PBR Textures Capture by CNN Trained in Virtual 3D Scene

Chih Yang, Tzung-Han Lin

zemtice@gmail.com
National Taiwan University of Science and Technology, Taipei 10607, Taiwan
Keywords: Physically based rendering, BRDF, Deep learning, CNN

ABSTRACT

This study proposes a PBR (physically based rendering) texture capturing system using CNN (convolutional neural network). To obtain training data, we built a virtual capturing system in computer graphics environment. The final trained CNN was able to generate a set of PBR texture after inputting 4 photos of different lighting conditions.

1 Introduce

Simulation of physical reflection on an object surface plays a major role in CG (computer graphics) field. The BRDF (bidirectional reflectance distribution function) defines the reflectance of non-transparent objects at every viewing angle and light direction. For complex materials, such as woods, fabrics, rusty metal, etc., the BRDF would suffer from unequal estimation in different surface positions. Instead, SVBRDF (spatially varying BRDF) provides various reflection properties which are usually difficult to obtain. PBR involves complex reflection models and it comprises textures as well as base color image, normal map, and roughness map, etc., to achieve SVBRDF. Nowadays, the commonly used reflection model of SVBRDF is based on GGX model [1]. In most situations, this approximate method can generate realistic appearance as SVBRDF can do.

The traditional way to obtain the PBR textures in real world is calculating the reflection properties based on SVBRDF model. Ward [2] designed a gonio reflectometer to capture the BRDF of objects. It measured the reflection light intensity in every incident and observation directions. This procedure is time consuming and a gonio reflectometer is difficult to build in practice. Today, the learning-based method is used in many application fields with outstanding performance. There are several studies utilizing the deep learning method to capture PBR texture. For example, Deschaintre [3] used a cellphone to capture different light condition photos of an object and used CNN to generate the PBR Textures. But for obtaining training data, it is still a difficult issue particularly for capturing the PBR textures from the real world. In this study, we present a dataset generating method using computer graphics technique, and train a deep learning model with these data. The final result shows that it is able to capture PBR textures in the real world.

2 Experiment

The pipeline of our proposed PBR capturing system is shown in **Fig. 1**. The input data are 4 photos, which have different lighting directions from right, top, left and bottom directions. These 4 photos will be stacked along the third dimension as the input for next stage. The stacked images are then fed into the approximation process and U-net model, respectively. The final result is the sum of the approximation result and the output of U-net. The two-way process is a residual learning trick from the image transform concept of deep learning like style transfer and super resolution, etc. In other words, our model focuses on generating the residual of final the textures image, instead of the actual content of an entire image.

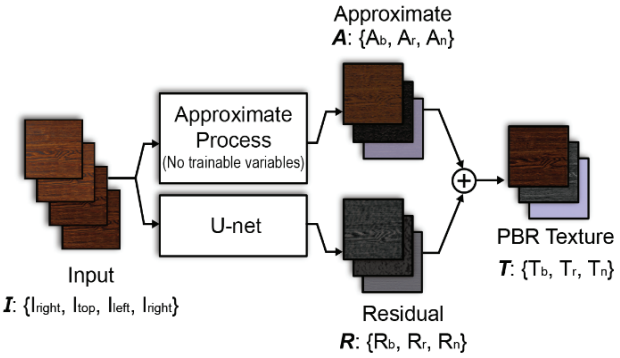

Fig. 1 The pipeline of our PBR texture generator

2.1 Dataset

We decide to train our model with three types of texture which are base color, roughness and normal map. These textures play a major role in realistic rendering. We collected 93 high resolution PBR textures (greater than 1024^2 pixels and under CC0 license) from the internet. To evaluate our model more comprehensively, 12 materials have been chosen as the test set, which contain the common materials in the real world, such as wood, metal, fabric, dirt and stone. We then crop each material image into 9 equal parts and resize them to square images of 256^2 pixels via Lanczos4 interpolation. Finally, we have 729 sets of data for training and 108 sets for testing.

