Efficient and Differentiable Shadow Computation for Inverse Problems

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

Max Planck Institute for Informatics, Saarland Informatics Campus

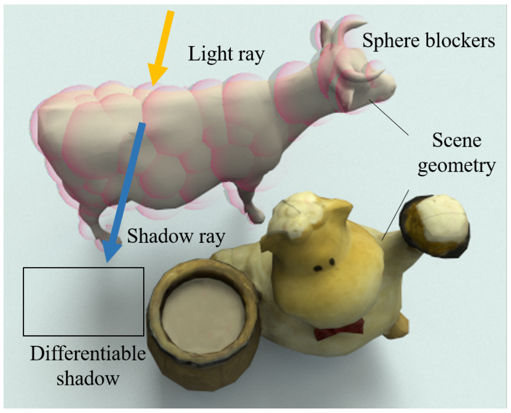

Figure 1. We propose a fast and differentiable visibility approximation for rendering which allows us to account for shadow effects. To this end, we approximate the geometry with a sphere set and compute visibility in the spherical harmonics space.

Abstract

Differentiable rendering has received increasing interest for image-based inverse problems. It can benefit traditional optimization-based solutions to inverse problems, but also allows for self-supervision of learning-based approaches for which training data with ground truth annotation is hard to obtain. However, existing differentiable renderers either do not model visibility of the light sources from the different points in the scene, responsible for shadows in the images, or are very slow which makes it difficult to train deep architectures over thousands of iterations. To this end, we propose an accurate yet efficient approach for differentiable visibility and soft shadow computation. Our approach is based on the spherical harmonics approximations of the scene illumination and visibility, where the occluding surface is approximated with spheres. This allows for a significantly more efficient shadow computation compared to methods based on ray tracing. As our formulation is differentiable, it can be used to solve inverse problems such as texture, illumination, rigid pose, and geometric deformation recovery from images using analysis-by-synthesis optimization.

1. Introduction

Rendering virtual scenes, objects, and characters has a wide range of applications in movies, video games, and many other areas which require synthesis of realistic images. While in these computer graphics applications the main interest is the generation of images from scene parameters like geometry, lighting, and texture, rendering can also be used to solve inverse problems, which attempts to recover exactly these scene parameters from real images. Analysis-by-synthesis optimization is commonly used [36, 3] where the estimated scene parameters are rendered as a synthetic image and then compared with the reference. If the renderer is differentiable, an energy function which compares the renderings and the ground truth images can be used for optimization. Differentiable rendering is also interesting for learning-based, notably neural network-based, approaches where ground truth annotations, e.g. of the dense geometry, are not easily available for large image-based training corpora. Instead, differentiable rendering allows for self-supervised learning using an analysis-by-synthesis approach where the rendered image is compared to the real one. This has been widely used in the vision and machine learning community, for solving problems such as reflectance estimation [1, 4, 20], free-viewpoint synthesis [35, 22], and human performance capture [10, 9, 34, 33].

Most differentiable rendering methods rely on rasterization-based techniques which only consider direct illumination effects [14, 26, 21, 18]. Shadows are global illumination effects, i.e., for any point in the scene, any other point could be occluding the light source, see Fig. 1. Thus, they are not modeled by the direct illumination renderers. As a result, inverse methods supervised with such differentiable renderers produce undesired artifacts. For instance, the estimated texture, geometry, and illumination may exhibit baked in errors trying to represent real world effects, in particular due to shadows which are not accounted for by these simplified rendering assumptions. In this paper, we address the problem of illumination visibility, i.e., whether the light source in a direction is visible from any point in the scene. Illumination visibility is simply called ”visibility” for readability in the paper, not to be confused by camera visibility, i.e., which points of the scene are visible in the image.

To account for these limitations, differentiable ray tracing methods [19, 40] were proposed which use ray tracing methods to solve the rendering equation. Some approaches [1] use them for reconstructing more accurate scene parameters compared to direct illumination-based techniques. While these renderers can render shadows and other higher-order illumination effects, they are computationally inefficient, which makes training large networks difficult on consumer-grade hardware.

To this end, we propose a new method for differentiable and efficient visibility computation for rendering scenes with soft-shadows. Our work builds upon the literature of efficient global rendering [27, 30, 8] where the goal is to approximate visibility for a faster runtime. More precisely, our approach first approximates the scene geometry with spheres. These spheres are attached/rigged to the underlying geometry mesh, which allows for deforming and posing the mesh through the sphere representation. Scene illumination is modeled with the commonly used spherical harmonics representation [25]. Interestingly, the same representation can also be used to model visibility, e.g. whether the incident illumination is occluded in any direction from a point in the scene. This spherical harmonics representation allows for efficient rendering of soft shadows using fast spherical harmonics multiplications. We combine this soft shadow rendering with a diffuse spherical harmonics based shading model to obtain the final rendering which is fully differentiable enabling us to compute gradients with respect to geometry, light, and texture. We show applications of this differentiable renderer, by using analysis-by-synthesis optimization in order to recover the rigid pose, surface deformation, scene illumination, and texture of objects in scenes with shadows. In summary, our contributions are:

- A differentiable and efficient renderer which can synthesize soft-shadows for dynamic scenes.
- The integration of our renderer in an optimization-based setup for the reconstruction of scene parameters from monocular images.

