Supplementary Document for
Inverse Rendering of Translucent Objects using Physical and Neural Renderers

A. Introduction

In this document, we provide the supplementary material of the proposed model. In Section B, we demonstrate the additional experiment results. In Section C, we introduce the proposed dataset. In Section D, we introduce our scene representation. Section E shows the training details of the proposed model. Section F demonstrates our network structure.

B. Additional Results

B.1. Real-world objects decomposition

We demonstrate the additional results of real-world translucent objects in Figure 1. From the figure, we can obseve that our model estimates reasonable subsurface scattering parameters (d) and the re-rendered images (i) look similar to the original input ones (a).

B.2. Parameter space exploring

In this section, we explore how well the neural renderer learned the subsurface scattering parameter space by two experiments. The first one is the linear interpolation of volumetric albedo α. Given a target volumetric albedo and an original one, we linearly interpolate them and input to our neural renderer. Figure 2 shows that the generated images are similar to their GTs. Second, we use the same method to edit the extinction coefficient σ_t from a extremely large value to a small value. We demonstrate the results in Figure 3.

B.3. Relighting of surface reflectance

Although relighting is not one of our target applications, it is easy to understand how well the normal and depth are estimated by showing the results under novel illumination coditions. We demonstrate them in Figure 4. It can be observed that although our model fails to capture some high frequency details of surface normals, it achieves overall good performance.

B.4. Comparison with the existing works

We compare our model with a surface reflectance only method that also uses a two-shot setup proposed by Boss *et al*. [1]. They use a novel stage-wise neural network for the shape and SVBRDF estimation. However, if we compare our model with the pure surface reflectance model, a problem arises. For a fair and equitable comparison, all models' training and testing data must be the same. However, training their model requires the GT value of diffuse albedo, which is not a part of the translucent object parameters. So we choose to use their own dataset to train the model and not compare diffuse albedo during testing. We illustrate the results of visual comparison of real-world translucent objects in Figure 5. Considering we do not have the same problem setting as their work, the result here is just for user reference.

C. Datasets

Measuring a large number of real-world translucent objects that include shape, surface reflectance, and subsurface scattering parameters is time-consuming. However, training data is an indispensable part of deep neural networks. Thus, we created a large-scale synthetic dataset by photo-realistic rendering. In the following few subsections, we discuss how we prepared the assets for rendering, including 3D objects, BSDF maps, subsurface scattering parameters, and illumination.

3D Objects Some existing works [1, 7, 9] use the Domain Randomized method to synthesize 3D objects by randomly assembling some simple shapes such as spheres, cylinders, cones, etc. For the shape complexity and diversity of our dataset, we collected human-created 3D objects from ShapeNet [2] and some other public resources. However, some objects in the ShapeNet have flipped surface normal, which can result in entirely black pixels when intersecting with light. We used a script to delete these objects, and finally, 5,847 3D objects were retained. All objects were scaled into a cube with a length of 50cm and placed at the origin and were randomly rotated, scaled, and translated during the rendering process. We use 5,000 objects for training and 857 for testing.

Roughness and auxiliary normal maps To make a meaningful surface reflectance pattern, we collected a large number of human-created surface reflectance maps from several open-source websites. Each of them contains a nor-

mal auxiliary map and a roughness map. We use a "smart uv mapping" function of Blender [5] to apply the collected roughness map and normal maps to objects. The normal map was applied on the original 3D object to modify its surface normal that makes more realistic surface. Before being applied to the objects, all roughness maps and normal maps were randomly resized. In the end, we achieved 2,745 surface reflectance maps and used 2,470 for training and 275 for testing.

IoR In order to reduce the ambiguity problem of inverse rendering, we set the index of refraction (IoR) to a constant (1.5046) and do not estimate it. We choose this value because many of our world's popular materials like glass (1.5046), amber (1.55), and polyethylene (1.49) have similar IORs.

Subsurface scattering parameters Follow the previous work [3], we randomly sampled the subsurface scattering parameters from a uniform distribution with extinction coefficient $\sigma_t \in [0, 32]$, volumetric albedo $\alpha \in [0.3, 0.95]$, and Henyey-Greenstein phase function parameter $g \in [0, 0.9]$.

