Eurographics Symposium on Rendering 2021
A. Bousseau and M. McGuire
(Guest Editors)

Volume 40 (2021), Number 4

Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering: Supplemental Material

Fujun Luan[1,3], Shuang Zhao[2], Kavita Bala[1], Zhao Dong[3]

[1]Cornell University
[2]University of California, Irvine
[3]Facebook Reality Labs

In this document, we provide additional analyses and results in addition to those in the main paper.

Specifically, we include detailed information on the performance of our analysis-by-synthesis optimization in §1. Then, §2 contains additional ablation studies and comparisons with existing methods. Lastly, in §3, we show additional reconstruction results (using both synthetic and real data).

1. Optimization performance

We provide a detailed analysis on the performance of our technique in this section. The table below shows the total optimization time for each example as well as the percentages for differentiable rendering (DR), loss computation (Loss), backpropagation (BP), and geometric processing (GP) that involves steps such as mesh evolution and remeshing. The performance numbers are acquired on a workstation equipped with an Intel Xeon E5-2630 v3 processor, 64 GB of RAM, and an Nvidia Titan RTX graphics card.

	Opt. time	**DR**	**Loss**	**BP**	**GP**
Kitty	18.54 min	2.02%	22.57%	7.16%	68.25%
Duck	18.63 min	2.07%	22.16%	8.91%	66.86%
Bear	19.64 min	1.96%	21.02%	8.28%	68.74%
Bell	40.90 min	2.03%	20.01%	7.61%	70.35%
Pig	41.98 min	2.09%	21.04%	7.58%	69.29%
Pony	27.89 min	2.23%	21.11%	7.27%	69.39%
Camera	64.83 min	2.77%	34.00%	7.37%	55.86%
Chime	82.39 min	3.18%	29.10%	7.79%	59.93%
Nefertiti	93.57 min	2.98%	31.50%	7.20%	58.32%
Buddha	100.55 min	2.80%	31.60%	6.88%	58.72%

We note that the geometric processing (GP) step takes a large fraction of total optimization time. This is mainly because the el-Topo library [B*09] is CPU-based and single-threaded. We expect a better implementation of this library to significantly improve the performance of our technique.

2. Ablation studies and comparisons

Additional ablation studies. Our technique is capable of generating plausible results using only 10 inputs. However, with too few input images, our analysis-by-synthesis optimization could become highly under-constrained, causing the reconstruction results to have overly smooth geometries. In practice, with 50 or more input images, our method can generate high-quality reconstructions that generalize well to novel conditions. Thus, we use 50 inputs for the synthetic results and 100 for the real ones in both the paper and the rest of this document.

Additional comparisons. We show in Figure 1 additional comparisons between geometries reconstructed using COLMAP [SZPF16], Kinect Fusion [NIH*11], and our method. Using coarse initializations provided with *KF (Low)*, our method consistently outperforms both *COLMAP* and *KF (High)*.

We further demonstrate the importance of having accurate geometric gradients in Figure 2. When using identical initialization and optimization configurations, gradients generated by our differentiable renderer (depicted in §3 of the paper) can yield high-quality optimization results. Using biased gradient estimates given by existing methods like SoftRas [LLCL19], PyTorch3D [RRN*20], Mitsuba 2 [NDVZJ19] and Nvdiffrast [LHK*20], on the other hand, produces reconstructions with consistently worse qualities.

3. Additional results

Reconstruction results. We show more reconstruction results of synthetic objects in Figure 3 and real ones in Figure 5.

Applications. Since our reconstructions use standard mesh-based representations, they can be easily used in a wide range of applications. In Figure 6, we show physics-based renderings of our reconstructed models in complex virtual scenes. Figure 7 shows augmented reality (AR) examples where our models are inserted into two real scenes.