We compare our approach with the state-of-the-art differentiable rendering techniques and show that our method offers a good trade-off between rasterization-based techniques which are efficient but do not model shadows, and the more accurate but inefficient ray tracing approaches.

2. Related Work

Differentiable rendering is a widely studied problem. In this section, we will discuss the most relevant methods. We refer to Kato *et al*. [14] for a recent detailed survey. Existing differentiable rendering methods are either based on efficient but inaccurate direct illumination or more accurate but inefficient global illumination.

2.1. Differentiable Rendering

Efficient but approximate differentiable approaches can be further split into two categories based on the type of approximations. Some works [23, 13, 14, 18] approximate gradients without modifying the rasterization step which has the advantage that camera visibility is modeled as in the real world. However, the camera visibility computation here is non-differentiable. In contrast, there are other works [28, 21] that treat the objects in the scene as semi-transparent volumes. This allows for differentiable camera visibility, but these methods does not reflect the real world properties of the object. All of the above methods do not account for the visibility of the light sources and hence cannot account for cast shadows and self-shadows. This leads to inaccurate results, e.g. the geometry can deform incorrectly to explain shadows in the image, or the recovered textures can contain baked in shadows. In contrast, our proposed approximation to global illumination results in meaningful supervision even in the presence of shadows as we explicitly model them.

Recently, physically-based differentiable rendering methods were proposed [2, 19, 40, 39], which can also account for global illumination. These methods build up on Monte Carlo ray tracing which provides derivatives for arbitrary bounces of light. Li *et al*. [19] proposed the first comprehensive method which can provide gradients for all scene parameters. Zhang *et al*. [40] proposed a similar approach which also accounts for volumetric derivatives along with meshes. Please refer to Zhao *et al*. [41] for a detailed analysis of the various physically-based differentiable rendering methods. While these methods provide accurate supervision with respect to global illumination effects, they are very slow to evaluate as for a single pixel many rays have to be sampled and accumulated. This makes it difficult to use them within the training of neural networks.

2.2. Efficient Global Illumination

Several methods in the computer graphics literature have explored faster global illumination approaches, see the survey of Ritschel *et al*. [29]. Here, precomputed Radiance Transfer (PRT) methods are related to our approach. Most PRT methods [5, 30, 16] assume the geometry to be fixed, although some methods work with dynamic geometry [31, 11]. Very recently, PRT was also used for inverse problems [37]. However, the scene geometry cannot be updated in this formulation. Several approaches have been proposed for efficient computation of soft-shadows for dynamic scenes [15, 42, 17]. Ren *et al*. [27] used spherical

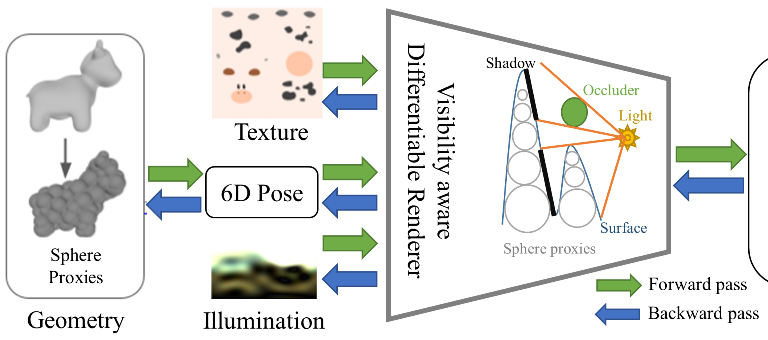

Figure 2. Overview of our approach. Given a surface mesh, we first approximate the geometry surface with a set of spheres. The global visibility can be calculated as a combination of the visibility function for each sphere blocker in the spherical harmonics space, where the function is associated with texture, pose, and illumination. Combined with a rasterizer, an image can be rendered in a differentiable way. Therefore, we are able to optimize the different scene properties, such as geometry, texture, and illumination by comparing the rendered image against the target image.

harmonics (SH) representations for the different scene components such as illumination and visibility and proposed an efficient, but non-differentiable method for computing SH products. Zhou *et al.* [42] proposed shadow fields to represent the light source radiance and occlusions, which allows for fast computation of the visibility. Efficient soft-shadow computation has mostly been explored in computer graphics for creating synthetic imagery. We investigate the inverse problem: using an efficient *differentiable* renderer for estimating scene parameters from monocular images.

3. Method

Our method solves an efficient approximation of the rendering equation for generating soft shadows for diffuse surface meshes. Traditional graphics rendering is extremely time-consuming, mainly due to the expensive sampling process used in ray tracing for computing global illumination. To approach this problem, we represent the geometry surface with a collection of sphere blockers and compute the global visibility as a combination of the visibility function for each sphere blocker (also referred to as blocker function) in the spherical harmonics (SH) space, see Fig. 2. This computation in the SH space is efficient and differentiable, enabling fast rendering as well as optimization of the scene parameters. Our approach is closely related to the work of Ren *et al.* [27] which uses a similar formulation. However, we provide a novel algorithmic formulation, which allows differentiating through the rendering process. Importantly, we show, for the first time, solutions of inverse problems with global effects using an efficient differentiable renderer. In the following, we will first introduce the rendering equation (Sec. 3.1) and its spherical harmonics approximations (Sec. 3.2) as our method builds up on these concepts. We will then describe the rendering process of our approach (Sec. 3.3-3.4) and the optimization of the scene properties with our differentiable renderer (Sec. 3.5).

