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

In the following, we provide more details regarding the sphere fitting process (Sec. 1), the geometry deformation (Sec. 2), the faster computation of SH multiplications (Sec. 3), the implementation (Sec. 4), and further limitations (Sec. 5).

1. Sphere Fitting

Given a water-tight geometry and its bounding sphere set, the Sphere Outside Volume (SOV) [4] 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

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

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. In addition, we use a surface term

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

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 100 depending on the type of object.

2. Geometry Deformation

After constructing the sphere set, the sphere centers can be directly used to drive the geometry deformation. Like Sumner *et al*. [3], 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 [3]. 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*. [3]. In contrast to the original approach, we use our 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.

3. SH Logarithm and Exponentiation

3.1. SH Logarithm

Our method leverages the 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,

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

Like Eq.11 in the main paper Section 3.3, the SH coefficients of this function can be easily computed using the SH rotation[1] and it is differentiable with respect to $\mathbf{x}$, the sphere center $\mathbf{c_i}$, and radius r_i.

3.2. SH Exponentiation

We follow the same Optimal Linear SH Exponentiation Approximation as Ren *et al*. [2]. Given the SH coefficient vector $\mathbf{f}$ 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 $\mathbf{f}$ in the SH space:

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

after the DC Isolation [2] one obtains $\mathbf{f} = \hat{\mathbf{f}} + exp(\frac{\mathbf{f}_0}{\sqrt{4\pi}})\mathbf{1}$. Ren *et al*. [2] use a tabulation for the function a and b, while we empirically use $a(x) = exp(0.1x)$ and $b(x) = exp(-0.2x)$ which allows differentiating the SH Exponentiation. Further, we also use Scaling/Squaring [2] to scale $\mathbf{f}$ to a suitable range for the SH Exponentiation.

4. Implementation Details

4.1. Optimization

For texture, lighting, and 6D pose reconstruction, we use the Adam optimizer with a learning rate of 10^{-2} for all the methods. For geometry deformation and reconstruction from shadows, we use Adam with a learning rate of 10^{-2} for the 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.

4.2. Initialization

For the texture and lighting optimization, we initialize the texture and environment map as pure dark. 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. 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, there are no meaningful gradients for supervising the scene parameters.

References

[1] 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.

[2] 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.

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

[4] 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.