Eurographics Symposium on Geometry Processing 2020
Q. Huang and A. Jacobson
(Guest Editors)

Volume 39 (2020), Number 5

DFR: Differentiable Function Rendering for Learning 3D Generation from Images – Supplemental Material

Yunjie Wu[1] and Zhengxing Sun[1]

[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P R China

1. Introduction

This supplemental material contains three parts:

- Section 2 provides more details on implementation of all applications mentioned in the main paper.
- Section 3 provides details on implementation of rendering for normal map, as mentioned in the main paper.
- Section 4 shows more qualitative results.

2. Details on Implementation

In this section, we provide more details for the applications with DFR mentioned in the main paper.

2.1. Data Processing

For preparing the training data of single-image 3D reconstruction, we render training shapes in ShapeNet from 24 random views. The distance between the camera and the original point is 2.732, the same as NMR [KUH18] and SoftR [LLCL19]. The elevation's range is [-15, 45], and the azimuth's range is [0, 360]. The resolution of rendered images is 64. The format of the images is RGBA, where the alpha channel is used as silhouette supervision. During training, 2 of 24 views are selected to make up a single training sample. During test, only one view of the 24 is selected randomly. The train/test split is the same as Choy [CXG*16].

The training data of 3D GAN Learning is the same as above. For computing the fid, we generate 100 3D shapes randomly and render them from some random viewpoints. Then we also render 100 shapes from training data. We compute the fid from these two sets of rendered images.

For the image fusion application, we select the Stanford 3D Scanning Repository [LGCP05]. We select the widely used "Stanford Bunny" and "armadillo" for this task. We use the Blender [Com18] to render the two shapes in 24 random views. The distance of the camera to the original point is 2.732. The ranges of elevation and azimuth are [-30, 60] and [0, 360] separately.

2.2. Network Architecture for Imgae-fusion

In main paper we have presented detailed architecture of networks used in single-image 3D reconstruction and 3D GAN Learning.

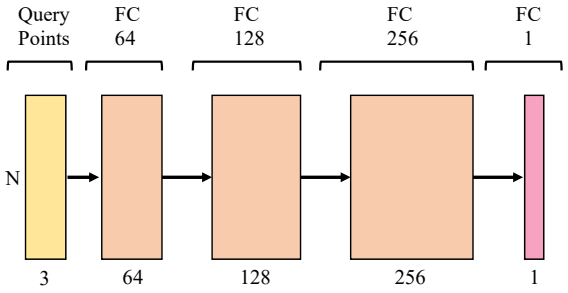

Figure 1: Network f in image fusion for 3D modeling. It takes 3D points' coordinates as input and predicts the function value for each point.

Here we present the network's structure for image fusion tasks. As it only need to represent single 3D object, the structure is much simpler and dosen't take the condition input. The network contains only a single decoder, which is consisted of four fully-connected layers as shown in Fig 1. The activation after each layer is ReLU.

2.3. Training Details

Single-image 3D Reconstruction. We train our network in a single GTX1080Ti GPU. The learning rate is decayed by rate 0.5 in 300 and 450 thousands of iterations. In practice, we find a model with the US sampling strategy can convergence faster than RS or SRS. To facilitate the training, in the first 200 thousands of iterations, we alternately employ the US and USS for every two iterations. After 20 thousands of iterations, we only adopt the SRS.

3D GAN Learning. The same as above, the learning rate is decayed by rate 0.5 in 300 and 450 thousands of iterations. For generating the random code z, we use a Gaussian distribution with mean = 0 and variance = 0.33. We find this can achieve a better performance compared with the variance = 1. Followed the WGAN [GAA*17], we employ a critic iterations strategy, which means train D for more iterations than D. We update D's parameters in every iteration and update the G's parameters in every three iterations.