3.1. Rendering Equation

The rendering equation [12] with only diffuse non-emitting surfaces in the scene is defined as:

$$B(\mathbf{x}) = a(\mathbf{x}) \int_{\Omega} L(\omega, \mathbf{x}) max((\omega \cdot \mathbf{n}(\mathbf{x})), 0) d\omega \,, \quad (1)$$

where $B(\mathbf{x})$ is the outgoing radiance at point $\mathbf{x}$, $L(\omega, \mathbf{x})$ is the incoming radiance at this point from direction ω, Ω represents the sphere of directions, $a(\mathbf{x})$ is the diffuse albedo, and $\mathbf{n}(\mathbf{x})$ is the normal at point $\mathbf{x}$. Assuming no inter-reflectance, the incoming radiance $L(\omega, \mathbf{x})$ at point x can be decomposed into the static environment lighting $L(\omega)$ and the visibility $V(\omega, \mathbf{x})$ at point x. Note that the environment light is only a function of the light direction while the visibility depends on both the light direction and the point location, since it is a global illumination property. Given this, we can rewrite the rendering equation as:

$$B(\mathbf{x}) = a(\mathbf{x}) \int_{\Omega} L(\omega) V(\omega, \mathbf{x}) H(\omega, \mathbf{x}) d\omega \quad , \quad (2)$$

where $H(\omega, \mathbf{x}) = max((\omega \cdot \mathbf{n}(\mathbf{x})), 0)$. One way to compute $B(\mathbf{x})$ is with Monte Carlo integration, which is computationally expensive as lots of samples have to be acquired to approximately evaluate the integral. Instead, we estimate a fast approximation of this integration by using spherical harmonics as introduced next.

3.2. Spherical Harmonics Approximation

A spherical function can be projected into spherical harmonics space and reconstructed back using the spherical harmonics bases. We will write variables corresponding to SH coefficients in bold letters in the following. When representing a function with a subset of basis functions within a limited bandwidth, we obtain a low frequency approximation of the original function. In detail, given a spherical function $f(\omega)$, its corresponding SH vector $\boldsymbol{f} = [\boldsymbol{f}_0, \boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_{n^2-1}]$ (with all SH coefficients stacked) is defined as

$$\boldsymbol{f}_i = \int_\Omega f(\omega) y_i(\omega) d\omega, \tag{3}$$

where $y_i(\omega)$ are the SH basis functions, and n is the number of bands chosen to approximate the signal. The indices are linearized, with $i = l^2 + l + m$, where l is the index of the SH band, and m, $-l \leq m \leq l$ is the index within the band. Computing Eq. 3 requires solving the integral which is usually achieved by Monte Carlo integration. For an inverse problem, this can be very expensive for both forward and backward computations. As a remedy, we precompute several terms which allows for efficient and differentiable computation as we show later. Given the SH coefficients and the basis functions, the original signal can be approximated as

$$\tilde{f}(\omega) = \sum_{i=1}^{n^2} \boldsymbol{f}_i y_i(\omega), \tag{4}$$

where i is the linearized index explained above. The accuracy of reconstruction increases with the number of bands. In our case, we choose a bandwidth of $n = 8$, which we found to be good for low-frequency approximations. The integral of a multiplication of two SH functions can be simply computed as a dot product of their coefficients,

$$\int_\Omega \tilde{a}(\omega)\tilde{b}(\omega) d\omega = \boldsymbol{a} \cdot \boldsymbol{b}\,. \tag{5}$$

The SH projection of this multiplication is computed as

$$\boldsymbol{a} * \boldsymbol{b} = \int_\Omega \tilde{a}(\omega)\tilde{b}(\omega) y(\omega) d\omega, \tag{6}$$

$$(\boldsymbol{a} * \boldsymbol{b})_i = \sum_{j=1}^{n}\sum_{k=1}^{n} C_{ijk} \boldsymbol{a_j} \boldsymbol{b_k}, \tag{7}$$

where $C_{ijk} = \int_\Omega y_i(\omega) y_j(\omega) y_k(\omega)$ is a transfer matrix which can be precomputed. Please refer to Green [7] for the *gritty details*. In our case, we first compute the SH coefficients for L, V and H in Eq. 2. With that, the solution of $B(\mathbf{x})$ can be analytically computed as

$$B(\mathbf{x}) = a(\mathbf{x}) \Big(\boldsymbol{L} \cdot (\boldsymbol{V}(\mathbf{x}) * \boldsymbol{H}(\mathbf{x})) \Big)\,. \tag{8}$$

This way of computing the outgoing radiance does not require inefficient sampling-based integration, which can also be sensitive to the sampling strategy used. Next, the efficient computation of these SH coefficients is explained.

