APPENDIX

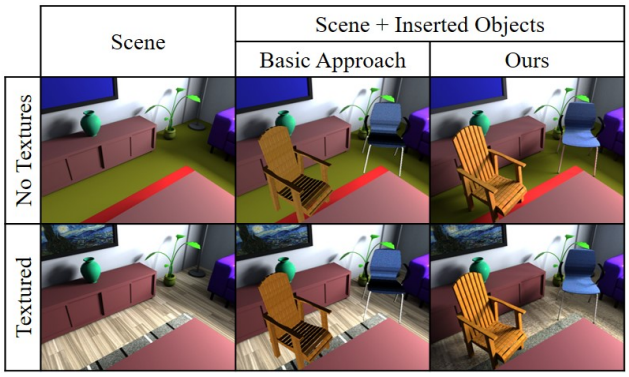

Figure 1: Mixed-reality setting: we insert two new 3D objects (chairs) into an existing 3D scene. Our goal is to find a consistent lighting between the existing and newly-inserted content. In the middle column, we show a naive compositing approach; on the right the results of our approach. The naive approach does not take the 3D scene and light transport into consideration, and fails to photo-realistically render the chair.

In this appendix, we provide additional quantitative evaluations of our design choices in Sec. A. To this end, we evaluate the choice of the batch size, the impact of the variance reduction, and the number of bounces for the inverse path tracing optimization. In addition, we provide additional results on scenes with textures, where we evaluate our subdivision scheme for high-resolution surface material parameter optimization; see Sec. B. Sec. C presents a quantitative comparison to another material estimation method. In Sec. D, we provide examples for mixed-reality application settings where we insert new virtual objects into existing scenes. Here, the idea is to leverage our optimization results for lighting and materials in order to obtain a consistent compositing for AR applications. Finally, we discuss additional implementation details in Sec. E.

A. Qualitative Evaluation of Design Choices

A.1. Choice of Batch Size

In Fig. 2, we evaluate the choice of the batch size for the optimization. To this end, we assume the compute budget for all experiments, and plot the results with respect to time on the x-axis and the ℓ_1 loss of our problem (log scale) on the y-axis. If the batch size is too low (blue curve), then the estimated gradients are noisy, which leads to a slower convergence; if batches are too large (gray curve), then we require too many rays for each gradient step, which would be used instead to perform multiple gradient update steps.

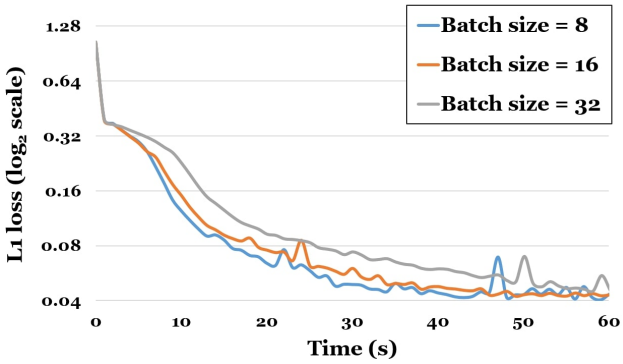

Figure 2: Convergence with respect to the batch size: in this experiment, we assume the same compute/time budget for all experiments (x-axis), but we use different distributions of rays within each batch; *i.e.*, we try different batch sizes.

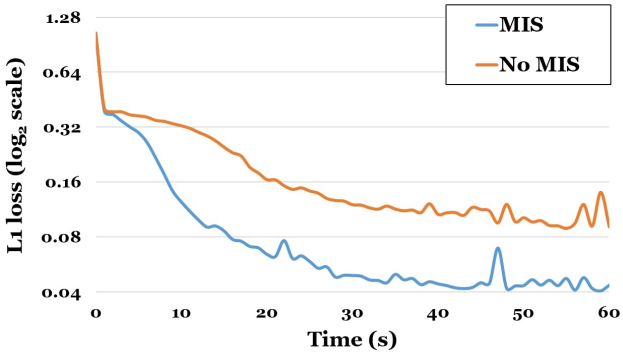

Figure 3: Use of Multiple Importance Sampling during path tracing significantly improves the convergence rate.

A.2. Variance Reduction

In order to speed up the convergence of our algorithm, we must aim to reduce the variance of the gradients as much as possible. There are two sources of variance: the Monte Carlo integration in path tracing and the SGD, since we path trace only a small fraction of captured pixels in every batch.

