Shape and Material Capture at Home: Supplementary Material

Daniel Lichy[1] Jiaye Wu[1] Soumyadip Sengupta[2] David W. Jacobs[1]
[1]University of Maryland, College Park [2]University of Washington
dlichy@umd.edu, jiayewu@umiacs.umd.edu, soumya91@cs.washington.edu, djacobs@cs.umd.edu

Contents

1. Overview

In this supplemental material, we provide additional details and results on our method that could not be included in the paper due to space constraints. For the topics covered herein, please refer to the table of contents.

More Results in Video: 10304_supp_video.mp4

2. Video

We have included a video (10304_supp_video.mp4) with this supplementary material containing additional results and comparisons. In the video, we show the meshes reconstructed by our method with three input images and those reconstructed by SDPS-Net [3] retrained with three images. We also show the meshes reconstructed with our method using one image compared to those of Li'18 [6]. The meshes are show undergoing a rotation and being relit.

In the left column, meshes are rendered with just their diffuse albedo. In the middle column, they are rendered with a uniform diffuse albedo, and in the right column, they are rendered with their full estimated BRDF. Since SDPS-Net does not estimate BRDF for the diffuse and specular renderings we use a generic purple BRDF.

3. BRDF Model

We define SV-BRDF using the Cook-Torrance model [4], where the BRDF $B(V, L)$ is a 4D function of viewing direction V and lighting direction L. N is the surface normal, H is the halfway vector of V and L, θ_X is $\arccos(X \cdot N)$, A is the albedo and R is the roughness. The model is then defined by the equations:

$$B(V, L) = \frac{A}{\pi} + \frac{F(H, V)D(H, N)G(H, L, V)}{4(N \cdot L)(N \cdot V)}. \quad (1)$$

$$D(H, N) = \frac{1}{\pi R^2 \cos^4(H \cdot N)} e^{\frac{-\tan^2(H \cdot N)}{\alpha^2}}. \quad (2)$$

$$F(H, V) = F_0 + (1 - F_0)(1 - (H \cdot V))^5 \quad (3)$$

$$a(X) = \frac{1}{R \tan \theta_X} \quad (4)$$

$$G1(X, H) = \begin{cases} 1 & \text{if a} > 1.6 \\ 0 & \text{if } (H \cdot X)(X \cdot N) \leq 0 \\ \frac{3.535a(X) + 2.181\sqrt{a(X)}}{1 + 2.276a(X) + 2.577\sqrt{a(X)}} & \text{otherwise} \end{cases} \quad (5)$$

$$G(H, L, V) = G1(L, H)G1(V, H) \quad (6)$$

$F(H, V)$ is the Schlick approximation to the Fresnel term, $G(H, L, V)$ is Smith's masking-shadowing function

with fast rational function approximation [8], and $D(H, N)$ is the Beckmann distribution. We fix the Fresnel-Schlick F_0 value at 0.05 as done by [6]. Thus BRDF can be parametrized with albedo A and roughness map R.

4. Synthetic Data Generation and Augmentation

4.1. Random BRDF Generation

For spatially varying albedo we gathered 415 free textures from [1], which we divided into train and test sets with a 90-10 split. We augment these albedos, at render time, by multiplying each channel by a random Gaussian variable with mean 1 and standard deviation 0.2.

For generating roughnesses, we take a similar approach to [7]. In [7] they randomly sample Phong exponents uniformly from the bins 0–10, 10–20, 20–40, 40–80, 80–160, 160–320, 320–640,640–2560. We approximate this by sampling from an exponential distribution with median 80. We then convert the sampled Phong exponent to a Beckmann equivalent roughness, R, with the formula $R = \sqrt{\frac{2}{2+E}}$, where E is the Phong exponent as suggested by [5].

We found that the Fresnel term does not make a large difference in appearance visually, so to simplify things we fix $F_0 = 0.05$ as done by [6].

4.2. Scene Generation

We create two synthetic datasets to train our model. The data generation procedure is the same for both datasets, the difference being that the first dataset uses random geometry generated by [10] and the second uses realistic geometry from the sculptures dataset [9]. The first dataset has 20,000 training scenes and 500 test scenes. The second has 10,000 training scenes and 200 test scenes.

The scene layout consists of a rectangle in the x-z plane to represent a floor and either a shape randomly selected from the 5000 shapes generated by [10], which are composed of 1-9 primitives, or a shape from the sculpture dataset. Each primitive, including the floor, is assigned a random BRDF using the procedure described above. An orthographic camera is placed in the y-z plane, pointing at the shape and making an angle with the floor that is randomly sampled between 10° and 45°.

The scene is rendered with six directional lights with unit intensity. The right, front-right, front-left and left lights are first placed with azimuth angles -90°,-45°,45°,90° and elevations of 25° above the floor. They are then perturbed in both the azimuth and elevation randomly by up to 10°. The overhead light is placed with random elevation between 80° and 90° above the floor and random azimuth between 0° and 360°. The co-directional light is simply placed along the camera optical axis.

Scenes are rendered with Mitsuba2's [8] Path-Tracer at 256 samples per pixel in HDR. Our BRDF implementation is a Mitsuba2 port of Boss's Mitsuba1 code [2].

4.3. Training Time Augmentation

At training time images undergo one of two possible size transforms. With probability 0.7 they are randomly cropped to between 70% and 100% of their initial size and resized back to 256×256, and with probability 0.3 they are randomly resized to between 60% and 100% of their initial size. They are then padded with zeros back to a size of 256×256. We performed this procedure so that the network will see the same features at various scales.

