Multi-scene Representation Learning with Neural Radiance Fields

Bofeng Fu\textsuperscript{1,*}, Zheng Wang\textsuperscript{1,b}\textsuperscript{}

\textsuperscript{1}College of Intelligence and Computing, Tianjin University, Peiyang Park Campus: No.135 Yaguan Road, Haihe Education Park, Tianjin 300350, China

E-mail: \textsuperscript{*}2017218002@tju.edu.cn; \textsuperscript{b}wzheng@tju.edu.cn

Abstract. Getting representations of multiple objects or scenes is a raising research topic in Machine Learning (ML) community. Here, we propose a multi-scene representation model that can learn the representation of complex scenes and reconstruct them in high resolution given novel viewing directions. Our method represents a single scene with fully-connected layers. Each set of fully-connected layers are controlled by hyper-networks for multiple scenes modeling. For each scene, we take 3D coordinates ($x, y, z$) and 2D view-point orientations ($\theta, \phi$) as inputs. A set of fully-connected layers output volume density and RGB values at given 3D spatial positions. Then, we render the output volume density and RGB values along the camera rays into images using volume density rendering techniques. During training process, we optimize a continuous volume scene function with a small amount of input viewing directions. By designing versatile embedding module and multi-scene representation networks, our model can render photographic images with novel viewing directions for different complex scenes. Experiment results demonstrate the neural rendering and multi-scene representation abilities of our model. Several thorough experiments show that our method outperforms previous model on both reconstruction precision and scenes generation ability from novel viewing directions.

1. Introduction

In this paper, we focus on multi-scene representation without losing the information of details of the scenes. A good representation of complex scenes should preserve as many details as it can, and also can generate images from a given novel viewing direction. To tackle these problems, we refer to the neural rendering process proposed by NeRF\textsuperscript{[1]}. We represent static scenes as two sets of values, and implement a hyper-network structure to control the parameters of different layers in order to achieve multi-scene generation.

One latest research area in computer vision is getting representations of objects and scenes using Deep Neural Networks (DNNs). These methods can map a 3D dimension coordinates to an implicit encoding of the object\textsuperscript{[2]}. Similarly, some models also encode other graphics elements to low-dimensional coordinates using DNNs\textsuperscript{[3, 4, 5, 6]}. Another line of research focuses on the problem of representing high-resolution images with RGB values using volumetric representations. Previously, some volumetric approaches also tried to transform input images to voxel grids\textsuperscript{[7, 8, 9]}. Recently, some methods\textsuperscript{[10, 11, 12, 13]} train DNNs with datasets that contain a large amount of objects. A well-trained model can forecast a sampled volumetric representation of a group of input images\textsuperscript{[14]}. Scene Representation Networks (SRN)\textsuperscript{[15]} represent 3D scenes as multiple sphere surfaces, implicitly controlled by MLPs, which map every 3D coordinates...
to a feature vector. Here, we propose a model named Multi-scene Representation for Neural Rendering (MSRNR) that can learn the representation of complex scenes and generate images with novel viewing directions. In the experiment section, we test our model with different experiment settings and prove the outperformance of it. Our contributions are as following: (1) MSRNR can learn multiple complex scene representations at once. (2) MSRNR can generate photographic images given novel viewing directions. (3) Our method outperforms previous method under different experiment settings, demonstrating the high-resolution representation ability of our model.

2. Methodology

Our method represents scenes with volume density and RGB values. Each scene has various volume density and RGB values at different spatial locations. We use fully-connected layers to calculate the volume density and RGB values based on spatial locations and camera poses. For multi-scene representation learning, we further design hyper-networks that control the parameters of each fully-connected layer. Therefore, we train neural networks that have different sets of learnable parameters, which can map the location information to the volume density and RGB values. The overall structure of MSRNR is shown as figure1.

2.1. Multi-scene representation networks

Multi-scene representation networks take RGB images $I_i$, camera intrinsic matrix $K_{i}^{3 \times 3}$, and camera extrinsic matrix (pose) $E_{i}^{3 \times 4}$ as inputs. The output of multi-scene representation networks are view-point consistent volume density $\alpha$, and view-point independent RGB values $c$. We design a versatile embedding model that can deal with volume density and RGB values in different channels. The reason we build this structure is that volume density is consistent with viewing direction, while RGB values are independent with the view-point. Therefore, we design two hyper-networks to control parameters updating processes of two different parts. This kind of structure not only fits the physical definitions, but also disentangles the relationship between volume density networks and RGB values networks. After getting the volume density and RGB values for each spatial location of a certain complex scene, we render the reconstruction images using volume density rendering techniques.

* All layers are fully-connected layers.

Figure 1. The overall framework of MSRNR Model.
2.2. Overall processes
In this section we describe the overall processes of our model in details. MSRNR model learns the representation for different scenes through the following processes.

- **Ray generator.** First of all, we generate rays from a camera that pass through every pixel on the image plane. Then, we transform the rays from camera space to world space. The transforming process can be formulated as equation 1.

\[ r(d) = E^T(K^{-1}\{u, v, d\} - t), d > 0 \]  
where \( E \) and \( K \) are the transformation matrices described above. \( u, v, d \) are 3D coordinates of a given ray, respectively.