As mentioned in the main paper, the gradients of the rendering integral have similar structure to the original integral, therefore we employ the same importance sampling strategy as in usual path tracing. The path tracing variance is reduced using Multiple Importance Sampling (*i.e.*, we combine BRDF sampling with explicit light sampling) [5]. We follow the same computation for estimating the gradients with respect to our unknowns. A comparison between implementation with and without MIS is shown in Fig. 3.

A.3. Number of Bounces

We argue that most diffuse global illumination effects can be approximated by as few as two bounces of light. To this end, we render an image with 10 bounces and use it as ground truth for our optimization. We try to approximate the ground truth by renderings with one, two, and three bounces, respectively (see Fig. 4). One bounce corresponds to direct illumination; adding more bounces allows us to take into account indirect illumination as well. Optimization with only a single bounce is the fastest, but the error remains high even after convergence. Having more than two bounces leads to high variance and takes a lot of time to converge. Using two bounces strikes the balance between convergence speed and accuracy.

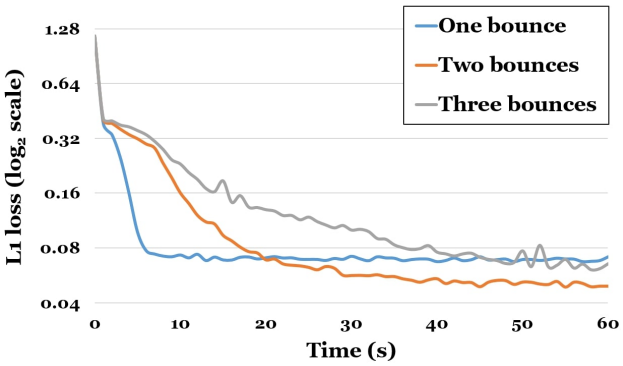

Figure 4: A scene rendered with 10 bounces of light is given as input to our algorithm. We estimate emission and material parameters by using one, two, and three bounces during optimization. Two bounces are enough to capture most of the diffuse indirect illumination in the scene.

B. Results on Scenes with Textures

In order to evaluate surfaces with high-frequency surface signal, we consider both real and synthetic scenes with textured objects. To this end, we optimize first for the light sources and material parameters on the coarse per-object resolution. Once converged, we keep the light sources fixed, and we subdivide all other regions based on the surface texture where the re-rendering error is high; *i.e.*, we subdivide every triangle based on the average ℓ_2 error of the pixels it covers, and continue until convergence. This coarse-to-fine strategy allows us to first separate out material and lighting in the more well-conditioned setting; in the second step, we then obtain high-resolution material information. Results on synthetic data [2] are shown in Fig. 5, and results on real scenes from Matterport3D [1] are shown in Fig. 6.

Method	LIME [4]	Ours
Object 1	0.45%	**0.00037%**
Object 2	1.37%	**0.14%**

Table 1: We compare the relative error between the estimated diffuse albedo for two objects. We outperform LIME even though our method is not restricted to the estimation of only a single material at a time.

C. Additional Comparison to Data-driven Approaches

We compare our approach to Meka et al. [4] and present quantitative results in Tab. 1. Please note that our approach is not limited to a single material of a single object at a time. The other data-driven references are mostly on planar surfaces only and/or assume specific lighting conditions, such as a single point light close to the surface.

D. Object Insertion in Mixed-reality Settings

One of the primary target applications of our method is insertion of virtual objects into an existing scene while maintaining a coherent appearance. Here, the idea is to first estimate the lighting and material parameters of a given 3D scene or reconstruction. We then insert a new 3D object into the environment, and re-render the scene using both the estimated lighting and material parameters for the already existing content, and the known intrinsics parameters for the newly-inserted object. A complete 3D knowledge is required to produce photorealistic results, in order to take interreflection and shadow between objects into account.

In Fig. 1, we show an example on a synthetic scene where we virtually inserted two new chairs. As a baseline, we consider a naive image compositing approach where the new object is first lit by spherical harmonics lighting and then inserted while not considering the rest of the scene; this is similar to most existing AR applications on mobile devices. We can see that a naive compositing approach (middle) is unable to produce a consistent result, and the two inserted chairs appear out of place. Using our approach, we can estimate the lighting and material parameters of the original scene, composite the scene in 3D, and then re-render. We are able to produce consistent results for both textured and non-textured scenes (right column).

In Fig. 2 of the main paper, we show a real-world example on the Matterport3D [1] dataset, where we insert a virtual teddy into the environment. To this end, we first estimate lighting and surface materials in a 3D scan; we then render the scene with and without the new virtual object; finally, we apply the delta image to the original input and composite the virtual object into the image. Compared to the SVSH baseline, our approach achieves significantly better compositing results.