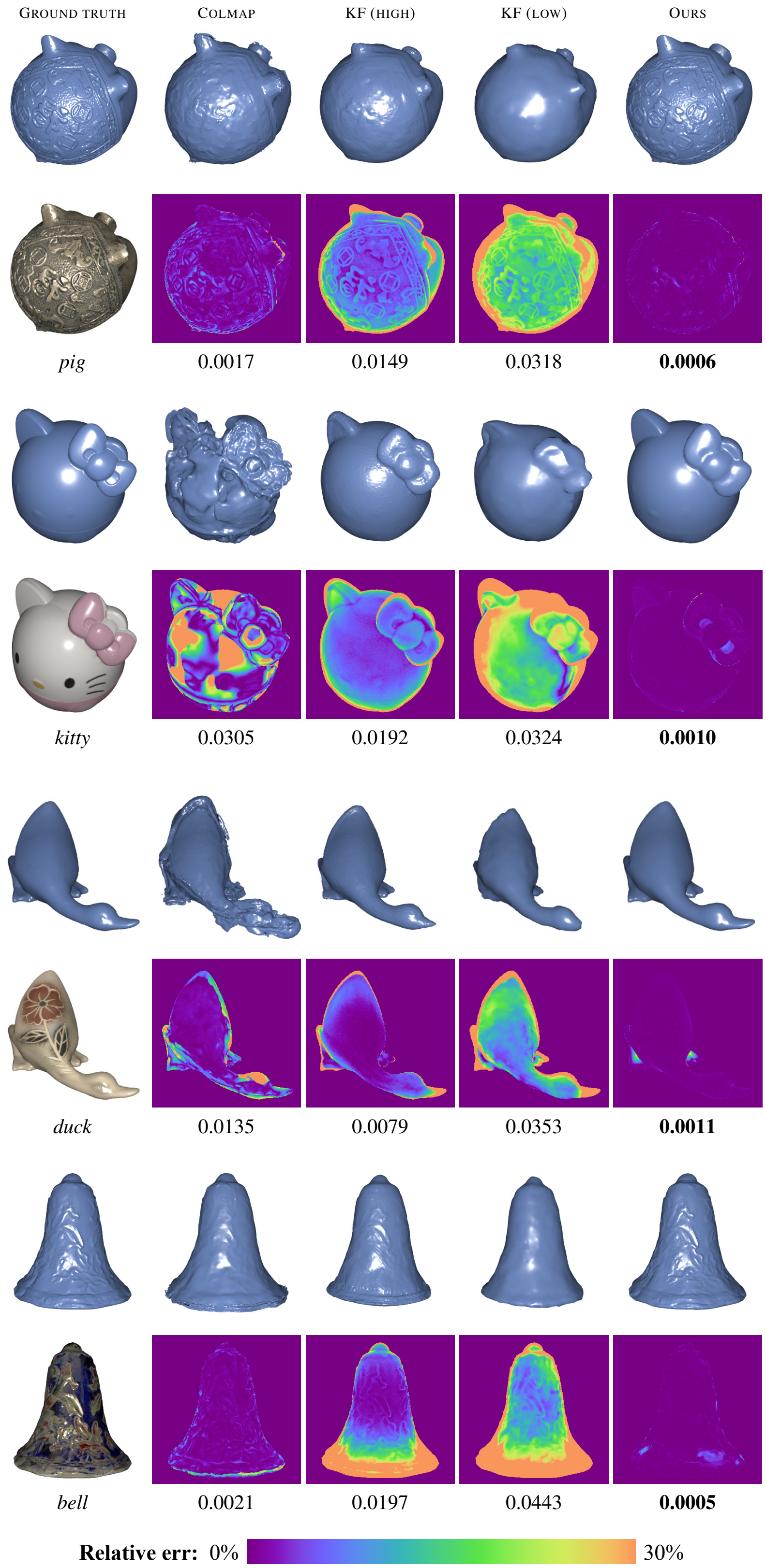

Figure 1: ***Comparison*** *with COLMAP [SZPF16] and Kinect Fusion [NIH*11] using synthetic inputs. The* COLMAP *results are generated using exact camera poses; the* KF (High) *and* KF (Low) *results are created using Kinect Fusion with high-resolution ground-truth depth and low-resolution noisy depth, respectively. Our method, when using crude initializations given by* KF (Low) *and RGB inputs, produces much more accurate geometries than* COLMAP *and* KF (High). *The number below each result indicates the average point-to-mesh distance.*