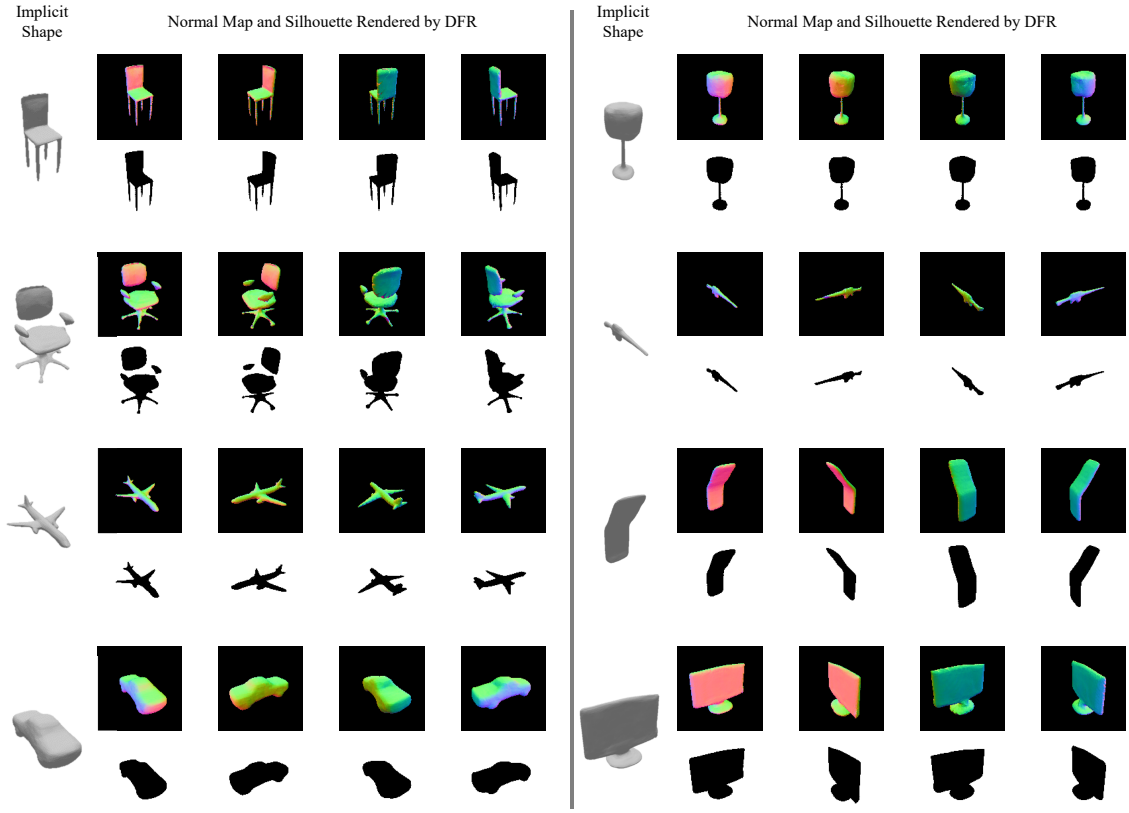

Figure 2: Examples of both normal map and silhouette rendering.

3. Rendering for Normal Map

In this section, we provide the details to extend our DFR to perform normal map's rendering, as mentioned in our main paper.

Rendering normal map requires the access to the surface normal of each Ray_i. As described in the main paper, the rays can be classified into hit rays and unhit rays. For the unhit rays, obviously no color should be shaded in the normal map. For a hit ray, we have already achieved an approximate surface point s_i^n along the ray. Then we need the normal direction of this surface point for rendering.

We can estimate the normal direction by sampling f at nearby points of s_i^n. It produces an estimation of the local surface curvature. Especially, let ε denotes a very small number, we sample the up, the left, and the front neighboring points of s_i^n:

$$\begin{cases} \text{left}_i = s_i^n + (\varepsilon, 0, 0) \\ up_i = s_i^n + (0, \varepsilon, 0) \\ \text{front}_i = s_i^n + (0, 0, \varepsilon) \end{cases} \tag{1}$$

We evaluate them via f and achieve the function values in the reforward process mentioned in the main paper. The the component of the normal direction in each axis can be estimated by computing the difference between the corresponding nearby point's values and the s_i^n's value:

$$n_i = \begin{bmatrix} f_c(left_i) - f_c(s_i^n) \\ f_c(up_i) - f_c(s_i^n) \\ f_c(front_i) - f_c(s_i^n) \end{bmatrix} \tag{2}$$

Then we normalize the n_i's length to 1:

$$\tilde{n}_i = \frac{n_i}{|n_i|} \tag{3}$$

As the ranges of $\tilde{n}_i$'s components are all $(-1,1)$, we perform a linear transformation on it, so that the ranges become $(0,1)$, which is consistent with the range of rgb values:

$$color_i = \tilde{n}_i \cdot 0.5 + 0.5 \tag{4}$$

the $color_i$ means the rgb value of the i-th pixel in a normal map. Note that this pixel is a foreground pixel (Ray_i is a hit ray), other wise its color is set to $(0,0,0)$.

We show some examples of the normal map rendering in Fig 2. As discussed in in the main paper, a small sampling number may

cause artifacts in the rendering. So we set the sampling number to 64 here.

4. More Qualitative Results

Here we provide more qualitative results of single-image 3D reconstruction and 3D GAN learning. We refer readers to our supplemental videos for a clearer effect.

4.1. single-image 3D reconstruction

Some additional results for single-image 3D reconstruction are shown in Fig 3, Fig 4 and Fig 5. It could be observed our method can handle with various structures of 3D shapes and produce visually satisfying meshes.

With the trained model, we are also able to perform shape interpolation with two input images. First, we extract the feature from images with trained encoder. Then interpolation operation is performed in the feature space. The interpolated features are fed into the decoder, producing interpolated shapes. We compare our method with the other state-of-the-art SoftR [LLCL19] and show some results in Fig 6. From the results, we can tell that, although two methods can both perform smooth interpolation and generate plausible shapes, our method is able to create more various topology (example 1, 2, 3) and more accurate surface (Lamp's base in example 4).

4.2. 3D GAN learning

Some additional results for 3D GAN learning are shown in Fig 7.