Illumination To mimic the completed light condition in our real world, we choose environment maps as the light source. We used the Laval Indoor HDR dataset [4], which consists of 2,357 high-resolution indoor panoramas. During rendering, the pitch and roll were fixed, and they only rotated around the yaw axis. We computed a 3×9 Spherical Harmonics coefficients for each environment map to supervise our model. In total, 1500 environment maps were used for training, and 857 were used for testing.

Initially, we were going to use a point light to simulate the flashlight. However, for some very smooth objects, we observed severe noise when using a point light. Therefore, we used a tiny sphere area light with a radius of 10cm instead of a point light. The area light is placed 10cm behind the camera. Taking into account the different flashlight intensity of various devices in the real world, we also used a random radiance between 35 to 75 $\mathrm{W} \cdot \mathrm{m}^{-2}\mathrm{sr}^{-1}$.

Synthetic dataset For each scene, we used Mitsuba2 [8] to render five images: flash image, no-flash image, and three altered images. Each scene contains a randomly selected object, normal map, roughness map, environment map, and subsurface scattering parameters. The camera is placed 70cm away from the origin on the positive z-axis and looks at the origin. For the flash image, we used a small area light source behind the viewpoint to simulate the camera flashlight. For the no-flash image, we remove the small area light source and slightly change the camera's look-at direction to simulate the camera shake. For the three altered images, the subsurface scattering parameters were edited from those of the original flash image. In addition, we also rendered the ground truth depth, normal, roughness, and binary masks for the intermediate supervision. We obtained a total of 100,000 training scenes and 17,140 test scenes.

Real-world dataset For a more comprehensive test of our model, we also constructed a real-world dataset consisting of several common translucent objects. We took a flash photo and a no-flash photo with a smartphone camera. We manually created a binary mask for each object.

D. Scene Representation

In this section, we introduce the scene representation of our model. Figure 6 illustrates the differences between our scene representation and the existing works. The simplification of the scene representation can greatly reduce the problem of ambiguity, thereby reducing the difficulty of parameter estimation. However, at the same time, the scenarios in which the model can be applied are also reduced. For example, if we assume a diffuse reflectance like (a) in Figure 6, we can only solve the inverse rendering problem of objects like paper, rubber, etc. We make the first attempt at an inverse rendering problem involving both surface reflectance and subsurface scattering. Most translucent objects in our world like wax, plastic, and jade satisfy this scene representation.

For the surface part, we use a microfacet BSDF model proposed by [10]. Let p be a point at the object surface, the outgoing light radiance $L(w_o)$ of direction w_o at point p is defined by:

$$L(w_o) = \int_{\Omega} L(w_i) f_r(w_i, w_o, R(p), N(p)) \max(w_i \cdot N(p), 0)\, dw_i + \int_{\Omega'} L(w_i^{'}) f_t(w_i^{'}, w_o, R(p), N(p)) \max(w_i^{'} \cdot N(p), 0)\, dw_i^{'}, \tag{1}$$

where Ω is the hemisphere outside the object surface and $\Omega^{'}$ is the hemisphere inside the object. $L(w_i)$ stands for the incident light radiance that comes from outside of the object and $L(w_i^{'})$ is the incident light radiance that comes from inside of the object. f_r and f_t are the reflectance term and transmission term of the microfacet BSDF [10] respectively. R is the roughness map and N is the surface normal map. The radiance $L(w_i^{'})$ is the result of multiple scattering through the volume before going out of the surface at p. We use the Radiative Transport Equation as our homogeneous subsurface scattering model:

$$(w_o^{'} \cdot \nabla) L(w_o^{'}) = -\sigma_t L(w_o^{'}) + \sigma_s \int_{S^2} L(w_i^{'}) f_p(w_i^{'}, w_o^{'}, g)\, dw_i^{'}, \tag{2}$$

where $L(w_i^{'})$ and $L(w_o^{'})$ are the incident and outgoing light radiance respectively. The integral domain S^2 is a sphere.

