained or fine-tuned on this dataset to ensure fair comparison. Qualitative results are displayed in Figure 5 which shows that our model has advantages over *invsfm*. While the color effects of *invsfm* may partially fail in some cases, our approach recovers images with color consistency. The main characteristic of our model's ability to maintain shapes is also prominent here. Moreover, from the quantitative results in Table 2, we observe that our model outperforms *invsfm*, despite having fewer 3D points in the input pointset, by a greater margin than for the scene revealing task.

Table 2. Quantitative results for the novel view synthesis task on ICL-NUIM. Our method samples 4096 or 8192 3D points as input while *invsfm* takes all points in the view field. Our model achieves better results despite having many fewer input points.

	Max Points	**MAE ↓**	**PSNR ↑**	**SSIM ↑**
invsfm [28]	>20000	0.146	14.737	0.458
Ours-Coarse	4096	0.134	15.6	0.381
	8192	0.138	15.4	0.374
Ours-Refined	4096	**0.097**	**18.07**	0.579
	8192	0.101	17.75	**0.587**

Table 3. Quantitative results on KITTI. We compare the output for scene revealing and view synthesis tasks on KITTI. Note that we did not train on any outdoor datasets, but our model is still able to generalize reasonably well to this data.

Type	**MAE ↓**	**PSNR ↑**	**SSIM ↑**
Scene revealing	0.154	13.8	0.514
Novel view synthesis	0.165	12.8	0.340

4.4 Results on the KITTI dataset

The LiDAR sensor and camera on the KITTI car are synchronized and calibrated with each other. While LiDAR provides accurate measurements of the 3D space, the camera captures the color and texture of a scene. By projecting the 3D pointset onto image planes, we can obtain the RGB values of each 3D point. Since the KITTI dataset also gives relative poses between frames in a sequence, novel view synthesis evaluation may be done on such a dataset. Figure 6 illustrates qualitative results for the scene revealing and view synthesis tasks. Although our model was not trained or fine-tuned on this dataset (or any outdoor dataset), it presents plausible results in that image colors, edges and basic shapes of objects are reconstructed effectively.

5 Conclusion

From the reported results above, it is clear that our pipeline has improved the performance over *invsfm*. This suggested it is possible to bypass a depthmap inpainting stage as used in *invsfm*. One possible explanation is that convolutions performed on the projected depthmap only share information between points that project nearby on the image, whereas processing directly on the point clouds removes this bias, sharing information between points that are nearby in 3D space. This difference in what is considered "nearby" is critical when reasoning about the geometric structure of a scene. It also means that the network is able to reason more intelligently about occlusion, beyond just z-buffering points that share a pixel. Indeed, the projection approach destroys information when multiple points project to the same pixel.

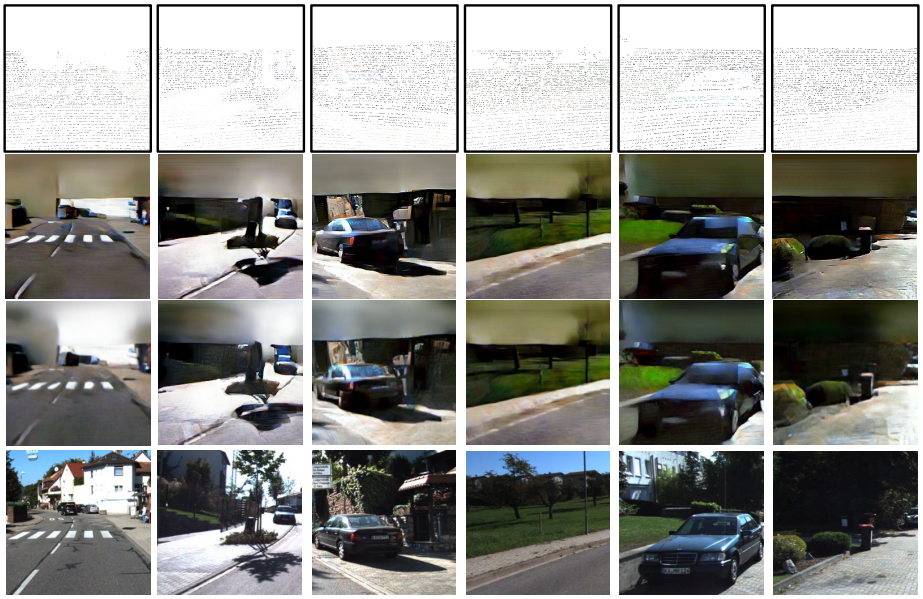

Fig. 6. Qualitative results on KITTI. (Top to bottom) Input pointset, scene revealing task results, novel view synthesis task results, ground truth images. The input pointsets are sampled to size 4096. Our model was not trained on any outdoor dataset, but still generates plausible images and recovers the shape of objects.

