DeepPS2: Revisiting Photometric Stereo using Two Differently Illuminated Images (Supplementary Material)

Ashish Tiwari[1] and Shanmuganathan Raman[2]

[1] Prime Minister Research Fellow
[2] Jibaben Patel Chair in Artificial Intelligence
CVIG Lab, IIT Gandhinagar, Gujarat, India
{ashish.tiwari,shanmuga}@iitgn.ac.in

Although the main paper is self-contained in terms of the main results, we believe that the supplementary material can be of help to understand the work in greater detail. Here, we describe the DeepPS2 architecture and remaining results on surface normal, albedo, shading, and illumination estimation. Also, we demonstrate qualitatively the results of image reconstruction and relighting on different objects from the DiLiGenT benchmark [3].

1 DeepPS2 Architecture

Table 1 describes the detailed network architecture. The design of all the modules (except the *illumination module*) is inspired by that of Hourglass networks [5].

2 Results on Normal Estimation

Figure 1 shows the qualitative comparison of the surface normal maps obtained using DeepPS2 with other baselines [4,2,1] over the six remaining objects on the DiLiGenT benchmark dataset.

3 Inverse Rendering Results

Figure 2 shows the qualitative comparison of the estimated illumination, albedo, and shading through DeepPS2. We also show the image reconstruction and relighting results along with the SSIM value.

4 General Comments

- The DeepPS2 framework can be generalized to PSn ($n > 2$) problem. However, since the PS2 problem is a special case of $n \geq 3$ and light sources are coplanar, we chose to rather evaluate the other methods under the PS2 regime for a fair comparison.

- The state of the art methods take multiple ($n > 2$) images to understand specularities (and other lighting dependent effects) through some global feature fusion strategy. However, DeepPS2 is designed to explicitly model specularities using just two images. The key idea here is to use albedo estimation (and refinement) and image relighting to better model lighting effects from just two images since accurate modeling of specularities can give us correct lighting estimates. Table 2 (main paper) shows the performance dip without lighting estimation. Interestingly, these design considerations in DeepPS2 enable it to outperform other methods under the PS2 regime (Table 1, main paper) primarily due to their dependence on feature extraction and fusion from multiple images.
- While the visualizations of images are dark, we found that lighting effects were best discernible with dark backgrounds (with zoom), and thus chose to retain the same.
- The number of bins for light space discretization might not be sufficient for sparse and high frequency specularities and we anticipate a failure if these specularities are concentrated in a small image region and albedo refinement can be affected. Such specularities have been observed primarily (to some extent) in the HARVEST and READING objects. However, our sampling strategy, i.e., taking images from a different bin as input and the albedo refinement method, has modeled these complexities as evident from quality of refined albedos in Fig. 2. Some artifacts can occur if the network fails to accurately estimate albedo for some paired combination of input images (Fig. 2).
- We realize that it would be interesting to observe the performance of DeepPS2 on surfaces with specularities heavily concentrated in smaller regions and we plan to work in this direction in the future. Understanding the particular light source configuration(s) for which DeepPS2 can fail is an interesting direction to explore.

Table 1. Detailed network architecture of DeepPS2

Module	Architecture
Encoder	conv(k=6, p=2, s=2, cin = 7, cout = 32), BN, ReLU conv(k=4, p=1, s=2, cin = 32, cout = 64), BN, ReLU conv(k=4, p=1, s=2, cin = 64, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 256), BN, ReLU conv(k=4, p=1, s=2, cin=256, cout = 512), BN, ReLU
Decoder (Normal and Albedo)	conv(k=4, p=1, s=2 cin = 512, cout = 256), BN, ReLU conv(k=4, p=1, s=2, cin = 512, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 256, cout = 64), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 32), BN, ReLU conv(k=4, p=1, s=2, cin = 64, cout = 64), BN, ReLU Normal: conv(k=5, p=2, s=1, cin=64, cout = 3), Tanh Albedo: conv(k=5, p=2, s=1, cin=64, cout=6), Tanh
Illumination	conv(k=3, p=0, s=1, cin = 9, cout = 64), BN, ReLU conv(k=3, p=0, s=1, cin = 64, cout = 128), BN, ReLU conv(k=3, p=0, s=1, cin = 128, cout = 256), BN, ReLU **Regress θ:** Linear(256, 256), ReLU, Dropout(0.25) Linear(256,64), ReLU, DropOut(0.25) Linear(64, 5) **Regress ϕ:** Linear(256, 256), ReLU, Dropout(0.25) Linear(256,64), ReLU, DropOut(0.25) Linear(64, 5)
Albedo Refinement	conv(k=6, p=2, s=2, cin = 44, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 256), BN, ReLU conv(k=4, p=1, s=2, cin = 256, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin =256, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin =256, cout = 64), BN, ReLU conv(k=5, p=2, s=1, cin =64, cout = 6), Tanh
Image Reconstruction	conv(k=6, p=2, s=2, cin = 15, cout = 64), BN, ReLU conv(k=4, p=1, s=2, cin = 64, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 256), BN, ReLU conv(k=4, p=1, s=2, cin=256, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin=256, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin=256, cout=64), BN, ReLU conv(k=4, p=1, s=2, cin=128, cout=64), BN, ReLU conv(k=5, p=2, s=1, cin=64, cout=6), Tanh
Image Relighting	conv(k=6, p=2, s=2, cin = 7, cout = 64), BN, ReLU conv(k=4, p=1, s=2, cin = 64, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin = 128, cout = 256), BN, ReLU lighting feature(256) conv(k=4, p=1, s=2, cin=256, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin=256, cout = 128), BN, ReLU conv(k=4, p=1, s=2, cin=256, cout=64), BN, ReLU conv(k=4, p=1, s=2, cin=128, cout=64), BN, ReLU conv(k=5, p=2, s=1, cin=64, cout=6), Tanh **Lighting Feature**: conv(k=1, p=0, s=1, cin=3, cout=64) conv(k=1, p=0, s=1, cin=64, cout=128), BN, Upsample(2) conv(k=3, p=1, s=1, cin=128, cout=128), BN, Upsample(2) conv(k=3, p=1, s=1, cin=128, cout=256), BN, Upsample(2) conv(k=3, p=1, s=1, cin=256, cout=256)

