Translucent Material Parameter Estimation

Number: 7

Abstract

In this paper, we present a workflow for optical material parameter estimation based on real-world images, numerical optimization, and differentiable rendering (building upon the publicly available *Mitsuba 3* code). In this work, we primarily focus on *translucent* materials and demonstrate our approach's applicability to both synthetic test cases and real-world specimens. Specifically, we estimate parameters for two well-known material models: either a *Principled BSDF* (a surface model based on Burley's Disney BSDF), or a *Rough dielectric BSDF*, describing volumetric *homogeneous* participating media.

In order to solve the inverse rendering problem of acquiring suitable material properties from a (set of) given reference image(s), we present a software tool that handles the entire material reconstruction workflow. Given a virtual 3D scene description and multiple images, our tool provides a user interface to select the optimization parameters, a gradient-based optimization algorithm, provides different formulations of the optimization objective, and allows the user to inspect the resulting material description compared to a given reference image.

Finally, we also propose an experimental workflow to acquire the necessary reference images and potentially also the 3D scene geometry itself. Our results show that our approach works on well-known materials such as acrylic glass, as well as newly designed materials, such as alginate, from experimental materials research.

Keywords: Numerical Optimization, Material Parameter Estimation, Differentiable Rendering, Appearance Modelling

1 Introduction

Early work in computer graphics relied on phenomenological, often physically implausible, shading models and focused primarily on rendering images under direct, or purely diffuse illumination [32, 3, 33]. The driving force behind many rendering methods was artistic expression rather than physical accuracy. However, with more powerful hardware supporting modern ray tracing algorithms [30, 16], photo-realistic and physically accurate rendering has become not only feasible but de-facto standard in many fields, both within and outside of computer graphics.

The main goal of physics-based rendering is to create images matching the way our eyes perceive the world by computational means, given a virtual scene description. The ability to create images that are hard to separate from reality, however, hinges on accurate knowledge of the optical material parameters of objects in the scene.

Figure 1: Reconstruction results for a surface BSDF model from synthetic data (left) and real-world alginate material (right). Dragon model by Delatronic [6], License: CC-BY.

Scattering occurring at the surface of objects is modelled by the *bidirectional scattering distribution function* (BSDF). Statistical micro-surface models provide physics-based, parametric BSDF formulations that describe the material properties of an object in the scene. For translucent materials, ray tracing methods, as well as material descriptions, have been extended to cover *volumetric scattering* in addition to surface effects.

While both surface and volumetric BSDF parameters can be obtained from careful measurements of real-world materials, or instead formulated in a data-driven (rather than a parametric) framework, these approaches heavily rely on specialized laboratory equipment. However, this precise approach might not always be feasible or cost effective. For example, research in materials engineering might produce new compounds for specific applications, whereas aging and weathering effects might alter a material's appearance over time.

Recent work on differentiable and inverse rendering, on the other hand, has enabled the identification of optical material parameters by applying (gradient-based) optimization methods in order to match the rendered result to an object's real-world appearance. This optimization approach is particularly suited to estimating parameters from only a few reference images. In this work, we rely on the *Mitsuba 3* renderer [11], a publicly available, state-of-the-art differentiable rendering system, to solve material parameter estimation problems. However, as a research-oriented system, Mitsuba 3 requires a lot of domain-specific knowledge to operate. To facilitate the parameter estimation workflow, we implement a software tool that combines a gradient-based optimization algorithm with the necessary data management and rendering interface.

In the remainder of this paper, we first provide an overview of previous work on physics-based (inverse) rendering and accumulate relevant background information. The subsequent sections then detail the experimental and computational aspects of our proposed approach, and present and analyze our results from both synthetic and real-world data. In summary, our contributions are:

- A software tool that incorporates the full power of the Mitsuba 3 renderer and various features to accomplish the task of inverse rendering.
- A gradient-based optimization procedure to acquire material properties from an image, including a careful evaluation of how optimization hyper-parameters and rendering quality affect the results based on synthetic ground-truth comparisons.
- A workflow to increase the reproducibility of our results, encompassing both synthetic as well as real-world examples.

2 Related Work

Physics-based rendering has its origins in the seminal paper by Kajiya [13], formulating the now well-known rendering equation. Veach [36] introduced the path-integral form, which provides the theoretical foundation of modern Monte Carlo (MC) ray tracing methods [10, 29, 5, 31]. In order to arrive at a photorealistic rendering, an accurate description of how various materials interact with light must be formulated. Different material models have been introduced to account for the light-scattering effects from surfaces [38, 9, 24]. Similarly, models for participating media describe volumetric scattering effects as light passes through the material [31, 24]. Physics-based rendering itself (which we also refer to as *forward* rendering in this context) is concerned with computing accurate solutions to the rendering equation, given a scene description, including all materials.