We simulate intensity variations by randomly scaling each linear image to have a median selected randomly between 0.01 and 0.2. The images are then sRGB tonemapped and clamped between 0 and 1 before being fed to the network. We found this gives the images intensity variations fairly similar to those observed in natural images.

5. Network Architectures

5.1. Notation

To describe the network architectures used in this paper we first define some notation:

- A-B := apply layer A then apply layer B
- BN := batch norm
- Relu := relu activation
- conv_kn_fm_sp := convolution layer with kernel of size n and m filters (i.e. output channels) and stride p. If stride is 1 we will omit _s1.
- convt_kn_fm_sp := transposed convolution layer with kernel of size n and m filters (i.e. output channels) and stride p.
- Res_n := conv_k3_fn - BN - Relu - conv_k3_fn - BN - +input. This defines the residual block. +input indicates adding the input value to the output value.

5.2. InitNet

InitNet consists of 3 separate networks: one for albedo, normal, and roughness. We will call these InitNetModules. Each InitNetModule takes in 19 channels that are the concatenation of the six three channel images and the segmentation mask. The InitNetModule architecture is given by:

- InitNetModule_c := conv_k7_f64 - BN - Relu - Res_64 - Res_64 - conv_k7_fc.

Where c is 3, 2 and 1 for albedo, normal, and roughness, respectively.

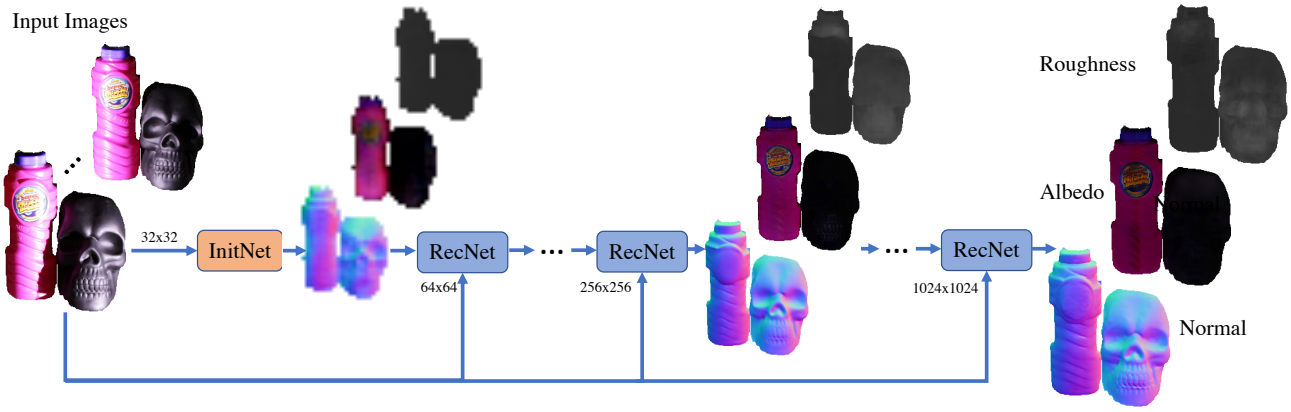

Figure 1: We propose a recursive multi-resolution architecture, RecNet, that predicts surface normal, albedo and roughness from the input image(s) and from the prediction at the previous step by continuously upsampling by a factor of 2. The recursion is initialized by InitNet.

5.3. RecNet

Similarly to InitNet, RecNet consists of three RecNet-Modules. Each takes in 25 channels: 19 for the images and masks, and 6 for the albedo, normal, and roughness estimated by the previous step, which are upsampled by a factor of two to match the size of the input images. The RecNetModule structure is defined as:

- RecNetModule_c := conv_k7_f64 - BN - Relu - Res_64 - Res_64 - conv_k3_f128_s2 - BN - Relu - Res_128 - Res_128 - conv_k3_f256_s2 - Res_256 - Res_256 - convt_k3_f128_s2 - Res_128 - Res_128 - convt_k3_f64_s2 - BN - Relu - conv_k7_fc

Where c is 3, 2 and 1 for albedo, normal, and roughness, respectively. A diagram of the InitNet-RecNet application is included in Figure 1 for reference.

5.4. ResNet for Ablation

For the ablation study we used what is referred to as ResNet in the paper. This is actually 3 separate ResNets, one for albedo, normal, and roughness; although they are trained simultaneously. Their architectures are all the same and are based on [11]. For consistence we refer to these ResNets as ResNetModules. Their architecture is given by:

- ResNetModule := conv_k7_f64 - BN - Relu - conv_k3_f128_s2 - BN - Relu - conv_k3_f256_s2 - BN - Relu - Res_256 - Res_256 - Res_256 - Res_256 - convt_k3_f128_s2 - BN - Relu - convt_k3_f64_s2 - BN - Relu - conv_k7_fc

Where c is 3, 2 and 1 for albedo, normal, and roughness, respectively.

6. SDPS-Net Retraining

SDPS-Net [3] consists of two networks: LCNet which takes n images of an object under varying lighting and estimates the lighting conditions for each image, and NENet which takes the same n images as well as the estimated lighting directions and predicts the normals. Although in principle these networks can take an arbitrary number of input images, we found that performance decreases significantly if the number of training images differs greatly from the number of test images. Therefore, to give SDPS-Net a fair chance, we retrained SDPS-Net three times with 1,2 and 3 input images at training time. For the case of 2 and 3 input images this consisted of a full retraining of LCNet and NENet using the author's default parameters. In the one image case, we only trained NENet by providing it with the ground truth lighting directions rather than those predicted by LCNet.