To obtain the input of training data, we construct a virtual capturing system by using an open software named *Blender*. The layout of our configuration is shown

in **Fig. 2**.

The sample in **Fig. 2** is a square plane wrapped with PBR texture. The camera is set right above the sample and 4 light bulbs are surrounded the camera. The mask with a square hole is used to identify the specific measurement area.

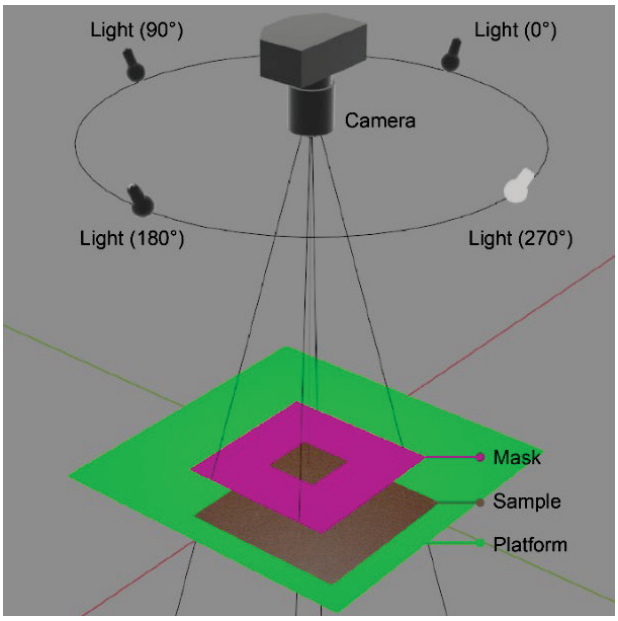

Fig. 2 The configuration of our virtual capturing system.

Fig. 3 shows one set of training data which are rendered by the virtual camera in **Fig. 2**. In **Fig. 3**, A, B, C and D are the images used as input; E, F and G are those as the ground truth of output. As a result, we can train our U-net to learn how the PBR texture is generated by these 4 lighting conditions photos.

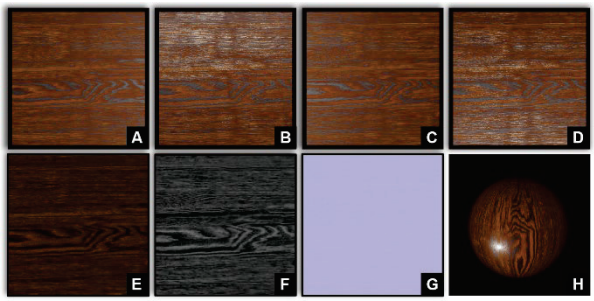

Fig. 3 One set of training data. (A), (B), (C) and (D): photos with light from 0˚, 90˚, 180˚, and 270˚; (E): base color map; (F): roughness map; (G): normal map, (H): render example using (E), (F) and (G).

2.2 Approximation Process

Three different textures have their own approximation method to generate “rough” textures. In this part, we will discuss how we implement each method. For clarity, we use I_r, I_t, I_l, and I_b to denote input images with light from right, top, left, and bottom, and I refers to the all 4 input images, A_b, A_r, and A_n as the rough texture of base color, roughness, and normal vector.

For base color approximation, we simply calculate the mean value of 4 input images by pixel, and use this mean image as the rough base color texture. The calculation is showed in **Eq. 1**.

$$A_b = \frac{I_r+I_t+I_l+I_b}{4} \qquad (1)$$

The roughness in CG is relative to the microstructure of an object surface. The bigger roughness means there is a lot of micro geometry in the surface. When a light ray hits a rough surface, it will bounce to every angle out from the reflection point. According to this phenomenon, if the range of lightness in 4 different lighting conditions is large, it means the reflected light can only be observed at specific viewing angle, and the roughness might be low. On the other hand, if the range of lightness is small, it means the reflection lights can be observed at many viewing angles, so the roughness is relatively high.