- **Versatile embedding.** Position embedding has huge effect on reconstruction results. According to the physics process of volume density rendering, we know that volume density is consistent with viewing direction, whereas RGB values are independent from viewpoints. Thus, we define two embedding processes. The first one takes position embedding, \( \text{embedding}_{\text{pos}} \), as input to get a feature vector of volume density \( f_\alpha \) at a certain coordinate, along with a 256 dimensions feature vector \( f \). The second one takes the concatenation of the feature vector \( f \) with direction embedding, \( \text{embedding}_{\text{dir}} \), and outputs a feature vector of RGB values \( f_{\text{rgb}} \). These two processes can be formulated as equation 2, 3.

\[ f_\alpha, f = \Phi_{6\times256}^{\alpha}(\text{embedding}_{\text{pos}}) \]  
\[ f_{\text{rgb}} = \Phi_{4\times256}^{rgb}([\langle d, w, t \rangle, f]) \]  

- **Generalizing across scenes.** Each scene has its own latent code vector \( z_j \) generated from \textit{nn.Embedding} network as we use PyTorch framework. This embedding network is optimized jointly with other parts of our model. We generate the parameters of multi-scene representation networks for each scene according to equation 4.

\[ \phi_j = \Psi(z_j) \]  

- **Raw images rendering.** We generate raw images based on the representation learned by multi-scene representation networks.

\[ \alpha = \text{ReLU}(G(f_\alpha)) \]  
\[ \text{RGB} = \text{Sigmoid}(G'(f_{\text{rgb}})) \]  
where \( G \) represents the fully-connected layers that output volume density. \( G' \) represents the dense layers that output RGB values. \( f_\alpha \) and \( f_{\text{rgb}} \) are the learned features of volume density and RGB values, respectively. We use \text{ReLU} activation function in our model to ensure that the output volume density is non-negative.

To train this model jointly, we simply compute MSE loss between the ground truth images and the reconstruction results. Because we incorporate an auto-encoder structure in our model, we also add L2-regulator term to the loss function. Thus, our training target can be formulated as equation 7.

\[ \arg\min_{\{\theta, \psi, \{z_j\}_{j=1}^M\}} \sum_{j=1}^M \sum_{i=1}^N \|\Theta_\theta(\Phi_\psi(z_j), E_i^j, K_i^j) - I_i^j\|^2_2 + \lambda_{\text{latent}} \|z_j\|^2_2 \]  
where \( \lambda_{\text{latent}} \) is a weighted hyper-parameter that balances the MSE loss term and the L2-regulator term.
3. Experiments
In the experiment section, we test multi-scene representation ability of our model, and conduct an ablation study. We compare the performance of our model with SRN[15] model during the experiments. We first describe the chosen dataset and the experiment settings.

3.1. Dataset description
We use Shapenet v2 cars[16] as our experiment dataset. This dataset contains about 2500 different car scenes. Each scene has some fine details that we want our model to capture. In our experiments, all images are normalized to 128 × 128 pixels. Camera poses are randomly selected on a sphere over cars, where the cars are located at the origins of spatial distributions.

3.2. Experiment settings
In order to test the multi-scene representation ability, we compare our model with SRN on Shapenet v2 cars dataset. We randomly select 500, 1000, and 2000 cars from the dataset. Each car scene has 50 different viewing directions. We use 42 of them in training process, and the rest of them in testing process. The batch size for all experiments is set to 16, and the maximum training steps is 200k. To compare scenes representation abilities of different models, we use mean square error (MSE) as metrics for reconstruction precision. Higher MSE values mean worse reconstruction performance. Thus, we expect our model to have lower MSE values.

3.3. Multi-scene representation ability
In this experiment, we demonstrate that MSRNR can learn different representations for complex scenes. As it shown in figure 2, we use 2000 instances scenario as an example to illustrate the reconstruction results of different models. From figure 2, we can see that MSRNR has better reconstruction results, which nearly indistinguishable from the ground truth images.

![Ground Truth](image1)

![MSRNR](image2)

![SRN](image3)

**Figure 2.** Ground truth images and reconstruction results of different models.

Now let’s take a deeper look at the reconstruction ability of MSRNR and SRN. In figure 3, the red line represents MSE changing for MSRNR model where 500 instances are used. The orange line and the navy blue line are for using 1000 and 2000 instances scenarios, respectively. The pink line represents MSE changing for SRN model where 500 instances are used. The green line and the blue line are for using 1000 and 2000 instances scenarios, respectively.

From figure 3, we can conclude that, under the same circumstance, MSRNR converges faster than SRN. MSRNR’s training process is more stable, which proves that volume rendering technique is efficient. For 200k iterations, SRN takes about 8 hours. However, MSRNR needs
about 48 hours. This means that ray-tracing methods are time consuming. Table 1 shows MSE values during testing process. Even though MSRNR consumes more time than SRN, MSRNR provides more realistic reconstruction results, which demonstrates the effectiveness of our model. MSRNR can learn to represent more fine details of the input images and generate images based on the learned 3D representations from novel viewpoints.