7. Integrating Normal Maps

Let $f(x, y)$ be the depth of the surface at pixel location (x, y), then the surface normal is given by $(n_1, n_2, n_3) = \frac{1}{\sqrt{(f_x^2+f_y^2+1)}}(f_x, f_y, -1)$. So to find f we can solve the system.

$$f_x = \frac{-n_1}{n_3} \tag{7}$$

$$f_y = \frac{-n_2}{n_3} \tag{8}$$

We discretize this as:

$$f(x_{i+1}, y_j) - f(x_i, y_j) = \frac{-n_1(x_i, y_j)}{n_3(x_i, y_j)} \tag{9}$$

$$f(x_i, y_{j+1}) - f(x_i, y_j) = \frac{-n_2(x_i, y_j)}{n_3(x_i, y_j)} \tag{10}$$

Where (i, j) are the pixel indices. For a normal map with n pixels these equations describe a sparse linear system of $2n$ equations in n unknowns. We find a least squares solution to this system using using Pytorch's LBGFS optimizer.

8. Comparison to SDPSNet (2 Image)

In Tables 1 and 2 we compare our method to SDPS-Net [3], and SDPS-Net retrained with 2 input images on DiLiGenT in the case where the network is given two input images. For the inputs we use either the images from the front and front-left 1 or front and front-right 2. We show superior performance to SDPS-Net even when it is retrained specifically for two input images.

9. Results on Li's Data

In Figures 2 and 3 we compare our results for albedo, normal, and roughness estimation to those of [6] on the data captured by [6].

10. More Results on Our Data

In this section, we show more comparisons to state-of-the-art methods for multi-image normal estimation and single image normal, albedo, and roughness estimation on the data we captured. We also show more results on images captured with our minimal setup using only an iPhone, flashlight, and improvised stand in Figure 4.

Comparisons to SDPS-Net on the three image input problem are found in Figures 5 and 6. Comparisons to Li'18 [6] and Boss'20 [2] on the single input image problem are found in Figure 7,8,9,10, 11,12 and 13.

References

[1] 3d textures. https://3dtextures.me/. Accessed: 2020.

[2] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik P.A. Lensch, and Jan Kautz. Two-shot spatially-varying brdf and shape estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] Guanying Chen, Kai Han, Boxin Shi, Yasuyuki Matsushita, and Kwan-Yee K. Wong. Sdps-net: Self-calibrating deep photometric stereo networks. In *CVPR*, 2019.

[4] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, Jan. 1982.

[5] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[6] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. In *SIGGRAPH Asia 2018 Technical Papers*, page 269. ACM, 2018.

[7] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[9] Olivia Wiles and Andrew Zisserman. Silnet : Single- and multi-view reconstruction by learning from silhouettes. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. BMVA Press, 2017.

[10] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, 37(4):126, 2018.

[11] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Table 1: **Two Image Results on DiLiGenT (Left)** Comparison of SDPS-Net [3], SDPS-Net retrained with 2 input images, and our method on DiLiGenT, using images front and front-left. MAE (in degrees) for each object is reported.

Algorithm	ball	cat	pot1	bear	pot2	buddha	goblet	reading	cow	harvest	mean
SDPS-Net	25.3	27.4	29.5	23.7	24.0	31.7	36.7	35.0	28.9	31.7	29.4
SDPS-Net (retrained)	5.7	17.1	14.9	8.7	16.3	20.3	25.2	24.6	12.7	26.1	17.2
Ours	6.2	14.7	11.4	7.7	11.2	15.0	18.7	17.7	9.7	24.7	13.7

Table 2: **Two Image Results on DiLiGenT (Right)** Comparison of SDPS-Net [3], SDPS-Net retrained with 2 input images, and our method on DiLiGenT, using images front and front-right. MAE (in degrees) for each object is reported.

Algorithm	ball	cat	pot1	bear	pot2	buddha	goblet	reading	cow	harvest	mean
SDPS-Net	27.0	31.0	33.8	24.2	26.0	30.8	41.8	37.1	29.5	31.6	31.3
SDPS-Net (retrained)	6.3	19.0	17.4	9.1	15.8	21.1	27.5	24.6	14.4	26.5	18.2
Ours	6.8	14.9	11.7	7.7	10.9	15.3	19.1	19.7	10.7	24.0	14.1