σ_t is the extinction coefficient, σ_s is the scattering coefficient. f_p is the Henyey-Greenstein phase function, it has one parameter g which defines whether the scattering is forward ($g > 0$), backward ($g < 0$), or isotropic ($g = 0$):

$$f_p(\theta, g) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}}, \tag{3}$$

where θ is the angle between $w_i^{'}$ and $w_o^{'}$. In the main paper, we estimate the volumetric albedo α which is defined as:

$$\alpha = \sigma_s / \sigma_t. \tag{4}$$

E. Model Details

We assume that the size of input images is 256×256. All pixels and physical parameters including extinction coefficient σ_t, volumetric albedo α, phase function parameter g, flashlight intensity i are normalized to -1 to 1. The network parameters are initialized by Normal Initialization with the mean equal to 0 and variance equal to 0.02. During the training, we set batch size to 32. We train the model for 20 epochs by using Adam optimizer [6] with β_1 = 0.5, β_2 = 0.999, and learning rate lr = 0.0002 in the first 10 epochs and a linear decay in the remaining 10 epochs. We also use Batch Normalization to stabilize training. For the loss functions, we empirically set the weight of depth L_D to 5 and the others to 1.

F. Network Structure

We illustrate the detailed network structure in this section. The estimator contains a encoder and several heads. We show the structure of the proposed encoder in Table 1, Normal, Roughness, and Depth head in Table 2, Scattering and Illumination head in Table 3. Our neural renderer contains a Surface Encoder, a Scattering Encoder and a Decoder, we show the structure in Table 5, 4, and 6, respectively.

References

[1] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. Two-shot spatially-varying brdf and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3982–3991, 2020. 1, 7

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1

[3] Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. Towards learning-based inverse subsurface scattering. In *2020 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2020. 2

[4] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017. 2

[5] Roland Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.6*. Focal Press, 2010. 2

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[7] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 1

[8] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019. 2, 5, 7

[9] Shen Sang and Manmohan Chandraker. Single-shot neural relighting and svbrdf estimation. In *European Conference on Computer Vision*, pages 85–101. Springer, 2020. 1

[10] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. *Rendering techniques*, 2007:18th, 2007. 2

Table 1. Network structure of the proposed encoder. Numbers in the building blocks mean kernel size, number of output channel, and stride, respectively.

stage	building blocks	output size
input convolution	$\begin{bmatrix} 7\times 7 & 64 & 1\times 1 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$H \times W \times 64$
downsampling convolution 1	$\begin{bmatrix} 3\times 3 & 128 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{2} \times \frac{W}{2} \times 128$
downsampling convolution 2	$\begin{bmatrix} 3\times 3 & 256 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{4} \times \frac{W}{4} \times 256$
resnet blocks	$\begin{bmatrix} 3\times 3 & 256 & 1\times 1 \\ & \text{BN} & \\ & \text{ReLU} & \\ 3\times 3 & 256 & 1\times 1 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix} \times 9$	$\frac{H}{4} \times \frac{W}{4} \times 256$

Table 2. Network structure for the Normal, Roughness, and Depth head. Numbers in the building blocks mean kernel size, number of output channel, and stride, respectively. Output channel C for Normal head is 3, and for Roughness and Depth head is 1.

stage	building blocks	output size
upsampling convolution 1	$\begin{bmatrix} 3\times 3 & 128 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{2} \times \frac{W}{2} \times 128$
upsampling convolution 2	$\begin{bmatrix} 3\times 3 & 64 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$H \times W \times 64$
output convolution	$\begin{bmatrix} 7\times 7 & C & 1\times 1 \\ & \text{BN} & \end{bmatrix}$	$H \times W \times C$

Table 3. Network structure for the Scattering and Illumination head. Numbers in the building blocks mean kernel size, number of output channel, and stride, respectively. Output channel C for SSS head is 7, and for Illumination head is 28.

stage	building blocks	output size
upsampling convolution 1	$\begin{bmatrix} 3\times 3 & 128 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{2} \times \frac{W}{2} \times 128$
upsampling convolution 2	$\begin{bmatrix} 3\times 3 & 64 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$H \times W \times 64$
output linear	$\begin{bmatrix} C \\ \text{BN} \end{bmatrix}$	C

Figure 1. Inverse rendering results of real-world translucent objects. Our model takes a (a) flash image, a (b) no-flash image, and a (c) mask as input. Then, it decomposes them into (d) homogeneous subsurface scattering parameters (denoted as SSS in the figure), (e) surface normal, (f) depth, and (g) spatially-varying roughness. We use the predicted parameters to re-render the image that only considers (h) the surface reflectance, and (i) both surface reflectance and subsurface scattering. Note that (d) subsurface scattering only contains 7 parameters (3 for extinction coefficient σ_t, 3 for volumetric albedo α, 1 for phase function parameter g), we use these parameters to render a sphere using Mitsuba2 [8] for visualization. Brighter areas in the depth map represent a greater distance, and brighter areas in the roughness map represent a rougher surface.