References

[Com18] COMMUNITY B. O.: *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org. 1

[CXG*16] CHOY C. B., XU D., GWAK J., CHEN K., SAVARESE S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision* (2016), Springer, pp. 628–644. 1

[GAA*17] GULRAJANI I., AHMED F., ARJOVSKY M., DUMOULIN V., COURVILLE A. C.: Improved training of wasserstein gans. In *Advances in neural information processing systems* (2017), pp. 5767–5777. 1

[KUH18] KATO H., USHIKU Y., HARADA T.: Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3907–3916. 1

[LGCP05] LEVOY M., GERTH J., CURLESS B., PULL K.: The stanford 3d scanning repository. *URL http://www-graphics. stanford. edu/data/3dscanrep 5* (2005). 1

[LLCL19] LIU S., LI T., CHEN W., LI H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2019). 1, 3

Figure 3: Results of single-image 3D reconstruction

Figure 4: Results of single-image 3D reconstruction

Figure 5: Results of single-image 3D reconstruction

Figure 6: Results of shape interpolation

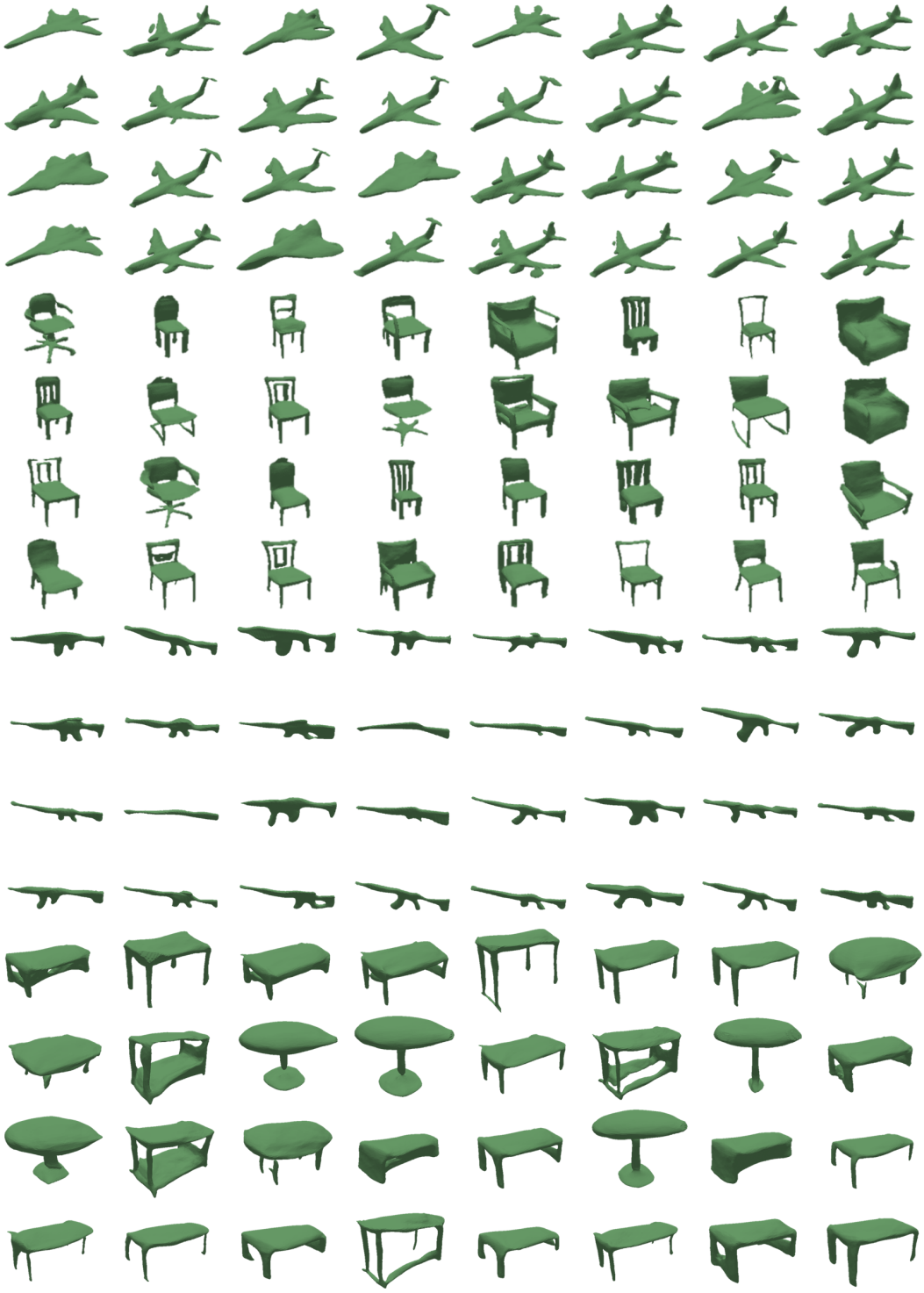

Figure 7: Results of 3D GAN learning