We use **Eq. 2** to express this relation between roughness and the variance of lightness of input images. This equation is not based on rigorous experiment, but for residual training purpose, this approximation is acceptable.

$$A_r = 1 - (1 - N_z) \cdot [max(L) - min(L)] \qquad (2)$$

L in **Eq. 2** is the 4 images with only lightness component, and the difference of maximum and minimum stands for the range of lightness each pixel. This range of lightness is inversely proportional to the roughness. The N_z is the z component of the normal vector at each pixel that has a range from 0 to 1. N_z can be obtained from **Eqs. 3-6**, which will be discussed later. According to Lambertian reflection, if the direction of the normal vector is close to the +z direction more, the lightness observed in same position with lights from same polar angle will have lower variance. So, we use *1 - N_z* as a weighting factor to fit this property.

In Lambertian reflection condition, if the light direction is closer to the normal vector of the point, the lightness would be higher. Based on the photometric method, we can use this phenomenon to calculate the x and y components by simply substrate lightness value of pixels on two horizontal images (I_r, I_l) and two vertical images (I_t, I_b). But we cannot use the same calculation to get the z component. We simply set the z component to the constant value 0.5, then scale the x, y and z values to be a normal vector. The whole calculation is shown in **Eqs. 3-6**.

$$N'_x = I_r - I_l \quad (3)$$

$$N'_y = I_t - I_b \quad (4)$$

$$N'_z = 0.5 \quad (5)$$

$$A_n = [N_x \quad N_y \quad N_z]^T = \frac{[N'_x \quad N'_y \quad N'_z]^T}{\left\|[N'_x \quad N'_y \quad N'_z]^T\right\|} \quad (6)$$

2.3 Architecture of model

Our model is based on U-net architecture which is Ronneberger's contribution [4], because it has the ability to generate image data with multiple levels of detail. U-net model has 2 paths – down sampling (hereinafter called DS) and up sampling (hereinafter called US). The input image data first are compressed to a low resolution feature map by DS process, then expanded to original resolution by US process. Our U-net contains 5 stages, the resolution of each stage is respectively 256^2, 128^2, 64^2, 32^2, and 32^2; and the depth of each stage is 64, 128, 256, 512, and 1024. Each stage has 3 Resnet blocks with batch normalization, and each mirror block pair in DS and US has skipped connection between them.

The role of DS is to extract the features of the input image. By contrast, US reconstructs the final output residuals of textures. If the extracted-features have enough information, all types of textures are able to be reconstructed by this features. So, we use the same US to extract the universal features and reconstruct the different types of texture by individual US. The benefit of this approach is that the size of the whole model is smaller than the one which has individual US and DS for different types of textures with not far-off performance in experiment. In addition, the time on training and inference are as less as a result of the less calculation.

2.4 Training processing

In training phase, we transformed the input data and the target base color texture into CIELAB color space, because minimizing the mean square error (MSE) of output and target data in CIELAB color space is equal to minimizing the CIELAB ΔE* between them, the final visual appearance might be much closer to the target compare to the one trained in sRGB color space. After the color space transformation, we mapped the L*, a* and b* values from [0, 255] to [-1, 1]. For the targets of roughness and normal textures, we simply mapped their range to [-1, 1].

We use the mean square error between the synthesized textures and target textures as the loss function, and the Adam optimizer has been chosen to train our mode. The β1 and β2 of Adam optimizer are set to 0.9 and 0.999, and the learning rate is set to 0.001.

3 Result

We had chosen 12 materials to evaluate our model. **Fig. 4** shows the PSNR values of the testing dataset. The yellow lines are for the approximated result, and the blue lines are for the final result. The averaged PSNRs are shown in Table 1. Over all, the final results which are augmented by our deep learning model are better than the approximated one.

Despite our model can generate fine base color and normal textures, it doesn't perform as well in generating roughness maps. The averaged PSNRs of final roughness textures are only reach 14.87, this causes huge visual difference to the ground truth roughness texture. But for rendering, the impact of roughness is not as much as base color or normal, so the rendering result may still look realistic.