Figure 2. Linear interpolation results of volumetric albedo α.

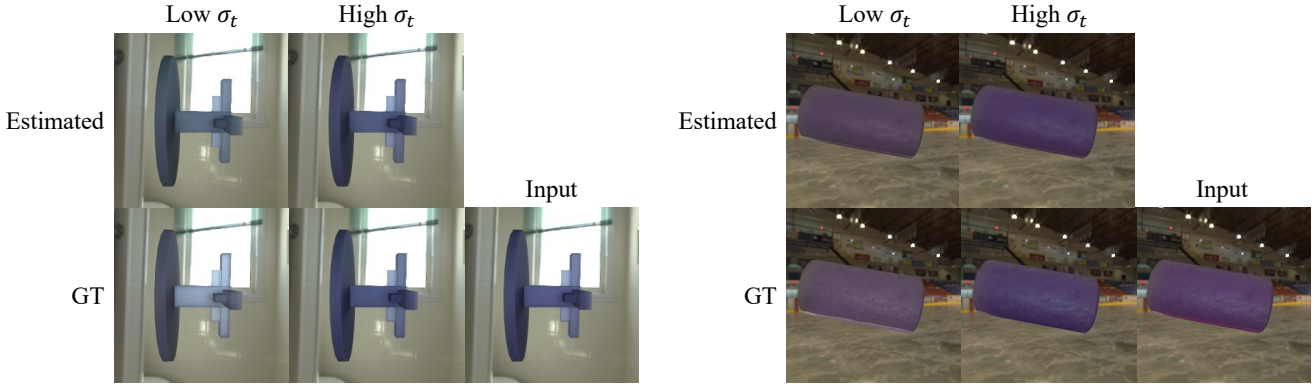

Figure 3. Extinction coefficient σ_t editing results. We edit the σ_t of the input translucent object and compare the estimated images with their ground truth at two different levels.

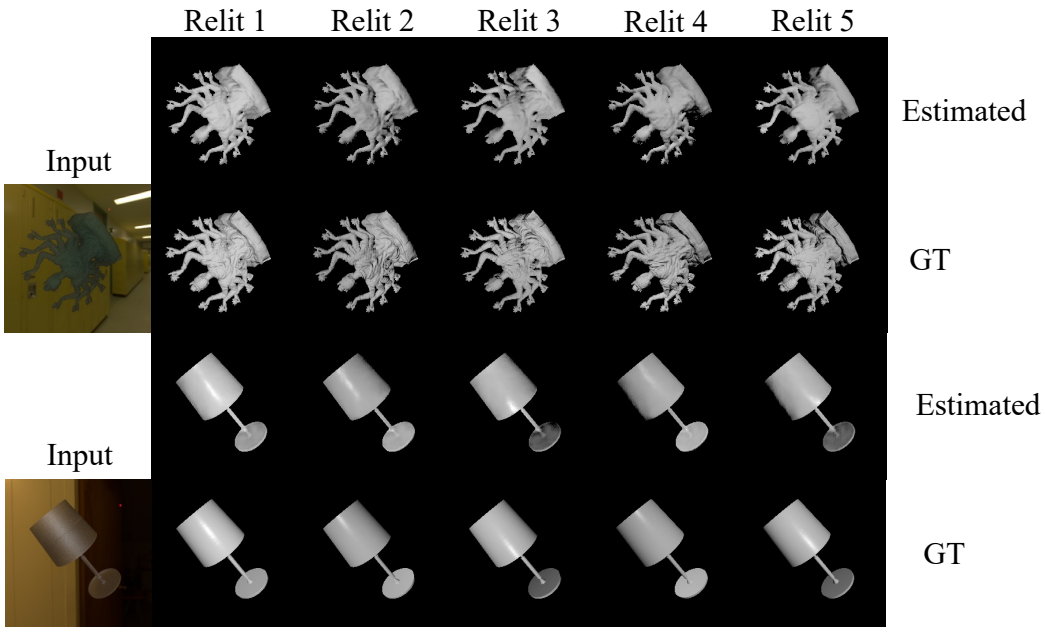

Figure 4. Surface reflectance under different illumination conditions.