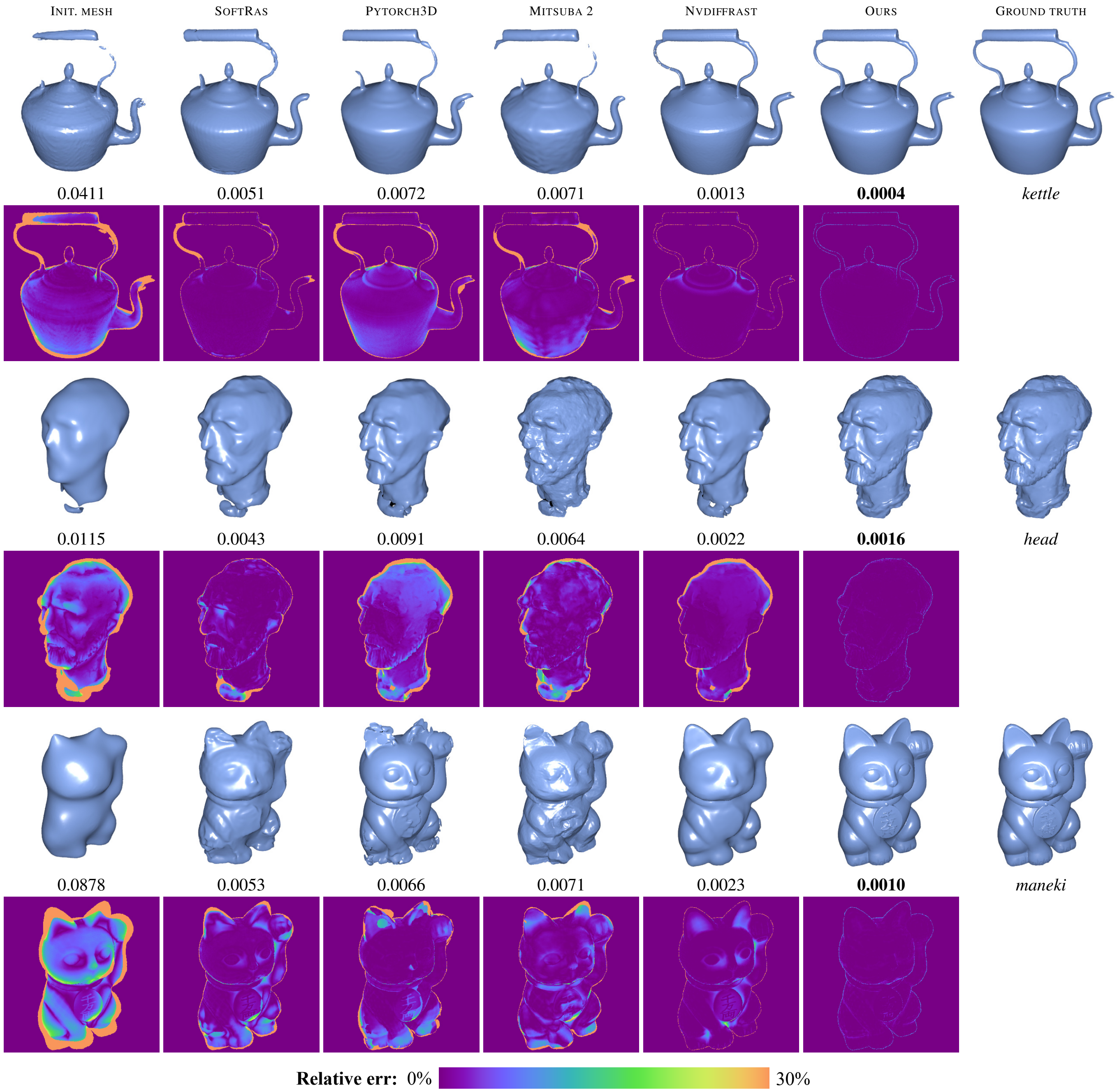

Figure 2: ***Comparison*** *with SoftRas [LLCL19], PyTorch3D [RRN*20], Mitsuba 2 [NDVZJ19] and Nvdiffrast [LHK*20]. We render all reconstructed geometries using Phong shading and visualize depth errors (wrt. the ground-truth geometry). Initialized with the same mesh (shown in the left column), optimizations using gradients obtained with SoftRas and PyTorch3D tend to converge to low-quality results due to gradient inaccuracies caused by soft rasterization. Mitsuba 2, a ray-tracing-based system, also produces visible artifacts due to biased gradients resulting from an approximated reparameterization [LHJ19]. Nvdiffrast is using multisample analytic antialiasing method to provide reliable visibility gradients, which yields better optimization result overall. When using gradients generated with our differentiable renderer, optimizations under identical configurations produce results closely resembling the targets. The number below each result indicates the average point-to-mesh distance.*