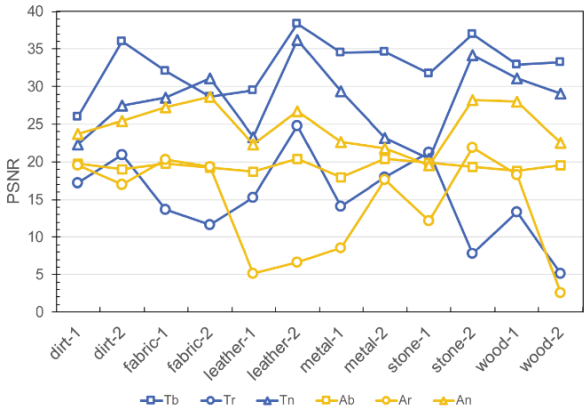

Fig. 4 PSNR of the 12 testing samples.

Table. 1 Averaged PSNRs of the 12 testing samples.

result \ type	Base color	Roughness	Normal
Approximate	19.42	14.00	24.74
Final	**32.86**	**14.87**	**28.02**

In addition, we captured photos of 6 materials in the real world to test the practicality of our deep learning model. The samples we chose is according to descending order, says 2 different fabrics, smooth and rough wood and 2 different stone look tiles, are shown with rendering result in **Fig. 5**.

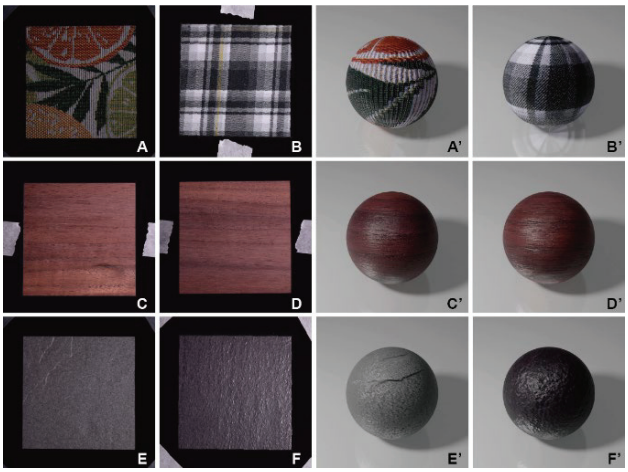

Fig. 5 Real photo testing. Left: the original photos with the right side lighting; right: render results of the synthesized textures.

The rendering results show that the synthetic textures produced by the proposed method are able to reproduce the visual appearance of these real world materials. But it still has insufficiency which should be improved in the future. For example, sample C and D are the different sides of the same piece of wood, but one had been fine sanded and waxed, another is wood with originally rough surfaces. The ideal roughness texture might reproduce this difference, but the synthesized textures seem unable to do the same effect.

The reason for getting inferior performance at roughness might be the lack of information caused by only using 4 orientations of lights on input data. We think this is because a complex material has various microstructures on different surface points, and that means more lights with different polar angles are required.

4 Conclusions

This study utilizes U-net deep learning model to generate PBR textures from photos with different lighting conditions. The current result shows that it has potential to generate accurate textures, such as layers of base color and normal.

Furthermore, we use these virtual training data to generate PBR textures. This procedure has 2 benefits. First, we can still obtain the training data with different system configurations. Secondly, we don't need to use a real instrument like a gonio reflectometer to capture the ground truth data. These features make our PBR texture generator much easier and more flexible to produce realistic effects.

Our model was trained using only 81 materials in this study. With the addition of more materials and lighting conditions, a more powerful and robust model can be trained. On the basis of this study, we are also developing a PBR texture capture system that does not require more than a single shot in the future.

References

[1] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, (2007). "Microfacet Models for Refraction through Rough Surfaces." *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, 195-206 (2007).

[2] G. J. Ward, "Measuring and modeling anisotropic reflection." *ACM SIGGRAPH Computer Graphics*, *26*(2), 265-272 (1992).

[3] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau, "Flexible SVBRDF Capture with a Multi-Image Deep Network." *Computer Graphics Forum*, *38*(4), 1-13 (2019).

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation." http://arxiv.org/abs/1505.04597 (2015)