Conversely, the question for *inverse* rendering is how to find an appropriate scene description in order to obtain a desired result. In particular, finding the parameters of a scattering model corresponding to a real-world material, has been a long-standing research problem. While some previous work has addressed this problem in a data-driven way [22, 17, 34], inverse rendering methods in contrast, seek to determine the parameters of established material models, resulting in a more constrained search space that ensures the use of physically sound models. Early work on inverse rendering for material parameter estimation [8] used material dictionaries, aiming to approximate the appearance of real-world materials as a combination of dictionary entries. More recently, advances in differentiable rendering [28, 37, 27, 18, 41] have enabled inverse rendering applications by applying gradient-based optimization directly on the material parameters.

Initially, differentiable renderers have applied code-level automatic differentiation to find gradients of either the rendered image, or an objective function defined on that image, with respect to input parameters. In order to reduce the computational cost and memory footprint of automatic differentiation, analytic back-propagation formulations have been developed [28, 37]. One issue that has received a lot of attention in recent research, is that derivatives of pixel values (which are integrated according to the rendering equation) can contain discontinuous integrands if a silhouette edge moves across the pixel due to a change of scene parameters [18, 40, 19, 42]. As we are primarily concerned with material parameters of standard material models, these discontinuities do not affect our results. Note that, however, this is not the case for geometry reconstruction-related experiments (shown in §5.2).

Modern ray tracing renderers use importance sampling to reduce noise due to MC sampling, based on either the material's scattering behaviour, the light sources, or combining both through multi-importance sampling. Consequently, similar sampling strategies have also been investigated for differentiable ray tracing. Zeltner et al. [40] analyzed numerous potential approaches and mathematically classified various estimators for differential light transport. Prior to the work by Vicini et al. [37], many techniques used statistically biased gradient estimation methods. Their work proposed a new back-propagation algorithm that runs in constant memory and linear computation time. Additionally, this method provides a way to handle highly specular materials such as smooth dielectrics and conductors. Our work builds upon their method, which is implemented in Mitsuba 3 [11].

Inverse Rendering for translucent material reconstruction. The work from Gkioulekas et al. [8] is one of the initial inverse rendering research regarding the reconstruction of homogeneous translucent materials. They introduced an optimization framework to measure the bulk scattering properties of homogeneous materials. Their primary focus was on the parameters in which the homogeneous materials are described by two scalar values and one angular function: scattering coefficient, absorption coefficient and phase function. Similar to our project, Gkioulekas et al. [8] also incorporated gradient-based optimization with MC rendering. However, they specifically used stochastic gradient descent for the optimization. Moreover, in their optimization procedure, they used a material dictionary to invert the radiative transfer equation. The material dictionary contained a variety of common materials, including liquid, solid, and publicly-available collections of tabulated phase functions.

Yang et al. [39] proposed another inverse rendering approach for heterogeneous translucent materials from a single input photograph. The proposed method can obtain the material distribution and estimate heterogeneous material parameters to render images similar to the provided reference image. Additionally, as in our second approach, they introduce an iteration process for optimizing the scattering coefficient and absorption coefficient, which they describe the sum as the extinction coefficient. Deng et al. [7] introduced another approach to reconstruct translucent

objects using differentiable rendering. They extended previous methods using a bidirectional scattering-surface reflectance distribution function (BSSRDF) for translucent materials. Moreover, to handle the noise introduced by the BSSRDF integral, they proposed a dual-buffer method for evaluating the loss during optimization. In this work, we make use of the dual-buffer method and observe improved optimization convergence over standard error metrics (see our results in §5).

3 Parameter estimation

3.1 Background and theory

In this section, we briefly summarize the underlying theory of the rendering process itself, the material models we use for our results, and finally the core principle of differentiable ray tracing. Let us start with the well-known rendering equation as formulated by Kajiya [13], also referred to as the light transport equation (LTE). As we are primarily interested in translucent materials, we consider the spherical form of the LTE, see also [31, 13, 12]:

$$L(p,\omega_o) = \underbrace{L_e(p,\omega_o)}_{Emission} + \underbrace{\int_{S^2} f(p,\omega_i,\omega_o) L_i(p,\omega_i) d\omega_i}_{Scattering}. \quad (1)$$

Here $L(p,\omega)$ describes the radiance (power per area, solid angle, and color channel) leaving a point p in direction ω due to light emitted at or scattered through p. The incident radiance results from recursively applying the LTE such that $L_i(p,\omega_i) = L(p',-\omega_i)$, where p' is a point directly visible from p in direction ω_i. Monte Carlo (MC) ray tracing approximates this recursive integral as a sum of randomly sampled scattering rays. The same concept has been extended to volumetric rendering, including scattering and attenuation in participating media, please refer to PBRT [31] for details.