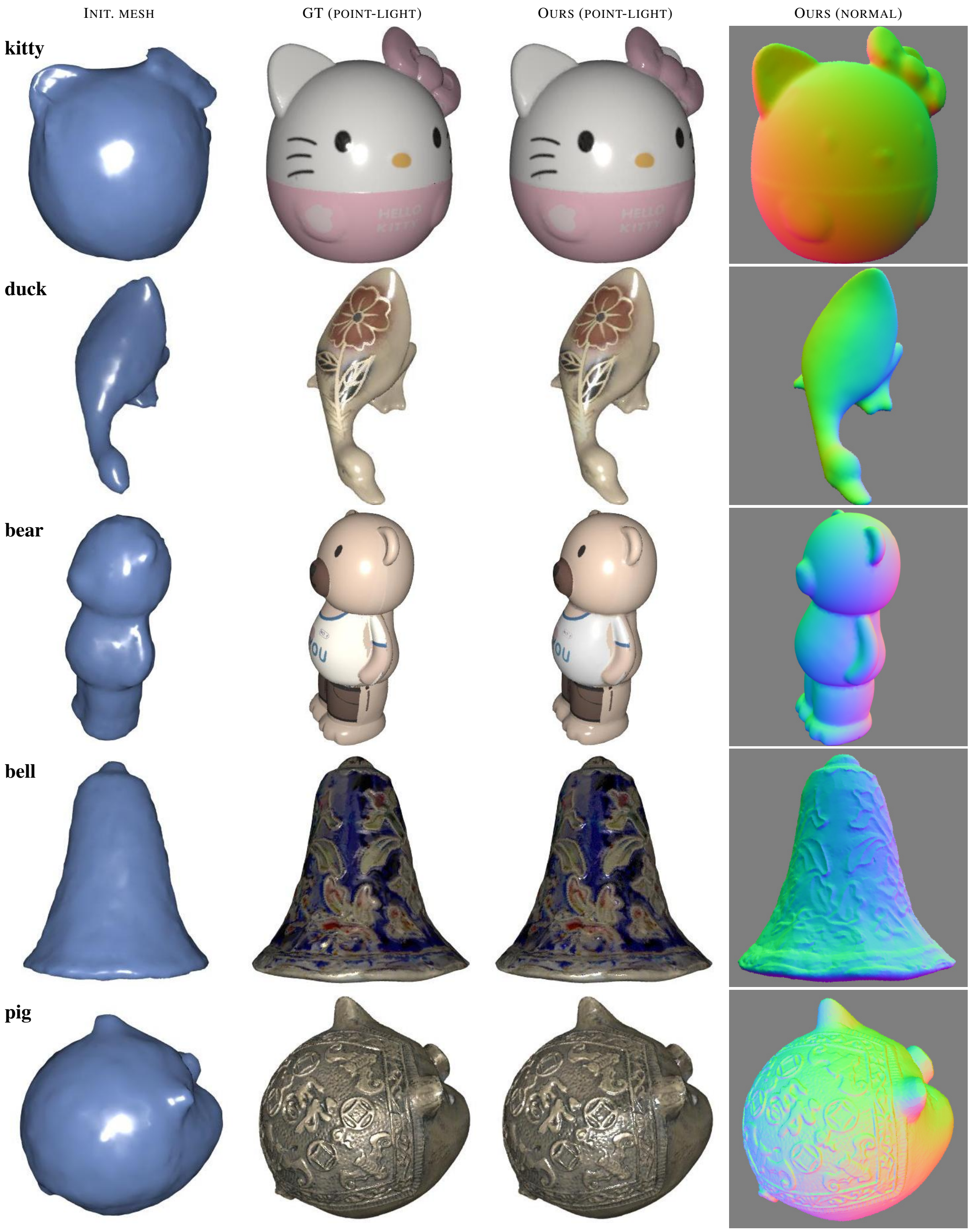

Figure 3: ***Reconstruction results*** *using synthetic inputs: We obtain the initial meshes (in the left column) using Kinect Fusion [NIH*11] with low-resolution* (48×48) *and noisy depths. The detailed normal variations in our models emerge entirely from the reconstructed geometries (and no displacement or normal mapping is used).*