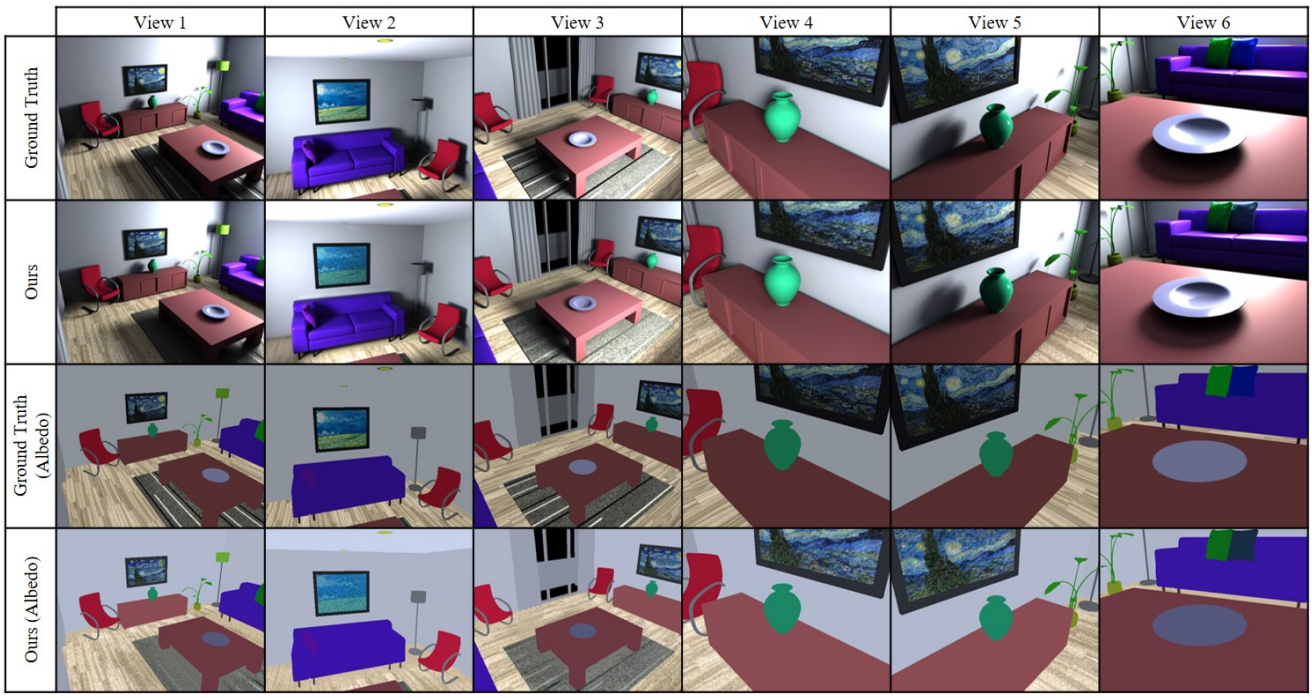

Figure 5: Results of our approach on synthetic scenes with textured objects. Our optimization is able to recover the scene lighting in addition to high-resolution surface texture material parameters.