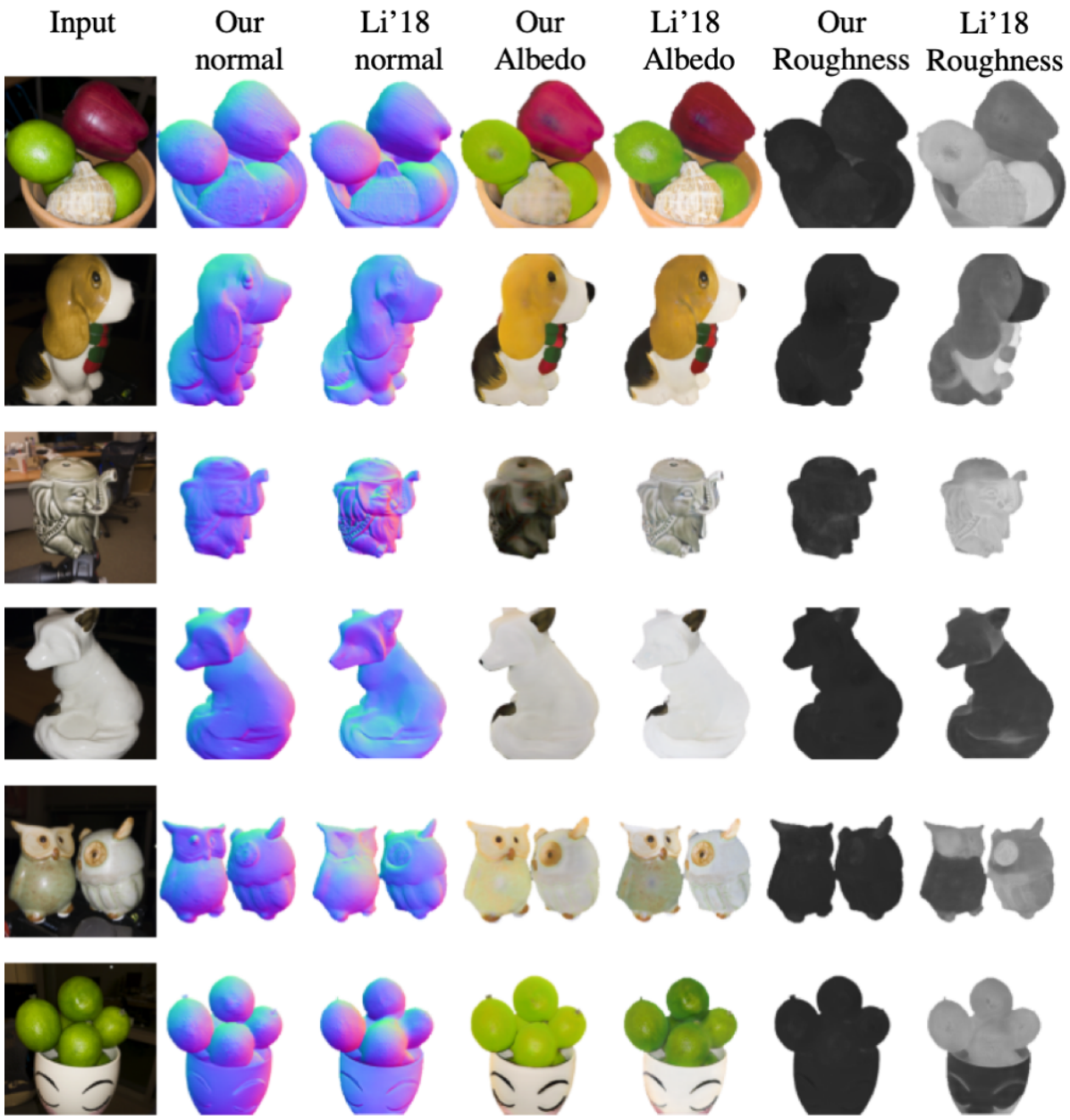

Figure 2: Comparison of our single image results vs. those of Li'18 [6] on data captured by Li'18.

Figure 3: Comparison of our single image results vs. those of Li'18 [6] on data captured by Li'18.

Figure 4: Results taken with only an iPhone, flashlight and improvised stand.

Figure 5: Normal comparison with 3 input images between our method and SDPS-Net [3] retrained with 3 input images.