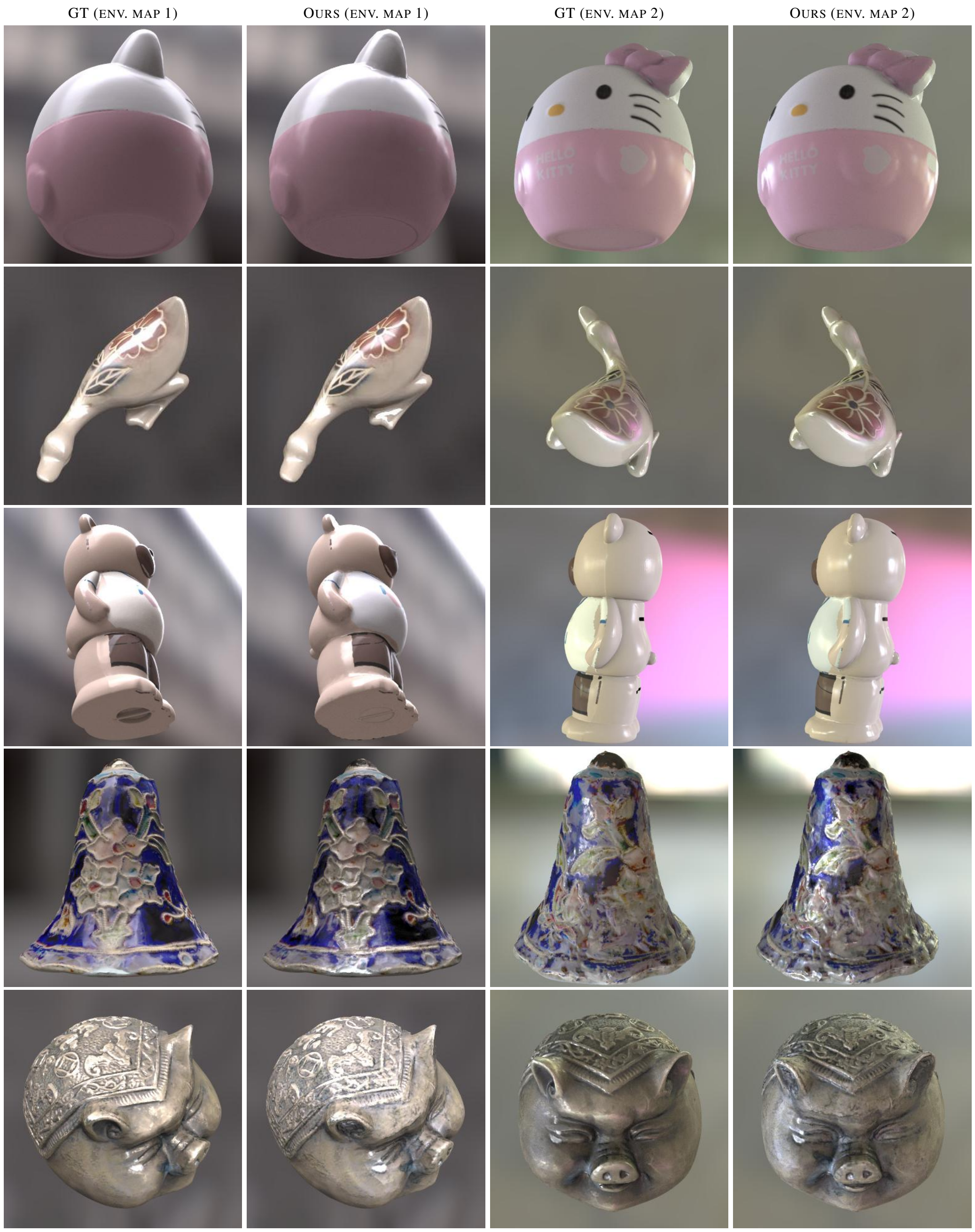

Figure 4: ***Reconstruction results*** *using synthetic inputs: Re-renderings of the reconstruction results from Figure 3 under environmental illuminations.*