![Loss vs Iterations](image.png)

**Figure 3.** Training loss and testing loss.

| Table 1. Reconstruction precisions of two models. |
|-----------------------------------------------|
| Num of Instances | MSE for MSRNR | MSE for SRN |
|------------------|---------------|-------------|
| 500              | **15.51**     | 16.19       |
| 1000             | **26.5**      | 27.72       |
| 2000             | **29.20**     | 31.36       |

### 3.4. Ablation study
In this experiment, we test the ability of versatile embedding module. We randomly select 500 instances from Shapenet v2 cars dataset. We thoroughly analyze the ability of versatile position embedding by comparing our model with and without versatile position embedding. We also take the performance of NeRF-like position embedding into consideration. The results of ablation study are shown as Table 2.

| Table 2. Ablation study. |
|--------------------------|
| Num of Instances | Training MSE | Testing MSE |
|------------------|--------------|-------------|
| No position embedding | 14.35        | 18.03       |
| NeRF-like position embedding | 10.98        | 16.57       |
| Versatile position embedding | **7.33**     | **15.51**   |
From table 2, we can conclude that position embedding is of great importance for gathering information during image encoding process. This is because the position information has its intrinsic inner-relationship. This result also gives us some inspirations on designing meaningful features. A good embedding feature should contain not only the intuitive information, but also some human knowledge. Therefore, we can provide fruitful information for our model helping it to explore underlying patterns. Furthermore, versatile position embedding shows information capability since it has the lowest MSE values. This illustrates the effectiveness of such structure. By releasing the tangle between volume density and RGB values, neural networks can learn better representations of underlying distributions, which will further improve the performance.

4. Conclusion and Future work
In this paper, we propose a multi-scene representation model that can learn representations of complex scenes and reconstruct them in high resolution given novel viewing directions. By designing a versatile embedding module and multi-scene representation networks, our model can render photographic images with novel view-points for different complex scenes. Experiment results prove the effectiveness of these two components, and show that our method outperforms previous model on both reconstruction precision and novel view generation ability. However, our experiment results also show that ray-tracing method that we used in our model is time consuming. In our future work, we will focus on optimizing data flows during volume density computing process.

References
[1] Mildenhall B, Srinivasan P P, Tancik M, et al. Nerf: Representing scenes as neural radiance fields for view synthesis[J]. arXiv preprint arXiv:2003.08934, 2020.
[2] Curless B, Levoy M. A volumetric method for building complex models from range images[C]//Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. 1996: 303-312.
[3] Henzler P, Mitra N J, Ritschel T. Learning a Neural 3D Texture Space from 2D Exemplars[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 8356-8364.
[4] Oechsle M, Mescheder L, Niemeyer M, et al. Texture fields: Learning texture representations in function space[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 4531-4540.
[5] Rainer G, Ghosh A, Jakob W, et al. Unified neural encoding of btf[C]//Computer Graphics Forum. 2020, 39(2): 167-178.
[6] Rainer G, Jakob W, Ghosh A, et al. Neural btf compression and interpolation[C]//Computer Graphics Forum. 2019, 38(2): 235-244.
[7] Broadhurst A, Drummond T W, Cipolla R. A probabilistic framework for space carving[C]//Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. IEEE, 2001, 1: 388-393.
[8] Seitz S M, Dyer C R. Photorealistic scene reconstruction by voxel coloring[J]. International Journal of Computer Vision, 1999, 35(2): 151-173.
[9] Seitz S M, Curless B, Diebel J, et al. A comparison and evaluation of multi-view reconstruction algorithms[C]//2006 IEEE computer society conference on computer vision and pattern recognition (CVPR’06). IEEE, 2006, 1: 519-528.
[10] Henzler P, Rasche V, Ropinski T, et al. Single-image Tomography: 3D Volumes from 2D Cranial X-Rays[C]//Computer Graphics Forum. 2018, 37(2): 377-388.
[11] Yao Y, Luo Z, Li S, et al. Mvsnet: Depth inference for unstructured multi-view stereo[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 767-783.
[12] Penner E, Zhang L. Soft 3D reconstruction for view synthesis[J]. ACM Transactions on Graphics (TOG), 2017, 36(6): 1-11.
[13] Tucker R, Snavely N. Single-view view synthesis with multiplane images[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 551-560.
[14] Porter T, Duff T. Compositing digital images[C]//Proceedings of the 11th annual conference on Computer graphics and interactive techniques. 1984: 253-259.
[15] Sitzmann V, Zollhöfer M, Wetzstein G. Scene representation networks: Continuous 3d-structure-aware neural scene representations[C]//Advances in Neural Information Processing Systems. 2019: 1121-1132.
[16] Chang A X, Funkhouser T, Guibas L, et al. Shapenet: An information-rich 3d model repository[J]. arXiv preprint arXiv:1512.03012, 2015.