Efficient and Differentiable Shadow Computation for Inverse Problems – Supplemental Document –

Linjie Lyu, Marc Habermann, Lingjie Liu, Mallikarjun B R, Ayush Tewari, and Christian Theobalt

Max Planck Institute for Informatics, Saarland Informatics Campus

1. Overview

In the following, we provide more details regarding the sphere fitting process (Sec. 2), geometry deformation (Sec. 3), faster computation of SH multiplications (Sec. 4), implementation (Sec. 5), more results (Sec. 6), and further limitations (Sec. 7).

2. Sphere Fitting

Given a water-tight geometry and its bounding sphere set, the Sphere Outside Volume (SOV) [9] is defined as the volume inside the sphere set while being outside the mesh. To approximate the geometry with our sphere representation as closely as possible, we want to minimize the SOV by optimizing the sphere positions and radii. Therefore, we first voxelize the space around the mesh such that the SOV can be computed as the number of voxels which are inside the sphere set and outside the mesh. Given a voxel center $\mathbf{v}_j$ and a fixed geometry, an outside-geometry indicator function Og_j can be pre-computed as:

$$Og_j = \begin{cases} 0, & \text{if } \mathbf{v}_j \text{ is inside the geometry} \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

Similarly, the outside-sphere indicator function with respect to a sphere represented by the sphere center $\mathbf{c_i}$ and radius r_i is approximated as:

$$Os(\mathbf{v}_j, \mathbf{c_i}, r_i) = \phi(\|\mathbf{v}_j - \mathbf{c_i}\|^2 - r_i^2). \tag{2}$$

ϕ is the activation function, which is a relu function combined with a tanh function. This function Os indicates the relative position between the voxel and the sphere. If the voxel is inside the sphere, Os is 0. When the voxel is far outside from the sphere, Os increases and will become 1 in the limit. Thus, the SOV energy

$$E_{SOV}(C,R) = \sum_{j \in V} Og_j \cdot (1 - \prod_{i \in (C,R)} (Os(\mathbf{v}_j, \mathbf{c_i}, r_i))) \tag{3}$$

approximates the number of voxels which are outside the geometry but inside at least one sphere. $C = \{\mathbf{c}_i\}$ and $R = \{r_i\}$ represent the set of sphere centers and radii, respectively, and V is the voxel set. In addition, we use a surface term

$$E_{surface}(C,R) = \sum_{k \in S} \prod_{i \in (C,R)} (Os(\mathbf{p}_k, \mathbf{c_i}, r_i)), \tag{4}$$

where $\mathbf{p}_k$ is the sampled points from the geometry surface S, to encourage that the sphere set covers as much of the surface as possible. We minimize

$$E(C,R) = E_{SOV}(C,R) + w_s * E_{surface}(C,R), \tag{5}$$

which is differentiable with respect to each sphere center $\mathbf{c_i}$ and radius r_i. w_s is the weight for the surface term which is between 10 and 150 depending on the type of object and the surface point sampling resolution.

3. Geometry Deformation

After constructing the sphere set, the sphere centers can be directly used to drive the geometry deformation. Like Sumner *et al*. [8], we first link the sphere centers to each other using the K nearest neighbor algorithm. These centers serve as the nodes of the Embedded Graph [8]. Then each vertex on the mesh is registered to its K nearest sphere centers with distance-dependent weights. For different objects, we choose different K varying between 4 and 8. During the deformation, a translation and a rotation matrix is defined at each sphere center which models the local deformation. Then, the vertices on the mesh are influenced by the local deformations of their nearby spheres. During geometry optimization, we regularize the deformation space by employing two spatial regularizers E_{rot} and E_{reg} as proposed by Sumner *et al*. [8]. In contrast to the original approach, we use our shadow-aware image loss (described in Section 3.5 in the main paper) as data term to optimize the 6D poses of the spheres, which enables us to guide the geometry deformation based on the monocular image observation.

4. SH Logarithm and Exponentiation

Next, we explain how the SH multiplication can be approximated using SH logarithm and exponentiation, which allows for faster computation.

4.1. SH Logarithm

Our method leverages SH Logarithm and Exponentiation computation. To project the logarithm of each blocker function

$$V_i(\omega, \mathbf{x}) = \begin{cases} 0, & \text{if } S_i \text{ blocks light in direction } \omega; \\ 1, & \text{otherwise.} \end{cases}$$

into the SH space and to avoid infinite logarithm for 0, we replace the 0-1 blocker function with

$$V_i^{'}(\omega, \mathbf{x}) = \begin{cases} e^{-\epsilon}, & \text{if } S_i \text{ blocks in direction } \omega; \\ 1, & \text{otherwise.} \end{cases} \quad (6)$$

In that case,

$$log(V_i^{'}(\omega, \mathbf{x})) = \begin{cases} -\epsilon, & \text{if } S_i \text{ blocks in direction } \omega: \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

As in Eq. 11 in the main paper (Section 3.3), the SH coefficients of this function can be easily computed using the SH rotation [2], and it is differentiable with respect to $\mathbf{x}$, the sphere center $\mathbf{c_i}$, and radius r_i.