3.3. Spherical Harmonics Computations

We first compute $\boldsymbol{L}$ and $\boldsymbol{H}(\mathbf{x})$ as

$$\boldsymbol{L} = \int_\Omega L(\omega) y(\omega) d\omega\,, \tag{9}$$

$$\boldsymbol{H}(\mathbf{x}) = \int_\Omega max((\mathbf{n}(\mathbf{x}) \cdot \omega), 0) y(\omega) d\omega\,. \tag{10}$$

We solve Eq. 9 using numerical integration at initialization. This equation does not have to be differentiable, since we only want to update the SH coefficients of the environment map. For Eq. 10, we formulate the computation of $max((\mathbf{n}(\mathbf{x}) \cdot \omega), 0)$ as

$$\boldsymbol{H}(\mathbf{x}) = \phi_{\mathrm{SH}}(\mathbf{n}(\mathbf{x})) \int_\Omega max((i_z \cdot \omega), 0) y(\omega) d\omega\,. \tag{11}$$

Here, $\phi_{\mathrm{SH}}(\mathbf{n}(\mathbf{x}))$ is a matrix which defines the rotation of spherical harmonic functions [16], and $i_z = (0, 0, 1)^\top$. This reformulation has several advantages. First, this makes the equation differentiable with respect to $\mathbf{n}(\mathbf{x})$. Second, spherical harmonics projections of functions which are symmetric along the z-axis can be computed analytically and efficiently. This leaves us with the visibility term.

3.3.1 Sphere Fitting and Geometry Deformation

Our method approximates the geometry of the non-deformed initial mesh with a set of spheres for computing the visibility, see Fig. 2. The mesh can then be deformed by translating and rotating these spheres. We can thus also update the geometry of the mesh using our renderer. Visibility related rendering effects like shadows will supervise the sphere parameters and therefore the underlying geometry. To determine the initial position and radius of each sphere for a given geometry, we minimize the total volume occupied by the sphere set which is outside the mesh, referred to as sphere outside volume [38] along with a term encouraging the sphere set to cover as much of the mesh surface as possible. This objective is optimized using gradient descent. Please refer to the supplemental document for more details. We then connect the sphere centers as an embedded graph to drive the deformation of the mesh geometry [32] where each sphere is connected to its K-nearest vertices on the template. The mesh deformation is parameterized by the rotation and translation of each sphere. Later, the inverse problems optimize the rotation and translation of the spheres using the rendering loss (see Eq. 15).

3.3.2 Visibility in Spherical Harmonics Space

Similar to Zhou *et al.* [42], we calculate $\mathbf{V}(\mathbf{x})$ as the product of the SH vectors of all the sphere blockers:

$$\boldsymbol{V}(\mathbf{x}) = \boldsymbol{V}_1(\mathbf{x}) * \boldsymbol{V}_2(\mathbf{x}) * \ldots * \boldsymbol{V}_n(\mathbf{x})\,. \tag{12}$$

where $\boldsymbol{V}_i(\mathbf{x})$ is the SH vector of the blocker function of the sphere S_i, which measures the blocking effect of a sphere:

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

Similar to Eq. 11, the computation of $\boldsymbol{V}_i(\mathbf{x})$ can be reparameterized to be differentiable with respect to $\mathbf{x}$. The higher-order product in Eq. 12 can be computed using a series of SH multiplications as defined in Eq. 6. However, this can be computationally expensive. To accelerate this computation, we adopt the method proposed by Ren *et al*. [27], where the exponential of logarithm of SH functions are computed instead. The logarithm of the product leads to a summation, which can be computed efficiently. We follow their scaling, squaring, and optimal linear approximation setting. Ren *et al*. [27] compute the logarithm of the SH functions based on a lookup table which is not differentiable with respect to the sphere orientations. Instead, we propose a differentiable approach which works well for our setting. We first approximate V_i as $V_i^{'}(\omega, \mathbf{x})$:

$$V_i^{'}(\omega, \mathbf{x}) = \begin{cases} e^{-\epsilon}, & \text{if } S_i \text{ blocks in direction } \omega: \\ 1, & \text{otherwise.} \end{cases} \tag{13}$$

ϵ is set to 3 such that $e^{-3} \approx 0.05$. This avoids the infinite logarithm for 0. We orient $\log(V_i^{'}(\omega, \mathbf{x}))$, the logarithm of this approximated function for each sphere, to the z axis and project it to SH space. This computation has an analytical form with respect to the distance to the sphere and its radius. Then, we apply the SH rotation as in Eq. 11 with the vector pointing to the sphere center from $\mathbf{x}$. Finally, we add them and exponentiate the result to compute $\boldsymbol{V}(\mathbf{x})$. Please refer to the supplemental for more details.

3.4. Rendering

Combined with a rasterizer, we can render an image as

$$I_R = R(B(\mathbf{x}_0), \ldots, B(\mathbf{x}_i), \ldots, B(\mathbf{x}_n), P) \tag{14}$$

where I_R is the image intensity, $B(\mathbf{x}_i)$ is the radiance of vertex $\mathbf{x}_i$ as computed in Eq. 2, P is a projection matrix implementing the camera using its intrinsics and extrinsics and R is the rasterization function. The radiance $B(\cdot)$ includes shadows as well as diffuse shading of the surface.