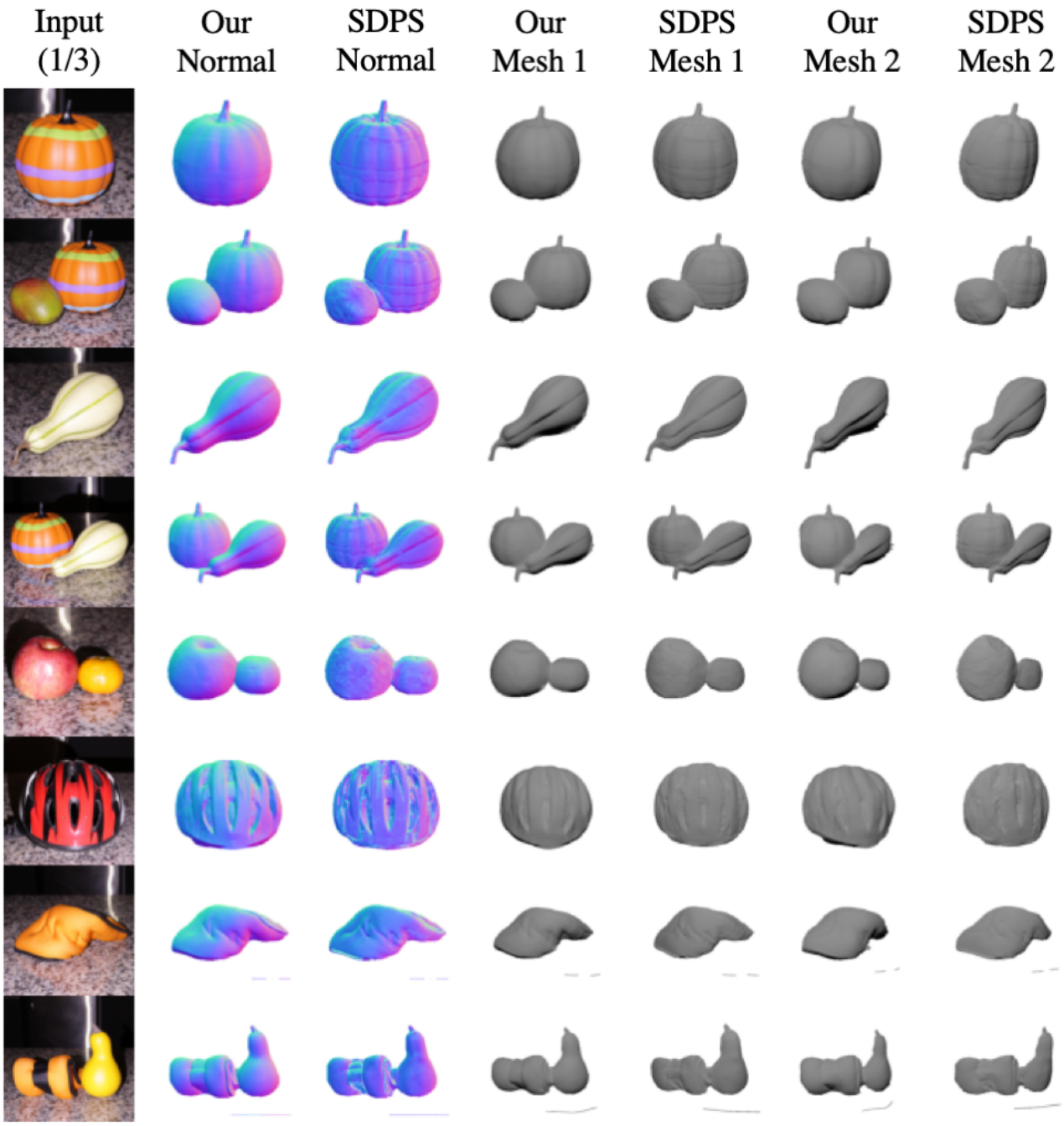

Figure 6: Normal comparison with 3 input images between our method and SDPS-Net [3] retrained with 3 input images.

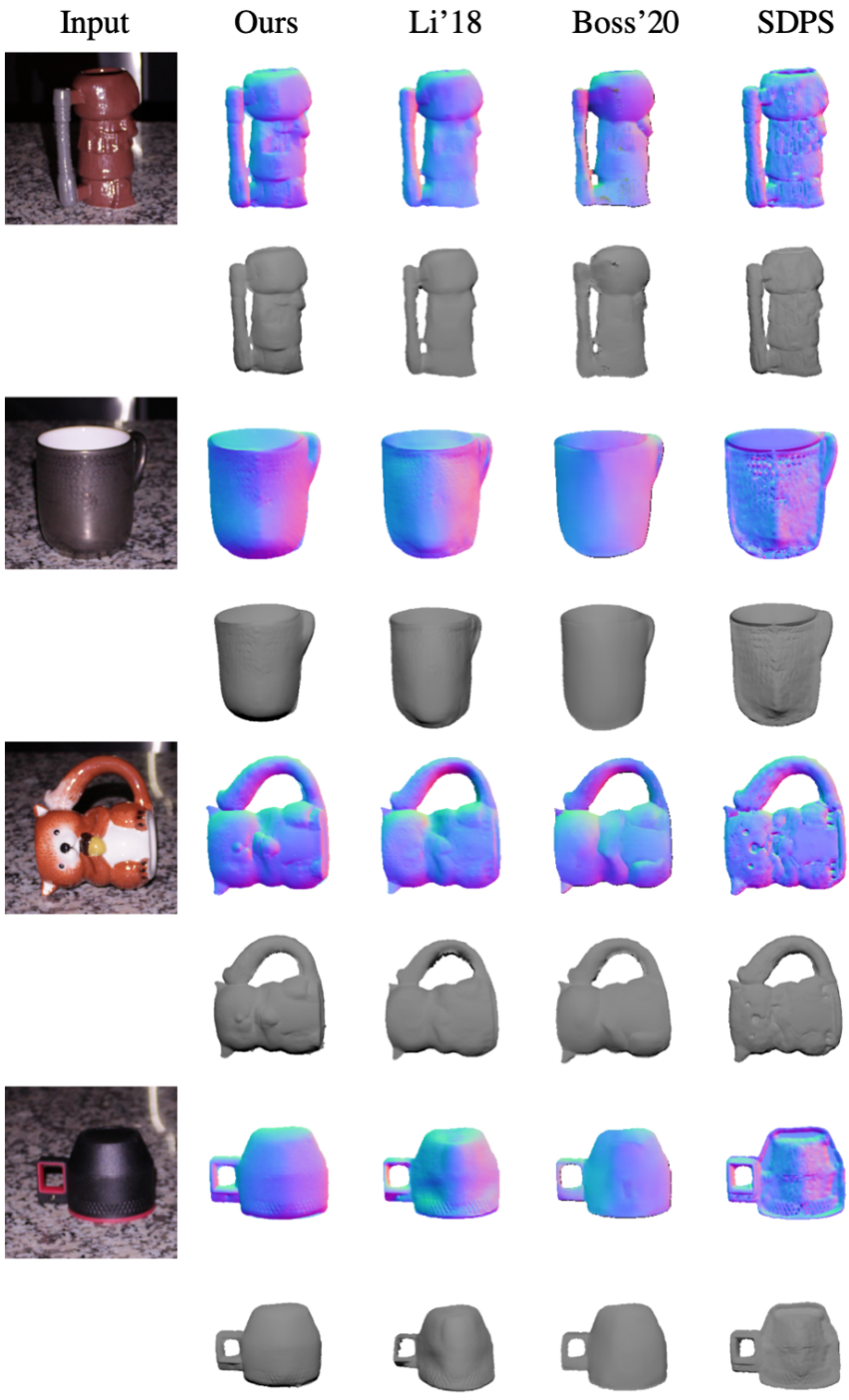

Figure 7: Normal comparison with 1 input image between our method, Li'18 [6], Boss'20 [2], and SDPS-Net retrained with 1 input image [3]

.