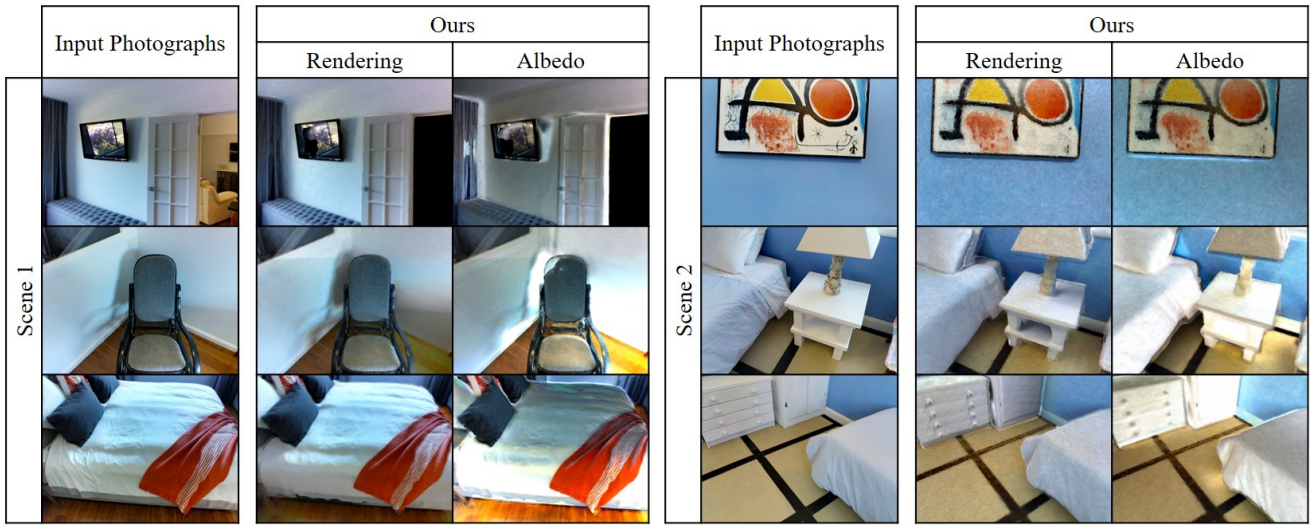

Figure 6: Examples from Matterport3D [1] (real-world RGB-D scanning data) where we reconstruct emission parameters, as well as high-resolution surface texture material parameters. We are able to reconstruct fine texture detail by subdividing the geometry mesh and optimizing on individual triangle parameters. Since not all light sources are present in the reconstructed geometry, some inaccuracies are introduced into our material reconstruction. Albedo in shadow regions can be overestimated to compensate for missing illumination (visible behind the chair in Scene 1), specular effects can be baked into the albedo (reflection of flowers on the TV) or color may be projected onto the incorrect geometry (part of the chair is missing, so its color is projected onto the floor and wall).

E. Implementation Details

We implement our inverse path tracer in C++, and all of our experiments run on an 8-core CPU. We use Embree [6] for the ray casting operations. For efficient implementation, instead of employing automatic differentiation libraries, the light path gradients are computed using manually-derived derivatives.

We use ADAM [3] as our optimizer of choice with an initial learning rate of $5 \cdot 10^{-3}$. We further use an initial batch size of 8 pixels which are uniformly sampled from the set of all pixels of all images. We found marginal benefit of having larger batches, but we believe there is high potential in investigating better sampling strategies. In all our experiments, the emission and albedo parameters are initialized to zero.

For every pixel in the batch, we need to compute an estimate of the pixel color based on the current value of the unknown material and emission parameters. This estimated color is compared against the ground truth color and a gradient is computed depending on the choice of the loss function. For most commonly used loss functions, this gradient will involve a multiplication of the estimated pixel color and its derivative with respect to the unknown parameters. Since these are random variables (approximated by Monte Carlo integration), it is important that they are calculated from independent samples to avoid bias. We use path tracing with multiple importance sampling for the computation of the pixel color, but any unbiased light transport method will produce the correct result.

We extend our path tracer to analytically compute derivatives w.r.t. emission and material parameters as defined by Eq. 5 and 6 of the main paper. To this end, we pass a reference to a structure holding the derivatives to our ray casting function. The product of BSDFs in Eq. 5 is incrementally calculated at each bounce. Given that $L_e(x_i)$ is the unknown emission parameter on surface i, the derivative w.r.t. this emission parameter is equal to the product of the BSDFs at each surface intersection from surface i to the sensor. The derivatives w.r.t. to the materials are computed in similar manner. As per chain rule, we multiply the throughput by the derivative of the BSDF w.r.t. the unknown material parameters to obtain the derivative of the pixel color w.r.t. the unknown material parameters.

We implement multiple importance sampling, a combination of light source sampling and BRDF importance sampling. The importance for light source sampling is based on the unknown emission parameters which may change in every iteration of our optimization. An efficient data structure is needed to store the sampling probabilities for every object. We implement a binary indexed tree (also known as Fenwick tree) for this purpose. This provides logarithmic complexity for both reading and updating the probabilities.

Finally, to make the optimization more robust, we propose a coarse-to-fine approach, where we first optimize for one emission and one material parameter per object instance. Most scenes have only a few emitters, so we employ an L1-regularizer on all the emission parameters. After convergence, the result is refined by optimizing for material parameters of individual object triangles. The light sources stay fixed in this phase, but their emission value may still change. In the end, the triangles may be subdivided as explained in Sec. B to further improve the results.

References

[1] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2, 3

[2] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014. 2

[3] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 4

[4] 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. 2

[5] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162. 1

[6] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. Embree: a kernel framework for efficient CPU ray tracing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33(4):143, 2014. 4