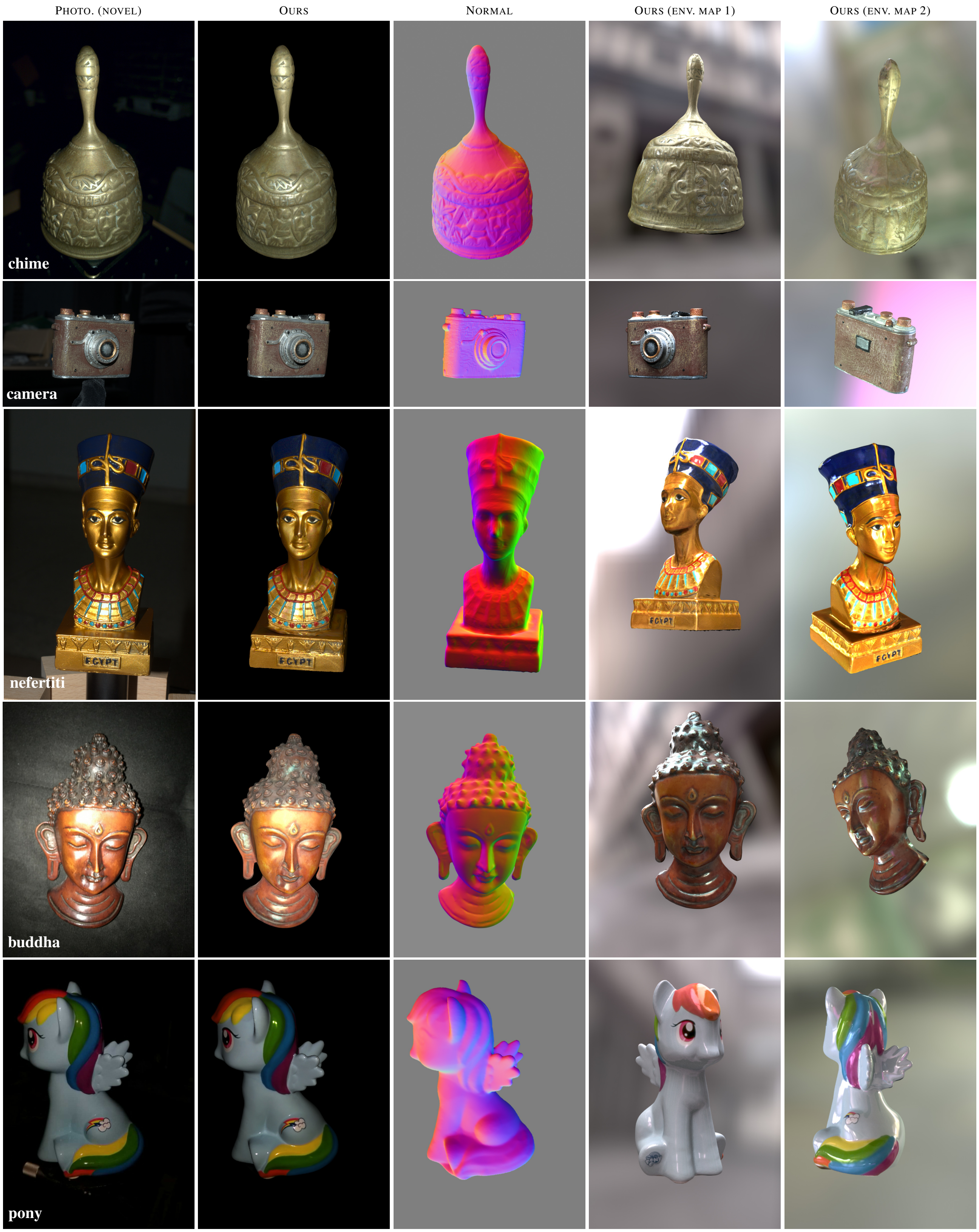

Figure 5: ***Reconstruction results*** *of real-world objects: Similar to the synthetic examples in Figure 3, our technique recovers detailed object geometries and reflectance details, producing high-quality results under novel viewing and illumination conditions.*

Figure 6: ***Physics-based renderings*** *of our models (i.e., everything on the tables in the top and bottom-left images; the buddha model on the wall in the bottom-right image) within complex virtual scenes.*

Figure 7: ***AR object insertion:*** *Our reconstructed* kitty *model (left) inserted into a real scene; the object on the right is real.*

References

[B*09] BROCHU T., ET AL.: El Topo: Robust topological operations for dynamic explicit surfaces, 2009. https://github.com/tysonbrochu/eltopo. 1

[LHJ19] LOUBET G., HOLZSCHUCH N., JAKOB W.: Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph. 38*, 6 (2019), 1–14. 3