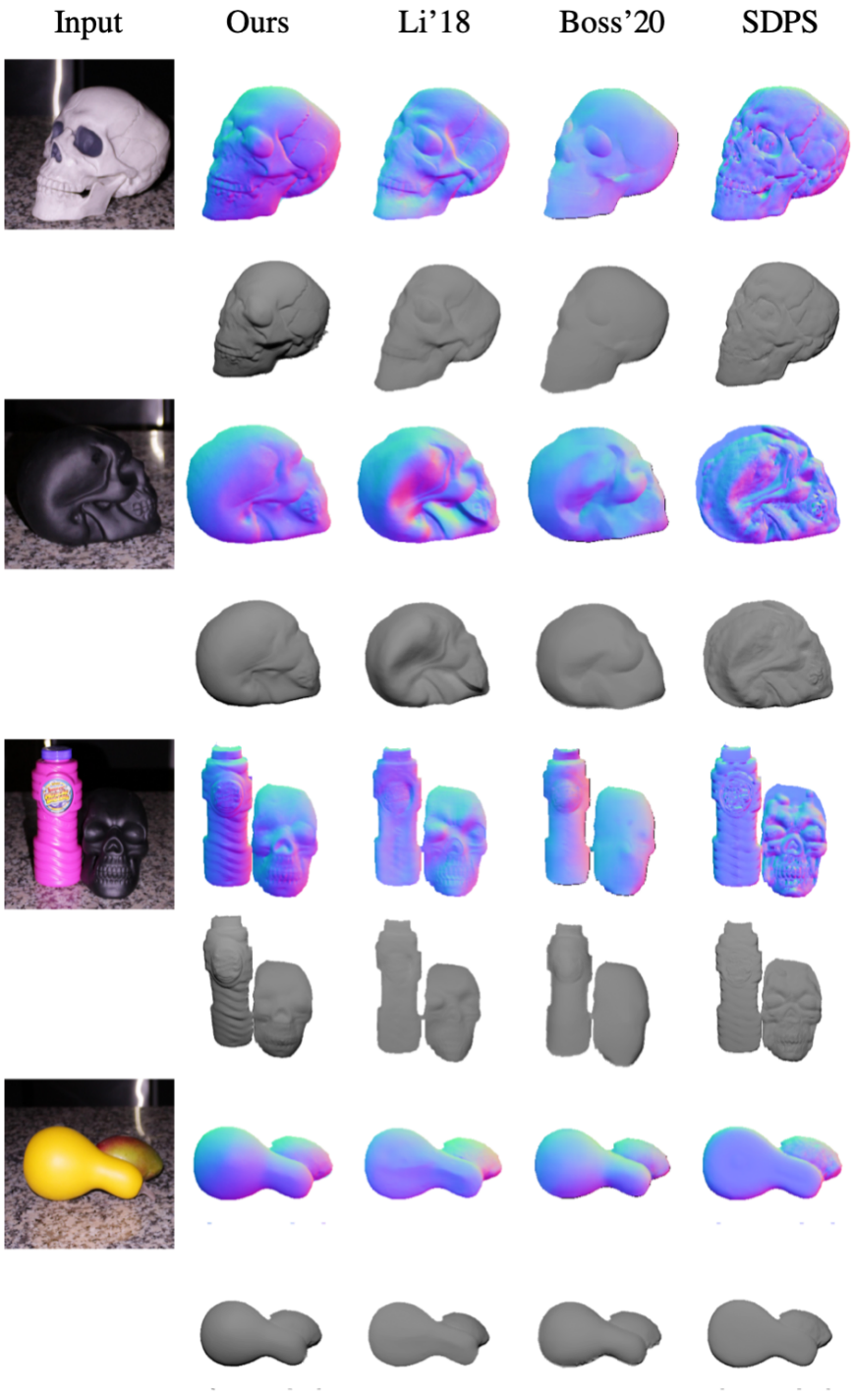

Figure 8: Normal comparison with 1 input image between our method, Li'18 [6], Boss'20 [2], and SDPS-Net retrained with 1 input image [3]

.