In this paper, we have demonstrated a deep learning solution to the view synthesis problem given a sparse colored 3D pointset as input. Our network is shown to perform satisfactorily in completing object shapes and reconstructing small objects, as well as maintaining color consistency. One limitation of the work is its sensitivity to outliers in the input pointset. Since outliers are common in many datasets, methods for filtering them from the point cloud could be investigated in future work, to improve the quality of the generated images. Our method assumes a static scene. A possible future extension is to synthesize novel views in a non-rigid dynamic scene [18].

Acknowledgements: This research was funded in part by the Australian Centre of Excellence for Robotic Vision (CE140100016), ARC-Discovery (DP 190102261) and ARC-LIEF (190100080) grants. The authors gratefully acknowledge GPUs donated by NVIDIA. We thank all anonymous reviewers and ACs for their comments." This work was completed when ZS was a visiting PhD student at ANU, and his visit was sponsored by the graduate school of Nanjing University of Science and Technology.

References

1. Alsadik, B., Gerke, M., Vosselman, G.: Visibility analysis of point cloud in close range photogrammetry. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences **2**(5), 9 (2014)
2. Atienza, R.: A conditional generative adversarial network for rendering point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 10–17 (2019)
3. Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Levine, J., Sharf, A., Silva, C.: State of the art in surface reconstruction from point clouds (04 2014)
4. Biasutti, P., Bugeau, A., Aujol, J.F., Brédif, M.: Visibility estimation in point clouds with variable density. In: International Conference on Computer Vision Theory and Applications (VISAPP). Proceedings of the 14th International Conference on Computer Vision Theory and Applications, Prague, Czech Republic (Feb 2019)
5. Choi, I., Gallo, O., Troccoli, A., Kim, M.H., Kautz, J.: Extreme view synthesis. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7781–7790 (2019)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255. Ieee (2009)
7. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE transactions on pattern analysis and machine intelligence **40**(3), 611–625 (2017)
8. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2414–2423 (2016)
9. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680 (2014)
11. Gross, H.M.: Points2pix: 3d point-cloud to image translation using conditional gans. In: Pattern Recognition: 41st DAGM German Conference, DAGM GCPR 2019, Dortmund, Germany, September 10–13, 2019, Proceedings. vol. 11824, p. 387. Springer Nature (2019)
12. Handa, A., Whelan, T., McDonald, J., Davison, A.J.: A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 1524–1531. IEEE (2014)
13. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. CoRR **abs/1611.07004** (2016), http://arxiv.org/abs/1611.07004
14. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1125–1134 (2017)
15. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. pp. 694–711. Springer (2016)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR) (May 2015)

17. Kohl, S., Romera-Paredes, B., Meyer, C., De Fauw, J., Ledsam, J.R., Maier-Hein, K., Eslami, S.A., Rezende, D.J., Ronneberger, O.: A probabilistic u-net for segmentation of ambiguous images. In: Advances in Neural Information Processing Systems. pp. 6965–6975 (2018)
18. Kumar, S., Dai, Y., Li, H.: Monocular dense 3d reconstruction of a complex dynamic scene from two perspective frames. In: International Conference on Computer Vision (2017)
19. Le, T., Duan, Y.: Pointgrid: A deep network for 3d shape understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9204–9214 (2018)
20. Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-gan: a point cloud upsampling adversarial network. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7203–7212 (2019)
21. Liu, M., He, X., Salzmann, M.: Geometry-aware deep network for single-image novel view synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4616–4624 (2018)
22. Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. In: Advances in Neural Information Processing Systems. pp. 469–477 (2016)
23. Lowe, D.G., et al.: Object recognition from local scale-invariant features. In: International Conference on Computer Vision. vol. 99, pp. 1150–1157 (1999)
24. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015)
25. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
26. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. CoRR **abs/1901.00212** (2019)
27. Niklaus, S., Mai, L., Yang, J., Liu, F.: 3d ken burns effect from a single image. ACM Transactions on Graphics (TOG) **38**(6), 184 (2019)
28. Pittaluga, F., Koppal, S.J., Kang, S.B., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 145–154 (2019)
29. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
30. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems. pp. 5099–5108 (2017)
31. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 234–241. Springer (2015)
32. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
33. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: European Conference on Computer Vision. pp. 746–760. Springer (2012)
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
35. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6924–6932 (2017)

36. Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., Brox, T.: Demon: Depth and motion network for learning monocular stereo. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017), http://lmb.informatik.uni-freiburg.de//Publications/2017/UZUMIDB17
37. Xiao, J., Owens, A., Torralba, A.: Sun3d: A database of big spaces reconstructed using sfm and object labels. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1625–1632 (2013)
38. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2790–2799 (2018)
39. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. In: SIGGRAPH (2018)
40. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2223–2232 (2017)