Most importantly for us, the function $f(p,\omega_i,\omega_o)$ encodes the material's optical properties, and is commonly referred to as the *bidirectional scattering distribution function* (BSDF). For convenience of notation, we assume the cosine term accounting for the projected solid angle is included in the BSDF. The BSDF is composed of a reflective part, the bidirectional reflectance distribution function (BRDF) and a transmissive part, the bidirectional transmittance distribution function (BTDF).

Material parameters. For surface-only materials, we use the *Disney Principled BSDF* implementation as described by Burley [9], focusing on the following parameters:

- base colour (albedo) c (can be textured),
- surface roughness α,
- specular transmission σ_s (blending between the BRDF and BTDF lobes), and the
- index of refraction η.

Note that we do not use secondary reflective lobes such as clear-coat materials or metallic reflections.

For volumetric rendering, we use Mitsuba's *Rough Dielectric BSDF* with a *homogeneous participating medium*. This model extends the surface BSDF by

- a volumetric albedo colour (texture) and
- an extinction coefficient σ_t describing the absorption as light traverses the material.

Inverse and differentiable Rendering. So far we have summarized how a virtual scene, including material descriptions, can be rendered via ray tracing. We now move on to the problem of inverse rendering, where we aim to find scene parameters, such that the resulting image fulfills certain requirements. In particular, our problem is finding the parameters of the material models, such as to match given reference images. Inverse rendering is commonly defined as an optimization problem of the form

$$\min g(y(x)),\ \text{s.t. } h(x) \leq 0, \quad (2)$$

where g defines the optimization objective (or loss) function on the rendered image y given scene parameters x, and h defines additional constraints (such as, for example, min/max parameter values).

In order to solve this optimization problem more efficiently, differentiable rendering provides derivatives ($\delta y/\delta x$), which allows gradient-based optimization techniques to successively improve the parameters with respect to specified objective, see also [42]. In the simplest form, the objective takes the L2 norm of the difference between the rendered and the reference image (interpreting the sequence of pixel values as a vector), thus $g(y) = ||y - y_{ref}||^2$. The *dual buffer method* by Deng et al. [7] proposes an alternative objective function that is better suited to handle the noise introduced by the MC rendering. Although we work with a slightly different BSDF model than their work, we still find this loss formulation useful in practice. Rather than taking one differentiable rendering step, the dual buffer method takes two independent rendering steps y_1, y_2, per iteration. Instead of the standard L2 loss, we now compute the loss in each iteration as $g(y_1,y_2) = (y_1 - y_{ref}) \cdot (y_2 - y_{ref})$.

Finally, in order to acquire the gradients required for optimization, Mitsuba 3 implements a path-replay backpropagation method, see also Nimier-David et al. [28], and Vicini et al. [37]. Formally, we take the derivative of Eq. (1) with respect to the scene parameters x:

$$\begin{aligned}\delta_x L_o(p,\omega_o) = &\underbrace{\delta_x L_e(p,\omega_o)}_{Emission} \\ &+ \int_{S^2} [\underbrace{\delta_x L_i(p,\omega_i) f(p,\omega_o,\omega_i)}_{Transport} \\ &+ \underbrace{L_i(p,\omega_i)\delta_x f_x(p,\omega_o,\omega_i)}_{Material}] d\omega_i.\end{aligned} \quad (3)$$

This equation describes the scattering of derivatives analogous to the LTE. The individual terms are:

- Emission: differential radiance is emitted when the emitted radiance L_e depends on x.

- Transport: differential radiance *scatters* in the same way as normal radiance, according to the BSDF f.
- Material: surfaces with a parameter-dependent BSDF emit differential radiance proportional to incident radiance.

Note that along each ray, we find $\delta L_i(p, \omega_i) = \delta L(p', -\omega_i)$, analogous to the forward rendering. While in an abstract sense, the gradient of the objective function could be computed as $\delta g/\delta x = (\delta g/\delta y)(\delta y/\delta x)$, doing so would be computationally expensive due to the large size of $\delta y/\delta x$ (which can be thought of as a matrix of derivatives for each image pixel with respect to each scene parameter). Instead, the back-propagation method uses the (partial) derivative of the objective function $\delta g/\delta y$ as a source of "adjoint radiance" which is then traced from the virtual camera into the scene, scattered according to Eq. (3) and "collected" at the scene parameters.