3.5. Image-Based Optimization

We optimize the different scene properties, such as geometry, albedo, and illumination by comparing the rendered image to the reference image. In all experiments, we use the ℓ_2 loss function between the rendered and reference image for optimization. The objective function can be written as:

$$\mathcal{L}(\theta, \boldsymbol{L}, a_{[0,n]}) = ||I_R(\theta, \boldsymbol{L}, a_{0,\ldots,n}) - I||_2^2\,, \tag{15}$$

$$I_R(\theta, \boldsymbol{L}, a_{[0,n]}) = R(B(D_{\mathbf{x}_0}(\theta)), \ldots, B(D_{\mathbf{x}_n}(\theta)), P)\,.$$

Here, $D_{\mathbf{x}}(\theta)$ is a function that takes the global rigid pose and embedded deformation [32] parameters θ of an object in the scene and poses and deforms the respective vertex $\mathbf{x}$.

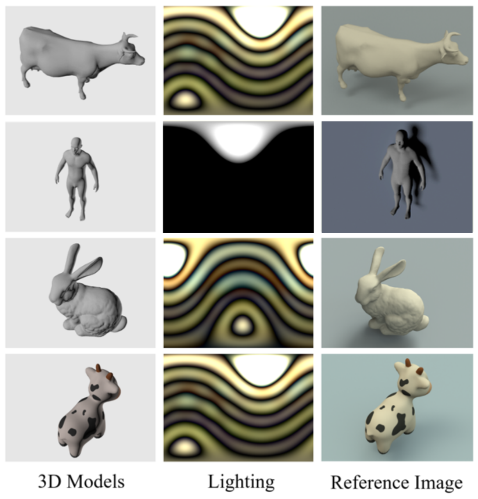

Figure 3. Example scenes of our evaluation dataset. Note that geometry as well as lighting conditions are very complex making it a challenge for inverse problems.

I is the reference image. We can also optimize for a texture map by re-parameterizing a in 2D as a pre-process. If there are multiple objects in the scene, they can have different rigid poses. This loss function is differentiable since we use a differentiable rasterizer [26] as $R(\cdot)$, and our radiance computation $B(\mathbf{x})$ is differentiable with respect to θ, $\boldsymbol{L}$ and $a_{0,..,n}$. We use gradient descent with a step size of 10^{-2} for optimization with $200 - 600$ iterations.

4. Results

Our method is implemented in Pytorch [24]. We use an Intel(R) Xeon(R) Gold 6144 CPU and an Nvidia V100 graphics card for all results.

Evaluation Dataset. To evaluate our approach, we create different virtual scenes containing various challenging geometries, textures, and lighting conditions. In total, we use 7 different mesh models, e.g. animals and humans. We acquired 10 environments maps [6] containing various lighting conditions, e.g. outdoor skies. These environment maps are projected onto the SH space. We use a ray tracing approach [19] to render the reference images, where we choose 1024 sample rays per pixel to obtain a noise free reference. Fig. 3 shows some examples.

4.1. Inverse Problems

We conduct several experiments where the individual scene parameters are reconstructed from monocular images using analysis-by-synthesis optimization (Sec. 3.5). We compare to Redner [19], which models global illumination using ray tracing, and to the implementation of Ravi *et al*. [26] of a rasterization-based direct illumination (DI) method [25], which uses SH shading without shadows. For

Texture Reconstruction Accuracy	
Method	**MSE↓**
Redner [19]	**0.0213**
DI [26]	0.2763
Ours	0.0272

Table 1. Texture reconstruction accuracy averaged over 10 scenes. We quantitatively outperform DI method using mean squared error, as shadows are baked in the texture for these approaches.

Redner, we set the number of rays per pixel to 64 for all experiments, which was the smallest number that still gave us noise free renderings. We use single bounce, as we do not evaluate indirect illumination effects.

Texture Optimization. First, we evaluate our approach for the purpose of texture estimation. Given the geometry, lighting, and one reference image of the scene, we optimize for the diffuse albedo texture, initialized as pure white, using the different rendering methods. Note that the reference images shown in Fig. 4 contain shadows cast by the occluders, as well as self-shadows which makes it particularly challenging to recover the correct albedo. In the first row, the texture of the cow is optimized. However, a large sphere (not visible in the image) is also placed in the scene blocking the light coming from a large area of the environment map. In the second row, the texture of the ground plane should be recovered onto which the armadillo is casting a shadow. In both cases, our approach recovers albedo which is almost free from shadows, due to our shadow-aware renderer. Note that DI method [26] cannot account for the shadow which manifests in the bright initialization (top row), as the occluding sphere is ignored and all the light arrives at the surface of the cow. More importantly, these methods bake the shadows into the texture. In contrast, Redner [19] also obtains shadow-free textures but at a significantly slower speed. This is also quantitatively evaluated in Tab. 1 where we measure the mean squared pixel error (MSE) between the optimized and the ground truth texture maps for all pixels that are visible in the rendered image. Although Redner performs slightly better, our approach is very close in quality while being much faster. Our method clearly outperforms the direct illumination renderer.