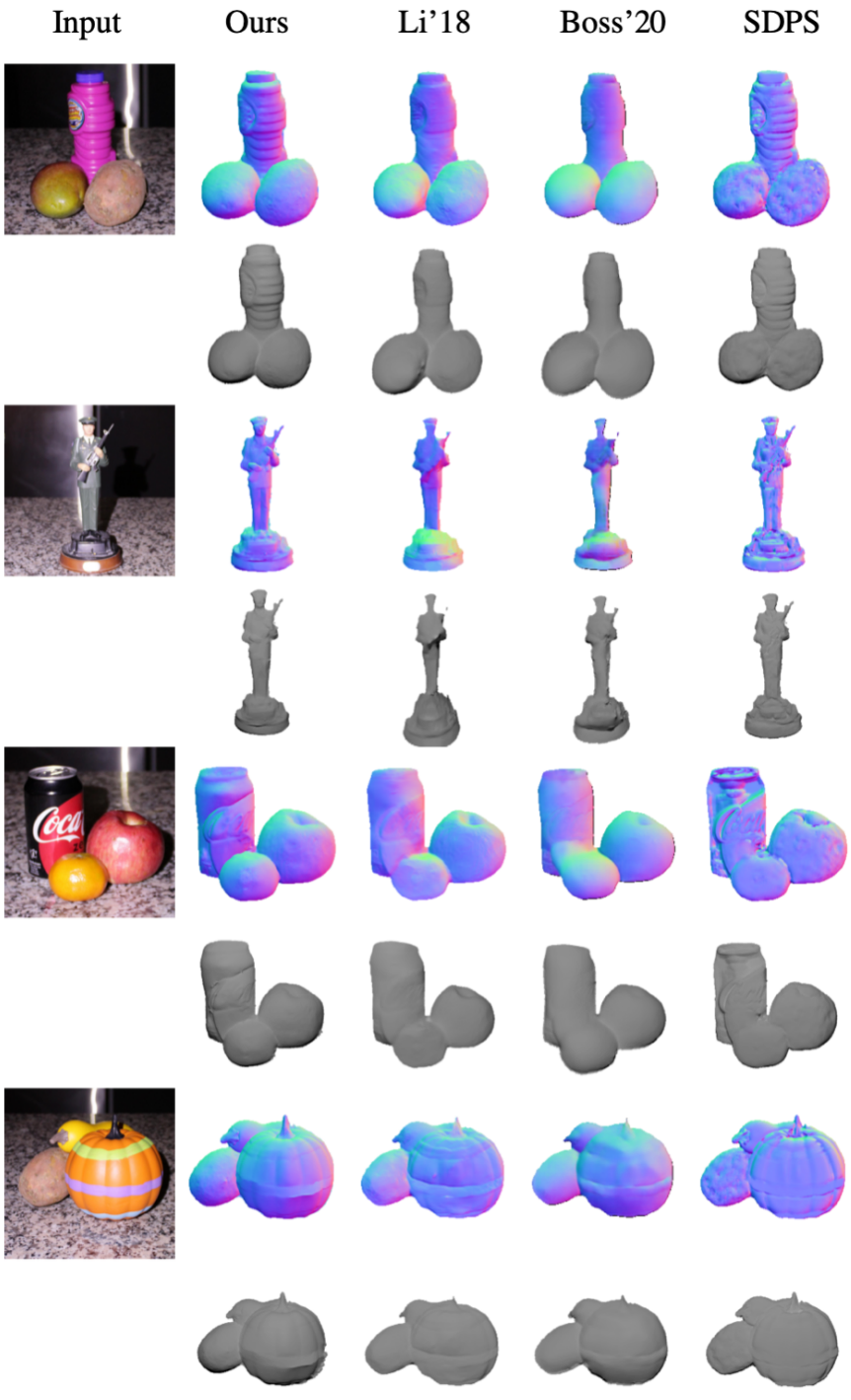

Figure 9: Normal comparison with 1 input image between our method, Li'18 [6], Boss'20 [2], and SDPS-Net retrained with 1 input image [3].

Figure 10: Albedo estimation comparison between between our multi-image method with six input images (ours six), our single-image method (ours single), Li'18 [6], and Boss'20 [2]. Li'18 uses only a single flash image and Boss'20 uses one image with a flash and one without a flash.