In this work, we rely on the ADAM optimizer [15] to solve the parameter estimation problem. This method is a general-purpose method that applies gradient descent with momentum and an adaptive strategy to estimate the learning rate per parameter. As such it is well suited to non-linear optimization problems that may contain spurious local minima. Using a momentum strategy can prevent getting stuck in these local minima and increase the chances of finding a good solution.

3.2 Problem statement

In summary, our goal is to estimate optical material parameters $x = \{c, \alpha, \sigma_s, \eta, \sigma_t\}$ in a given 3D scene, such that the rendered result best approximates one (or more) reference image(s). Therefore, we require

- a (set of) reference image(s) and
- a Mitsuba scene file, including the initial guess for the material parameters, and virtual cameras corresponding to the reference images as input to our system.

In order to solve this problem, we apply differentiable rendering as summarized above, and provided by Mitsuba 3, combined with a gradient-based optimization procedure, which we describe in the following section.

4 Method

In this section, we will discuss our approach to solving the defined problem in Section 3.2. This section is divided into two parts: *experimental* and *computational*. In the experimental part, we are going to introduce our workflow to meet the requirements defined in Section 3.2. In the computational part, we will show our procedure to reach the goals that we defined in Section 3.2.

4.1 Scene and image acquisition

In this section, we discuss scene and image file acquisition. For validation tests where the Bunny [35] model is in use, we utilized the readily available Mitsuba scene from [25] and modified the parameters we introduced in Section 3.1. However, for synthetic data, users may also construct a scene in a 3D computer graphics software tool such as Blender [5]. For example, for the Dragon [6] model test case in Section 5.1, we constructed a scene in Blender and exported it using Mitsuba's Blender Add-on [2]. For validation tests, we then rendered (using Mitsuba) acquired scenes to obtain corresponding reference images.

The real-world case is more complicated. In the imaging phase, it is important to have a controlled environment where all the scene parameters that could impact the resulting image, especially the scene geometry and light position(s), are accurately known. We propose two approaches, *first*, users may reconstruct the imaged environment in Blender. This task can get seriously complicated and requires relatively adequate skills in modeling. *Fortunately*, for the alginate [20] material test cases we acquired from Prof. Stavric and colleagues at TU Graz, we were provided by Prof. Ferschin (TU Wien) and his former masters' student Cheng Shi a "material scanner"—a small light-proof container with fixtures for camera and light placement—and a corresponding 3D model that we used in Blender. Our *second* approach utilizes a photogrammetry tool Metashape [1]. We not only acquired the geometry of the imaged bird-statue (first and third row in Fig. 7) but also the camera positions from photographs. Using Metashape [1], (1) we loaded and aligned images. (2) Used the *Build Mesh* functionality. (3) Exported mesh and camera locations with the X3D format. (4) Imported the X3D file into Blender. Once we had a scene in Blender, we utilized the Mitsuba Blender Add-on [2] to export and use it in our tool. We found our second approach useful specifically for *opaque* materials. Another technique we employed involved *naive* vertex position optimization that is supported by Mitsuba 3 and our tool. As shown in the second and fourth row of Fig. 7, from a scaled-sphere object the optimization recovers a *rough* bird statue. The geometry reconstruction step can also be achieved with different NeRF procedures [23, 26, 14, 21]. Geometry reconstruction on its own is a fascinating and challenging research topic, and we will omit the details in this work.

4.2 Optimization

We are now ready to introduce the computational part of our approach, combining the differentiable renderer Mitsuba 3 with an ADAM optimizer to estimate the material parameters.

Using our software tool (Fig. 2), we first load a virtual 3D scene file, which includes the initial material parameters x_0. We also load the (set of) reference image(s), y_{ref}, which define the objective function (selecting either the L2 norm or the dual buffer method as introduced earlier). For each reference image (and corresponding virtual camera), we initialize an ADAM optimizer, and select the following optimization hyper-parameters (default values in braces). The

maximal number of iterations for the optimization (100); the number of *samples per pixel (spp = 4)* for each rendered image. Note that scene and (reference) image resolution is overridden by our tool according to the aspect ratio of the loaded image and restricted into $(256 \times AspectRatio, 256)$. This restriction originates mainly from memory limitations. And finally, the *loss tolerance* ε, where optimization stops if the loss $g < \varepsilon$ (0.001), the *learning rate*, which adjusts the step size at each iteration (0.03), as well as *minimum and/or maximum clamp values*, which define box constraints for specific parameters. For example, an RGB color value must be in the interval $[0,1]$; by default, parameters are constrained, i.e. in $[0,1]$.