Lighting Optimization. Next, we evaluate our approach in terms of lighting reconstruction. Here the geometry and albedo texture of the object are known, and we are interested in reconstructing the lighting from a reference image. The reference images are shown in the left column of Fig. 5. We initialize all methods with no lights, i.e. all SH coefficients are set to zero. The converged results are shown in Fig. 5 for two different scenes. Moreover, we relight new objects with the optimized scene lighting, and compare it to the ground truth for an additional evaluation of the estimate. We do not directly compute quantitative errors on the estimated environment maps because it is not easy to compute the visible regions in the environment map. Computing the error on the full environment map would not be indicative of the quality, as the occluded regions could be arbitrarily different for each method. Our indirect metric using a different object does not face this issue. We outperform the DI method, as they cannot model the shadows. As expected, the ray tracing method gives the most accurate results at the cost of a slow runtime. In Tab. 2, we further quantitatively evaluate the accuracy of the lighting estimation *on different objects* in terms of mean squared error (MSE) by comparing the scene rendering with the ground truth. Our method offers a good compromise between the more accurate but slow ray tracing approach and the more efficient but less accurate direct illumination method.

Lighting Estimation Accuracy	
Method	**MSE↓**
Redner [19]	**1.854e-3**
DI [26]	9.512e-3
Ours	2.667e-3

Table 2. Lighting estimation accuracy averaged over 10 different scenes. Here, we evaluate the mean squared error between the rendered and ground truth images. Our approach outperforms the direct illumination method, and is close to the ray tracing method.

6D Pose Optimization. Here, we optimize the 6D pose of the object with known light and texture, see Fig. 6 and Tab. 3 from one image. Our renderer provides useful supervision for correctly optimizing the rigid pose which can be seen qualitatively as well as quantitatively. Compared to previous methods, we clearly outperform DI method as they cannot use the shadow cues as signal. In contrast, the cast shadows directly provide supervision for the unknown pose in our case. Interestingly, we also outperform the ray tracing approach [19]. We hypothesize that it is difficult for ray tracing renderers to optimize the geometry because of their local nature of computation. The gradients at any point in the scene are propagated through the rays which reach this point. Thus, the gradients mostly rely on local properties of the scene. In contrast, the shadow computation at any point in our renderer directly depends on the global scene geometry, which leads to less noisy gradients.

Geometry Optimization. In Fig. 7 and Tab. 4, we evaluate how our renderer can be used for estimating the geometric deformation of an object, given the *undeformed* template as well as the lighting and albedo of the scene. To this end, the rotation and translation parameters of the embedded graph are optimized. Like in the case of global

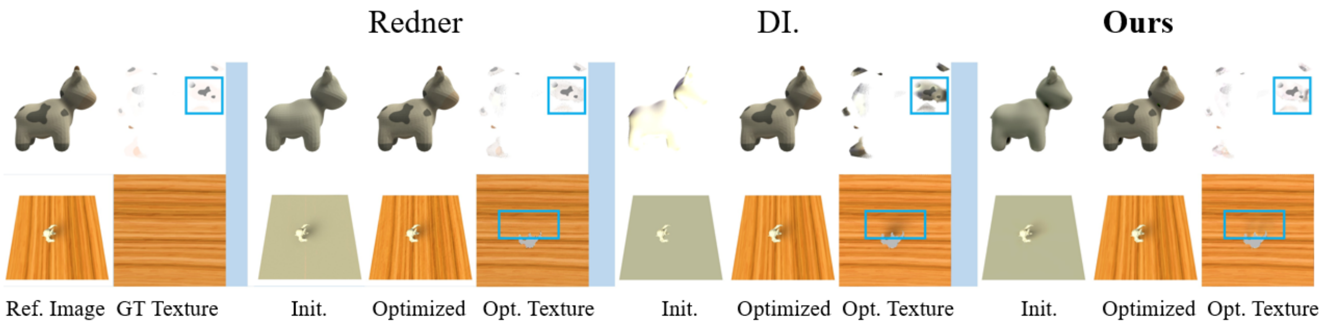

Figure 4. Texture optimization results. From left to right. Ground truth rendering and texture map. Rendering with initial and optimized texture map as well as the optimized texture map for Redner, DI and our method. Note that our approach outperforms DI method as they cannot remove the shadow in the texture and we are also close to Redner [19] while being much faster.

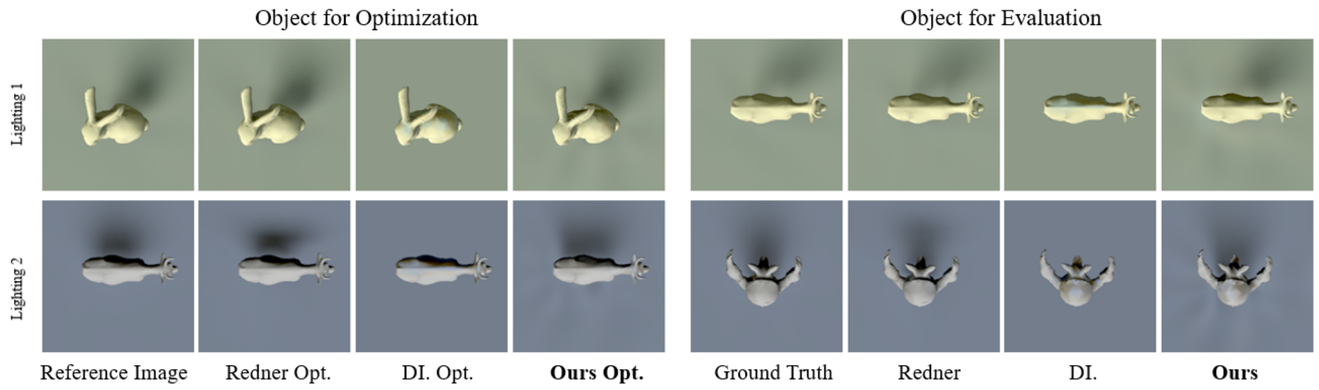

Figure 5. Lighting optimization results. From left to right. The reference image used to optimize the scene lighting. The rendered scene with the optimized scene lighting. The ground truth image for a new object. The rendering of the new object using the previously optimized scene lighting. Note that our approach gives a good compromise between runtime and accuracy, compared to other approaches.