Note that Ren *et al*. [7] leverage eigenvalue analysis to approximate a mapping table from the SH vector to its SH logarithm, which is applied after the projection of the blocker function to the SH space. This is not continuous with respect to the sphere orientations. Instead, the whole process can be made differentiable by directly projecting the logarithm of the blocker function to the SH space.

4.2. SH Exponentiation

We leverage optimal linear SH exponentiation approximation, as used in Ren *et al*. [7]. Given the SH coefficient vector $\boldsymbol{f}$ for a function f, the SH exponentiation, which is an approximation of the SH coefficients of $exp(f)$, can be efficiently computed as an operation on the SH vector $\boldsymbol{f} = [\boldsymbol{f}_0, \boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_{n^2-1}]$. We first use DC Isolation [7], to obtain the DC component $\boldsymbol{f}_0$, and the remaining component $\hat{\boldsymbol{f}} = [0, \boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_{n^2-1}]$. Then, the SH exponentiation is approximated as:

$$exp_*(\boldsymbol{f}) = exp(\frac{\boldsymbol{f}_0}{\sqrt{4\pi}})(a(\|\hat{\boldsymbol{f}}\|)\mathbf{1} + b(\|\hat{\boldsymbol{f}}\|)\hat{\mathbf{f}}\,, \quad (8)$$

where $\mathbf{1} = [\sqrt{4\pi}, 0, 0, ..., 0]$ is also a vector in SH space. Ren *et al*. [7] use a non-continuous tabulation method for the function a and b, while we empirically use $a(x) = exp(0.1x)$ and $b(x) = exp(-0.2x)$, which allows differentiating through the SH Exponentiation. Further, we also use scaling/squaring [7] to scale $\boldsymbol{f}$ to a suitable range for the SH Exponentiation.

4.3. Hyperparameters

The functions $a(x)$ and $b(x)$ in Eq. 8 affect the quality of the approximated visibility. The optimal choice of a and b is related to the number of overlapping spheres, the number of bands of spherical harmonics, and the range of $\|\hat{\boldsymbol{f}}\|$. We empirically select $a(x)$ and $b(x)$, and use scaling/squaring [7] so that the shadow looks reasonable in most of our experiments. The optimal approximation of the SH exponentiation requires adequate data from different settings, and a learning-based method could be a potential solution.

5. Implementation Details

Next, we provide more information about the implementation details, which includes the hyperparameters, initialization, and settings for comparisons with Redner.

5.1. Optimization

For texture, lighting, and 6D pose reconstruction, we use the Adam optimizer with a learning rate of 10^{-2} for all methods. For geometry deformation and reconstruction from shadows, we use Adam with a learning rate of 10^{-2} for translations, and 10^{-3} for the rotation matrices. Our optimization stops if $\frac{\Theta_n - \Theta_{n-1}}{\Theta_n} \leq 0.001$, where Θ_n is the objective function at the n_{th} iteration.

5.2. Initialization

For texture and lighting optimizations, we initialize the texture and environment maps as pure black. For 6D pose reconstruction, geometry deformation, and reconstruction from shadow, our optimization starts from a natural pose, which is reasonably close to the ground truth pose such that gradients can still guide the unknown scene parameters (see also the supplemental video).

5.3. Redner Settings

We extensively experimented with different settings for Redner [3] and choose the fastest setting, which leads to a successful optimization within a comparable number of iterations. In our experience, a sampling number of 64 per pixel is required for texture and environment map optimization. We keep the same settings for pose and geometry optimization. Notice that even with 4 samples per-pixel, our method is $\sim$10 times faster for inverse problems such as environment map estimation.

6. Additional Results

In the following, we show more results including a joint optimization of multiple scene parameters, results on real data, a comparison to another differentiable ray tracing approach [5], and a visualization of the optimized scene illumination.

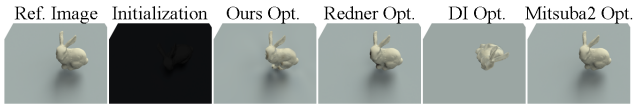

Figure 1. Joint optimization of 6D pose and scene illumination.

6.1. Joint Optimization

Our method is capable of optimizing multiple variables simultaneously, as shown in Fig. 1, where the 6D pose of the object and scene illumination are optimized at the same time. We outperform DI methods [6] as they cannot account for the shadow. Compared to ray tracing based methods [3, 5], our method achieves visually similar results while being much faster.