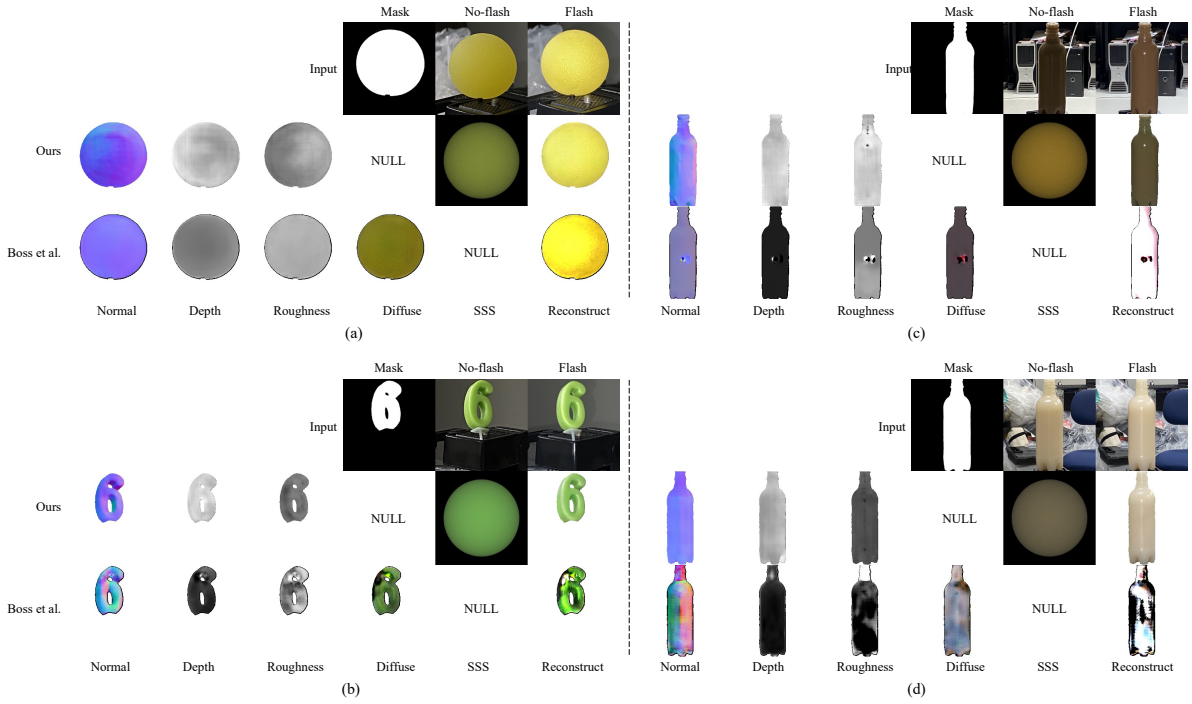

Figure 5. Visual comparison with Boss *et al*. [1]. "Diffuse" in the figure means the surface diffuse albedo. "SSS" denote the subsurface scattering parameters, we use these parameters to render a sphere using Mitsuba2 [8] for visualization. We select two translucent object (a) and (b) with low transparency, and two translucent object (c) and (d) with high transparency for qualitative comparison. Boss *et al.* [1] method only works well for low transparency when subsurface scattering is close to surface scattering (diffuse reflectance). Meanwhile, the proposed method works well for both cases.