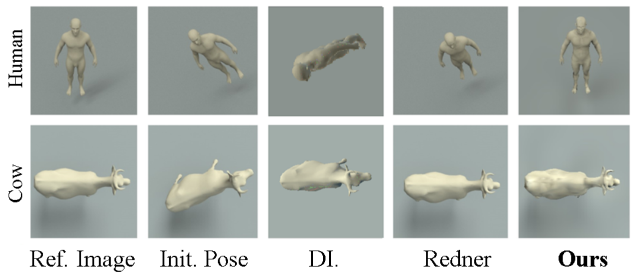

Figure 6. 6D Pose optimization result. We outperform both direct illumination and global illumination methods.

Pose Estimation Accuracy			
Method	**MDE ↓**	**RE ↓**	**TE ↓**
Redner [19]	12.72	0.2289	**4.633e-4**
DI [26]	23.53	1.0154	7.719e-2
Ours	**6.420**	**0.0702**	1.075e-3

Table 3. Pose reconstruction accuracy averaged over 10 different scenes. The mean distance error is calculated between the ground truth and optimized vertex positions. We report MDE as a percentage of the diagonal length of the bounding box around the ground truth shape. Rotation error (RE) is calculated as the angle between the rotations, and translation error (TE) as the squared distance between the translations. We outperform both approaches.

pose reconstruction, our approach outperforms the previous works [19, 26] both qualitatively and quantitatively.

4.2. Runtime

We evaluate the runtime of our approach and compare it with the state of the art. Tab. 5 reports the average runtime per iteration in milliseconds (ms) and frames per second (fps). Our approach is significantly (up to two orders of magnitude) faster than the ray tracing approach [19]. We are close to the runtime of the method of Ravi *et al*. [26] for most tasks except for pose and geometry estimation. This shows that our shadow computation requires a minimal overhead compared to DI method, with the advantage of higher-quality reconstruction of all scene properties.

4.3. Reconstruction from Shadows

In Fig. 8, we show how the shadow alone can be used to optimize the object deformations. We optimize the embedded graph deformation parameters and compare the rendered shadows with the reference shadow image. This

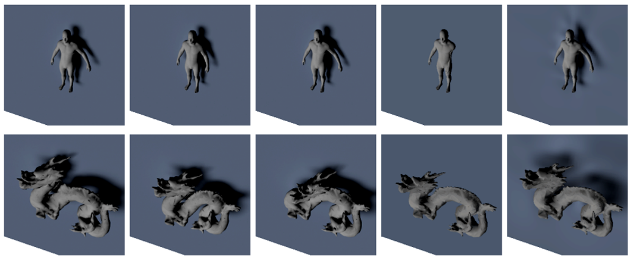

Figure 7. Geometry optimization results. We outperform both direct illumination and global illumination methods.

Geometry Reconstruction Accuracy	
Method	**MDE ↓**
Redner [19]	18.66
DI [26]	28.98
Ours	**11.80**

Table 4. Geometry reconstruction accuracy averaged over 10 different scenes. We outperform both approaches in terms of MDE (explained in Tab. 3).

Runtime Performance (in ms)↓				
Method	**Texture**	**Light**	**Pose**	**Geometry**
Redner [19]	3261	2938	4473	4897
DI [26]	**18**	**18**	**81**	**233**
Ours	**18**	**18**	216	391

Table 5. Runtime performance averaged over multiple iterations including forward and backward passes. Our approach clearly outperforms the ray tracing renderer, and is close to the DI method, indicating that our shadow computation is efficient.

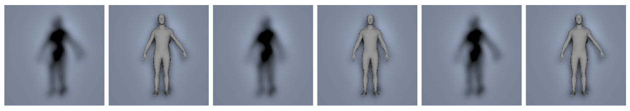

Figure 8. Shadow fitting result. The shadow alone can be used to recover the geometric deformations.

demonstrates that shadows offer very strong cues about the scenes. Our method, in contrast to direct illumination renderers can utilize these cues for accurate reconstruction.

4.4. Ablation

In Tab. 6, we study the influence of the number of spheres used to approximate the underlying geometry of the object. We solve for lighting reconstruction, see Fig. 5. Even a small number like 100 spheres can approximate the geometry quite well. Adding more spheres improves the quality of reconstruction. We also evaluate the influence of the number of spherical harmonics coeffcients while fixing the number of spheres to 100 using the same setting. A small number of coefficients can already approximate the scene lighting quite well. Adding more coefficients leads to better results as higher frequency lighting can be captured as well. Importantly, for all experiments with different numbers of spheres and lighting coefficients, we achieve a runtime of 18ms due to GPU parallelization.