[LHK*20] LAINE S., HELLSTEN J., KARRAS T., SEOL Y., LEHTINEN J., AILA T.: Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph. 39*, 6 (2020). 1, 3

[LLCL19] LIU S., LI T., CHEN W., LI H.: Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *ICCV* (2019), IEEE, pp. 7708–7717. 1, 3

[NDVZJ19] NIMIER-DAVID M., VICINI D., ZELTNER T., JAKOB W.: Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph. 38*, 6 (2019), 1–17. 1, 3

[NIH*11] NEWCOMBE R. A., IZADI S., HILLIGES O., MOLYNEAUX D., KIM D., DAVISON A. J., KOHI P., SHOTTON J., HODGES S., FITZGIBBON A.: Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR* (2011), IEEE, pp. 127–136. 1, 2, 4

[RRN*20] RAVI N., REIZENSTEIN J., NOVOTNY D., GORDON T., LO W.-Y., JOHNSON J., GKIOXARI G.: Accelerating 3D deep learning with PyTorch3D. *arXiv preprint arXiv:2007.08501* (2020). 1, 3

[SZPF16] SCHÖNBERGER J. L., ZHENG E., POLLEFEYS M., FRAHM J.-M.: Pixelwise view selection for unstructured multi-view stereo. In *ECCV* (2016), Springer. 1, 2