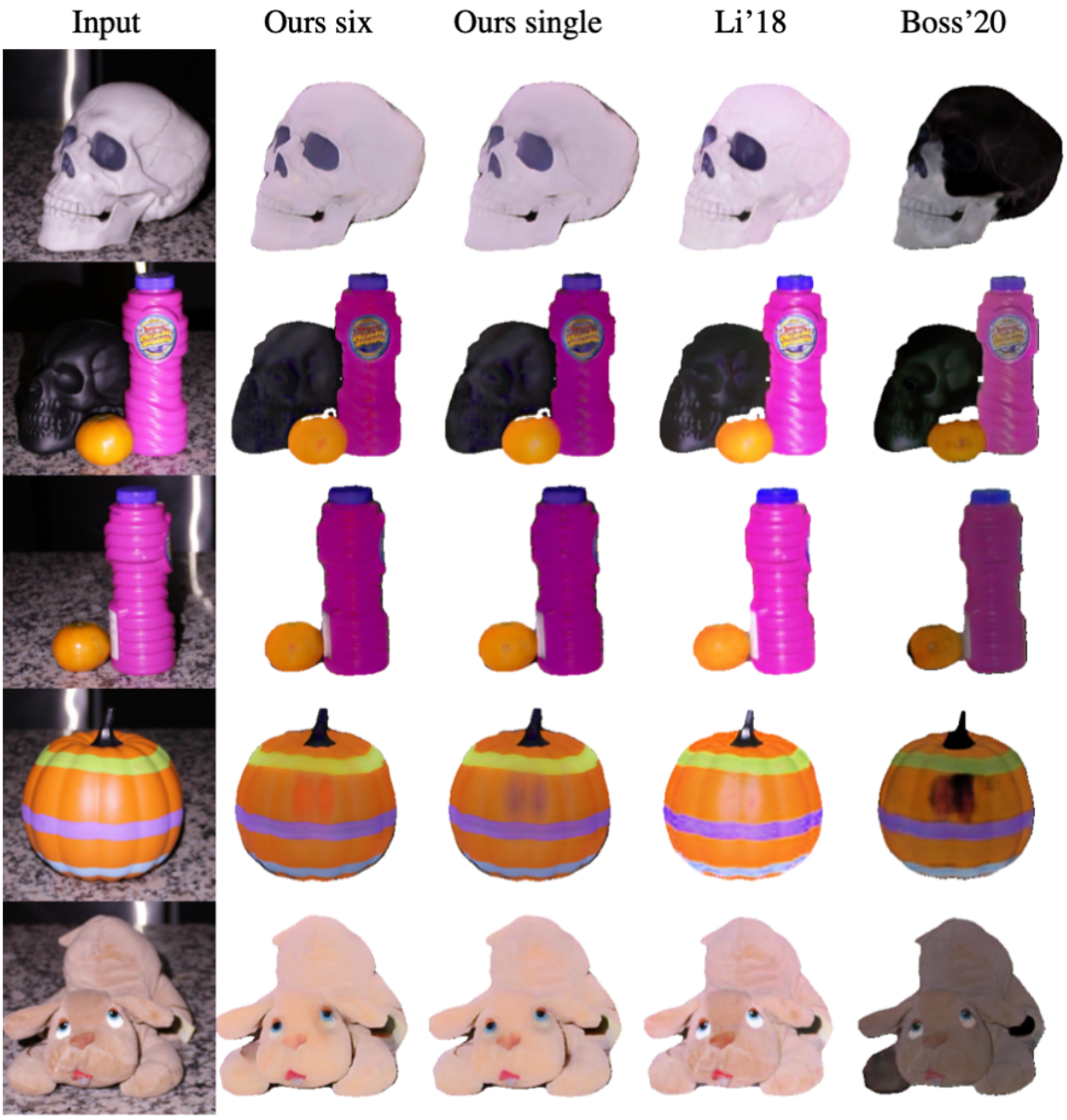

Figure 11: Albedo estimation comparison between between our multi-image method with six input images (ours six), our single-image method (ours single), Li'18 [6], and Boss'20 [2]. Li'18 uses only a single flash image and Boss'20 uses one image with a flash and one without a flash.

Figure 12: Roughness estimation comparison between between our multi-image method with six input images (ours six), our single-image method (ours single), Li'18 [6], and Boss'20 [2]. Li'18 uses only a single flash image and Boss'20 uses one image with a flash and one without a flash. Brighter indicates rougher i.e. less specular.

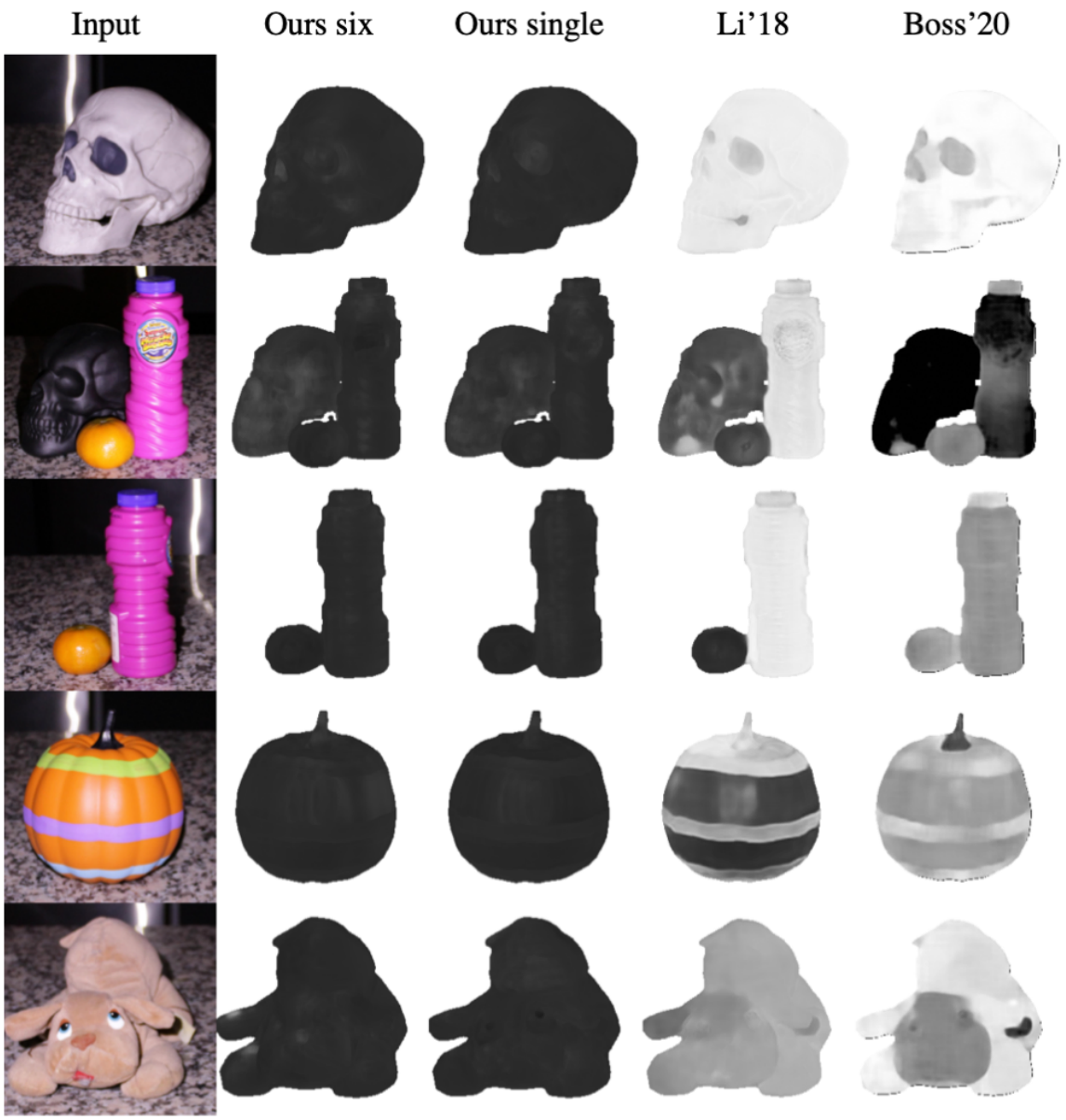

Figure 13: Roughness estimation comparison between between our multi-image method with six input images (ours six), our single-image method (ours single), Li'18 [6], and Boss'20 [2]. Li'18 uses only a single flash image and Boss'20 uses one image with a flash and one without a flash. Brighter indicates rougher i.e. less specular.