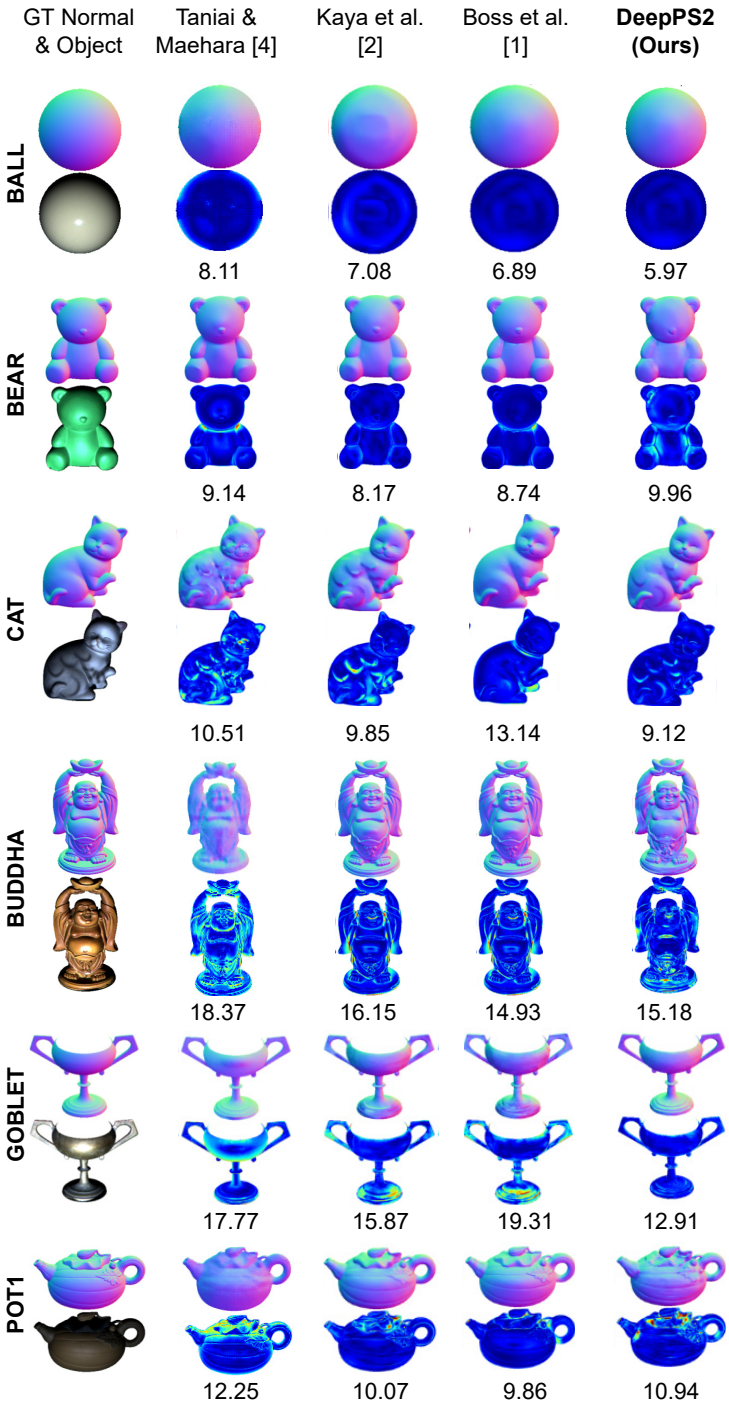

Fig. 1. Normal estimation results on remaining objects in the DiLiGenT Benchmark

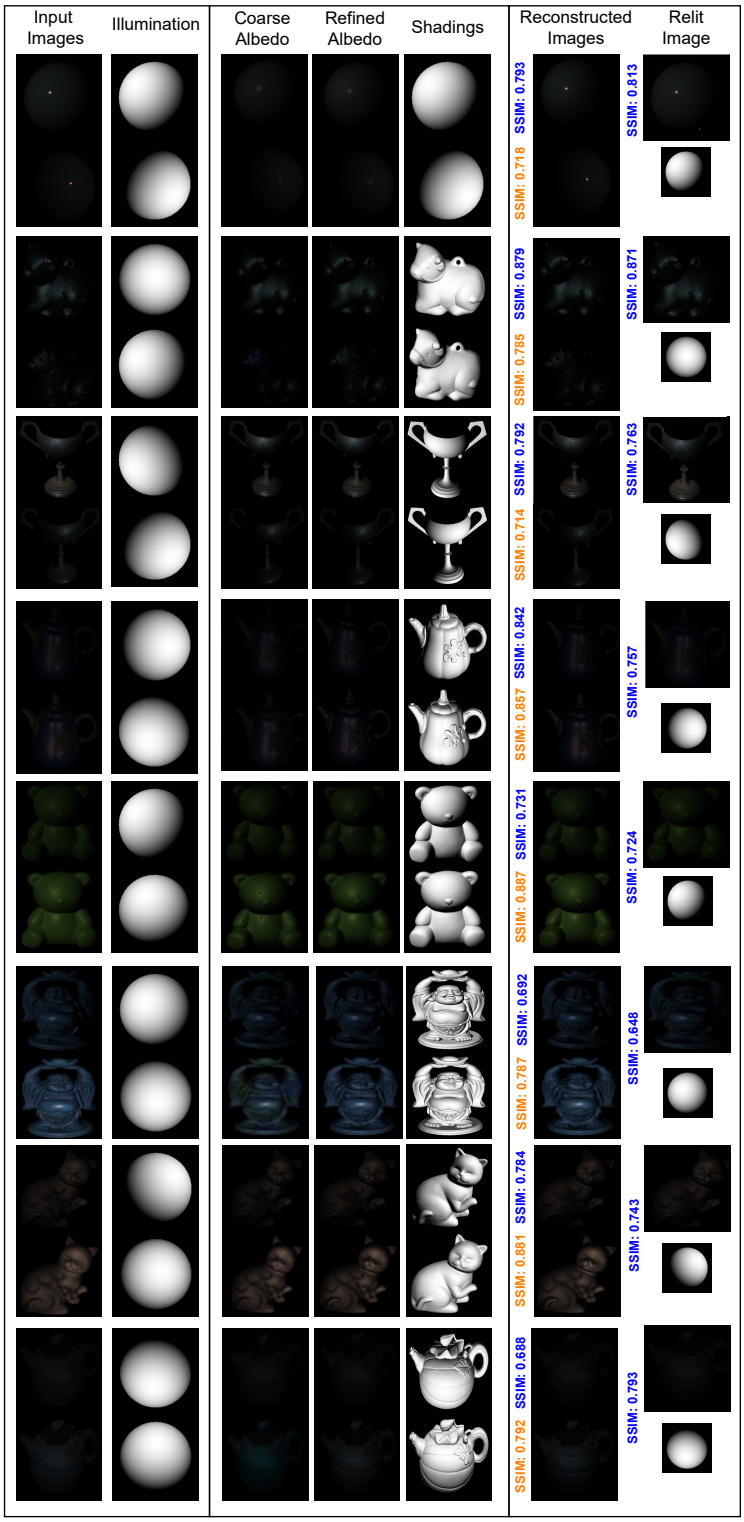

Fig. 2. Inverse rendering results on remaining objects from the DiLiGenT Benchmark

References

1. Boss, M., Jampani, V., Kim, K., Lensch, H., Kautz, J.: Two-shot spatially-varying brdf and shape estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3982–3991 (2020)
2. Kaya, B., Kumar, S., Oliveira, C., Ferrari, V., Van Gool, L.: Uncalibrated neural inverse rendering for photometric stereo of general surfaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3804–3814 (2021)
3. Shi, B., Wu, Z., Mo, Z., Duan, D., Yeung, S.K., Tan, P.: A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3707–3716 (2016)
4. Taniai, T., Matsushita, Y., Sato, Y., Naemura, T.: Continuous 3d label stereo matching using local expansion moves. IEEE transactions on pattern analysis and machine intelligence **40**(11), 2725–2739 (2017)
5. Yang, J., Liu, Q., Zhang, K.: Stacked hourglass network for robust facial landmark localisation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 79–87 (2017)