6.2. Real Data

We also evaluate our method on real data. In this experiment, we show that we can optimize for the illumination, given geometry, albedo and pose of the object. We use a light-stage dataset [10] where a human face is captured from 16 viewpoints and 150 one-light-at-a-time (OLAT) light sources. We use photogrammetry-based 3D reconstruction from Agisoft Metashape[1] to compute the 3D geometry and pose from the multi-view images. To compute the albedo, we average the color of the individual OLAT images. A reference image is synthesized by linearly combining the OLAT images using a real environment map from the Laval Outdoor dataset [1]. Given the albedo, geometry, and pose, we optimize for the environment map (see Fig. 2). Our method obtains results comparable in quality to Redner, and higher quality compared to DI. This can be seen in the color-coded error image, which shows the error between the optimized rendering and the ground truth image.

6.3. Mitsuba 2 Comparison

We also compare our method with the Mitsuba 2 renderer [5] using the reparameterization integrator [4] on texture, environment, and pose optimization, see Fig. 1. Due to the memory limits, we only use 16 samples per pixel. The optimized results using Mitsuba 2 are qualitatively close to those obtained with Redner (we use Redner with 32 samples per pixel for the result in Fig. 1.) Mitsuba 2 takes 1.7s to 2.1s for each iteration, depending on the optimization task, for an image resolution of 512×512. In contrast, our method only requires 18ms to 200ms, while achieving a visually comparable result.

6.4. Visualization of the Optimized Scene Lighting

In Fig. 3, we visualize the estimated scene lighting given just a monocular reference image of the scene. Since the object has a diffuse material, and due to the fact that only a monocular image of the scene is given, the optimization problem is very underconstrained. In consequence, multiple solutions exist, which still lead to a plausible rendering that matches the scene. However, for the DI method [6], the reference image can not be reproduced since shadows are not modeled in the rendering process.

[1] https://www.agisoft.com/

7. Additional Limitations

For the geometry and 6D pose optimization, our method can fail to optimize the deformation/6D pose parameters correctly when the initialization is too far away from the ground truth. For example, when the ground truth shadows and the estimated shadows do not overlap at all in image space. In this case, there are no meaningful gradients for supervising the scene parameters.

References

[1] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] Jan Kautz, John Snyder, and Peter-Pike J Sloan. Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics. *Rendering Techniques*, 2(291-296):1, 2002.

[3] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[4] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[5] 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.

[6] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[7] Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng, and Baining Guo. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *ACM SIGGRAPH 2006 Papers*, pages 977–986. 2006.

[8] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007.

[9] Rui Wang, Kun Zhou, John Snyder, Xinguo Liu, Hujun Bao, Qunsheng Peng, and Baining Guo. Variational sphere set approximation for solid objects. *The Visual Computer*, 22(9):612–621, 2006.

[10] Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, and Markus Gross.

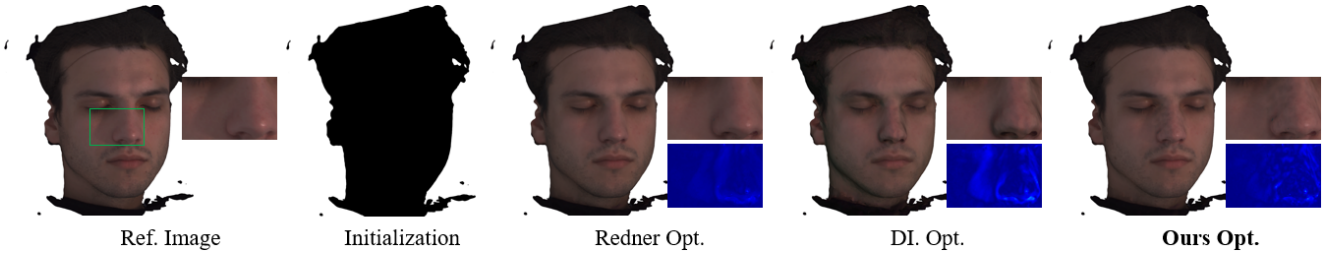

Figure 2. Optimization of the environment map using a real world image.

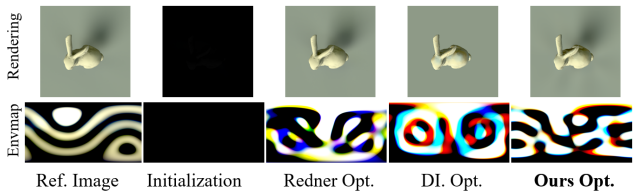

Figure 3. Visualization of the scene lighting (bottom) optimized using a single image observation. The rendered results are shown on top.

Analysis of human faces using a measurement-based skin reflectance model. *ACM Trans. on Graphics (Proceedings of SIGGRAPH)*, 25(3):1013–1024, 2006.