Ablation Study				
Spheres	50	100	150	200
MSE↓ $\times 10^{-3}$	1.445	1.349	1.336	**1.335**
Coeffs	16	25	36	49
MSE↓ $\times 10^{-3}$	2.437	1.982	1.629	**1.493**

Table 6. Ablation study. Even a small number of spheres and lighting coefficients can lead to plausible results. Adding more spheres and lighting coefficients constantly improves the result as surface details can be better approximated and higher frequency lighting can be modeled.

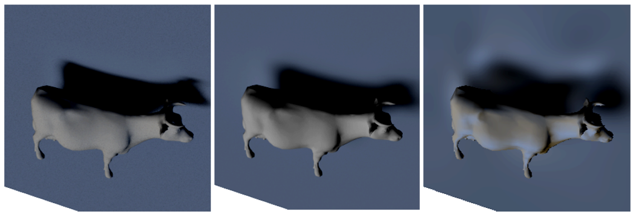

Figure 9. Limitation. Our method fails to reconstruct high-frequency lighting, such as a directional light in this example.

5. Discussion

High frequency lighting cannot be modeled well by our approach due to the low-dimensional SH representation, see Fig. 9, and we assume distant illumination and diffuse materials. Our method requires the precomputation of the embedded deformation graph. This graph is well-suited for one object category, however, it would not be sufficient to deform very different shapes. Our approach does not consider inter-reflectance and non-diffuse surfaces, which are interesting directions for future work.

6. Conclusion

We proposed a method for efficient and differentiable shadow computation that can be used for various inverse problems. We show that our approach achieves competitive results compared to ray tracing methods at much faster runtimes. Further, we outperform direct illumination renderers which do not model shadows. We demonstrate that shadows are important cues in images, and take the first steps towards using efficient high-quality differentiable and shadow-aware rendering for inverse problems.

Acknowledgements. This work was supported by the ERC Consolidator Grant 4DReply (770784) and the Lise Meitner Postdoctoral Fellowship.

References

[1] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[2] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020.

[3] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 187–194, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[4] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 37(128):15, aug 2018.

[5] Yoshinori Dobashi, Kazufumi Kaneda, Hideki Nakatani, Hideo Yamashita, and Tomoyuki Nishita. A quick rendering method using basis functions for interactive lighting design. In *Computer Graphics Forum*, volume 14, pages 229–240. Wiley Online Library, 1995.

[6] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. 2017.

[7] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the Game Developers Conference*, volume 56, page 4, 2003.

[8] P. Guerrero, S. Jeschke, and M. Wimmer. Real-Time Indirect Illumination and Soft Shadows in Dynamic Scenes Using Spherical Lights. *Computer Graphics Forum*, 2008.

[9] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[10] Marc Habermann, Weipeng Xu, Michael Zollhöfer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM Trans. Graph.*, 38(2):14:1–14:17, Mar. 2019.

[11] Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 35–44, 2007.

[12] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, Aug. 1986.

[13] Hiroharu Kato and Tatsuya Harada. Learning view priors for single-view 3d reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Jan Kautz, Jaakko Lehtinen, and Timo Aila. Hemispherical rasterization for self-shadowing of dynamic objects. *Rendering Techniques*, 2004:15th, 2004.

[16] 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.

[17] Janne Kontkanen and Samuli Laine. Ambient occlusion fields. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 41–48, 2005.

[18] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering, 2020.

[19] 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.

[20] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2475–2484, 2020.

[21] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019.

[22] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[23] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Computer Vision – ECCV 2014*, volume 8695 of *Lecture Notes in Computer Science*, pages 154–169. Springer International Publishing, Sept. 2014.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 497–500, New York, NY, USA, 2001. Association for Computing Machinery.

[26] 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.

[27] 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.

[28] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the 2015 International Conference on Computer Vision (ICCV 2015)*, 2015.

[29] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. In *Computer Graphics Forum*, volume 31, pages 160–188. Wiley Online Library, 2012.

[30] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, 2002.

[31] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (TOG)*, 24(3):1216–1224, 2005.

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

[33] Ayush Tewari, Michael Zollöfer, Florian Bernard, Pablo Garrido, Hyeongwoo Kim, Patrick Perez, and Christian Theobalt. High-fidelity monocular face reconstruction based on an unsupervised model-based face autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018.

[34] Ayush Tewari, Michael Zollöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Theobalt Christian. MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[36] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[37] Daniel Thul, Vagia Tsiminaki, L'ubor Ladickỳ, and Marc Pollefeys. Precomputed radiance transfer for reflectance and lighting estimation. In *2020 International Conference on 3D Vision (3DV)*, pages 1147–1156. IEEE, 2020.

[38] 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.

[39] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4):143:1–143:19, 2020.

[40] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. A differential theory of radiative transfer. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[41] Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. Physics-based differentiable rendering: From theory to implementation. In *ACM SIGGRAPH 2020 Courses*, SIGGRAPH 2020, New York, NY, USA, 2020. Association for Computing Machinery.

[42] Kun Zhou, Yaohua Hu, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Precomputed shadow fields for dynamic scenes. In *ACM SIGGRAPH 2005 Papers*, pages 1196–1201. 2005.