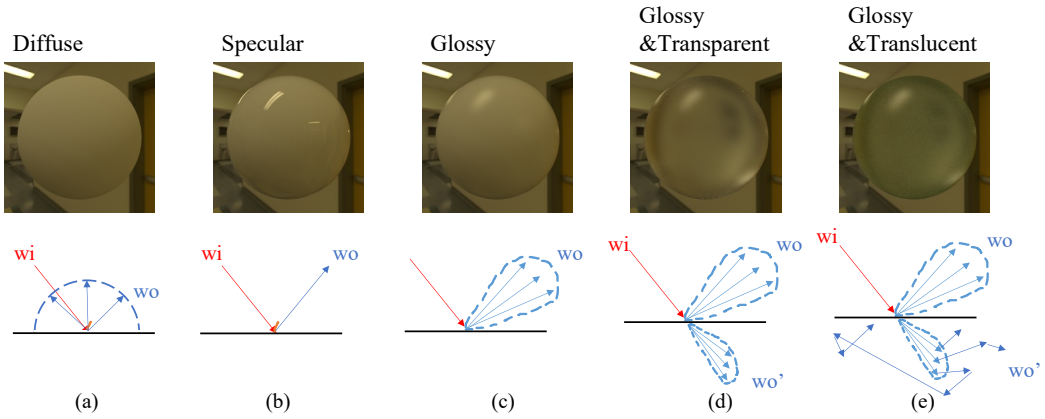

Figure 6. Illustration of different scene representations. The diffuse material (a) assumes that light is reflected uniformly, while the specular material (b) assumes that the incident light and the outgoing light are symmetrical along the surface normal. The glossy material (c) is a generalized version of (a) and (b) that assumes the outgoing light follows a distribution. Introducing refraction into (c) results along a glossy transparent material (d). However, (d) assumes that the light travels in a straight line inside an object. We focus the glossy translucent material (e) which introduces the multiple bounces and multiple paths inside the object.

Table 4. Network structure for the Scattering Encoder. Numbers in the building blocks mean kernel size, number of output channel, and stride, respectively.

stage	building blocks	output size
upsampling convolution 1	$\begin{bmatrix} 4\times 4 & 1024 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{128}\times\frac{W}{128}\times 1024$
upsampling convolution 2	$\begin{bmatrix} 4\times 4 & 512 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{64}\times\frac{W}{64}\times 512$
upsampling convolution 3	$\begin{bmatrix} 4\times 4 & 256 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{32}\times\frac{W}{32}\times 256$
upsampling convolution 4	$\begin{bmatrix} 4\times 4 & 128 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{16}\times\frac{W}{16}\times 128$
upsampling convolution 5	$\begin{bmatrix} 4\times 4 & 64 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{8}\times\frac{W}{8}\times 64$
upsampling convolution 6	$\begin{bmatrix} 4\times 4 & 32 & 2\times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{4}\times\frac{W}{4}\times 32$

Table 5. Network structure of the Surface Encoder. Numbers in the building blocks mean kernel size, number of output channel, and stride, respectively.

stage	building blocks	output size
input convolution	$\begin{bmatrix} 7 \times 7 & 64 & 1 \times 1 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$H \times W \times 64$
downsampling convolution 1	$\begin{bmatrix} 3 \times 3 & 128 & 2 \times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{2} \times \frac{W}{2} \times 128$
downsampling convolution 2	$\begin{bmatrix} 3 \times 3 & 256 & 2 \times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{4} \times \frac{W}{4} \times 256$

Table 6. Network structure of the Decoder. Numbers in the building blocks mean kernel size, number of output channel, and stride, respectively.

stage	building blocks	output size
resnet blocks	$\begin{bmatrix} 3 \times 3 & 288 & 1 \times 1 \\ & \text{BN} & \\ & \text{ReLU} & \\ 3 \times 3 & 288 & 1 \times 1 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix} \times 9$	$\frac{H}{4} \times \frac{W}{4} \times 256$
upsampling convolution 1	$\begin{bmatrix} 3 \times 3 & 128 & 2 \times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$\frac{H}{2} \times \frac{W}{2} \times 128$
upsampling convolution 2	$\begin{bmatrix} 3 \times 3 & 64 & 2 \times 2 \\ & \text{BN} & \\ & \text{ReLU} & \end{bmatrix}$	$H \times W \times 64$
output convolution	$\begin{bmatrix} 7 \times 7 & 3 & 1 \times 1 \\ & \text{BN} & \end{bmatrix}$	$H \times W \times 3$