Initializing $x_i = x_0$, we then run the following optimization loop for each camera pose (i.e. reference image): (1) Perform a differentiable rendering step with respect to x_i resulting in an image y_i. (2) Evaluate the objective function $g(y_i)$. (3) Back-propagate $\delta g/\delta y$ using Mitsuba 3, to obtain $\delta g/\delta x_i$. (4) Take an ADAM optimization step find updated parameters $\tilde{x}_{i+1}$. (5) Ensure legal values for x_{i+1} by clamping $\tilde{x}_{i+1}$ using box constraints. (6) Update the scene with x_{i+1}. Repeat until either the loss tolerance, or the maximal iterations are reached.

Additionally, our software tool also provides convenience functions. To streamline the above-mentioned process, we propose a mini-tool with a GUI [1]. In its simplest form, users only need to load a *Mitsuba 3* [11] scene and a reference image and pick the proper material parameters to execute the optimization. After loading a scene, our tool picks the qualified integrators, scene resolution, and provides default hyperparameters, and presents supported differentiable scene parameters for optimization. At the end of the optimization, our tool not only provides visualizations similar to the Figures in §5 but also allows to export of the obtained optimized parameter results.

5 Results

In this section, we examine results produced using our software tool and optimization approach. In the first part of this section, we evaluate results from synthetic data, where ground-truth solutions are available for comparison. In the second part, we show results for real-world data.

5.1 Validation tests

In this section, we examine material reconstruction from synthetic data. These tests start from a virtual scene, with given ground-truth parameters. The optimization must then recover the correct material parameters from a dark and opaque initial guess. First, in Fig. 3, we show successful results using the method described in Section 4.2. Table 1

[1]We expect the use of this tool for academic purposes. Consequently, the design of the GUI is constructed with simplicity. Thus the design and human-computer interaction related intricacies are beyond the scope of this paper.

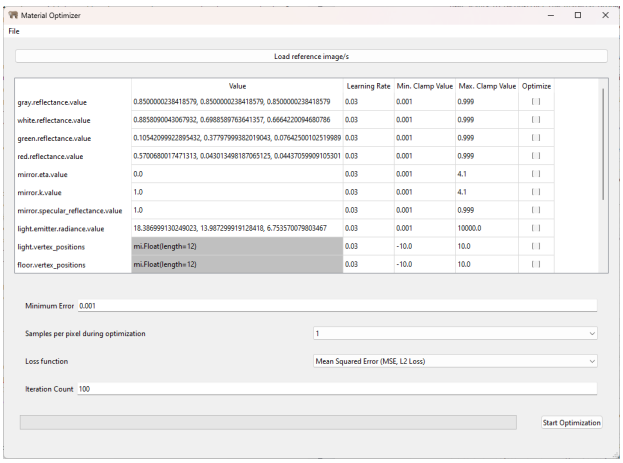

Figure 2: The main interface of our tool. The user first loads a scene (File menu), then proceeds from top to bottom: loading reference images, selecting optimization parameters, and setting constraint values, choosing hyperparameters, and finally executes the optmization process.

also shows the corresponding initial and optimized parameter values and optimization hyperparameters.

In all cases, we observe that the dual buffer method [7] is extremely useful in reconstructing the object's material properties, even when provided with *less* restrictive constraints. Conversely, using the standard (single-image) L2 error, we *often* require more restrictive contraints to prevent some parameters from diverging. Consequently, in this paper, we *primarily* focus on translucent material reconstruction using the dual buffer method.

Furthermore, we observe that noise inherent to MC sampling affects the optimization procedure, and a sufficient number of samples per pixel (*spp*) is required to obtain good convergence. We also use box constraints on certain parameters, i.e. clamping to minimum or maximum values to prevent some parameters from moving to physically implausible values[2]. Especially the index of refraction (η) often suffers from these issues. We believe this is due to total internal reflection introducing discontinuous jumps in light propagation paths. Another useful strategy to resolve these issues is to split the optimization process into two parts: first we optimize a group of parameters that work well together (for instance excluding η). We then continue from the optimized values from the first part and include all parameters. Results using this procedure are shown in the last two rows of Fig. 3.

Having established that our method is capable of recovering ground-truth material parameters, we now show additional comparisons and discuss potential pitfalls during translucent material reconstruction in a short ablation study, Fig. 4 and Table 2. We compare the dual buffer method (2×8 spp) to the single-image L2 error metric (1×16 spp) on the Stanford bunny using the Principled BSDF (first row

[2]The corresponding minimum and maximum clamp values can always be found in the last column (*opt. params*) of resulting Tables.

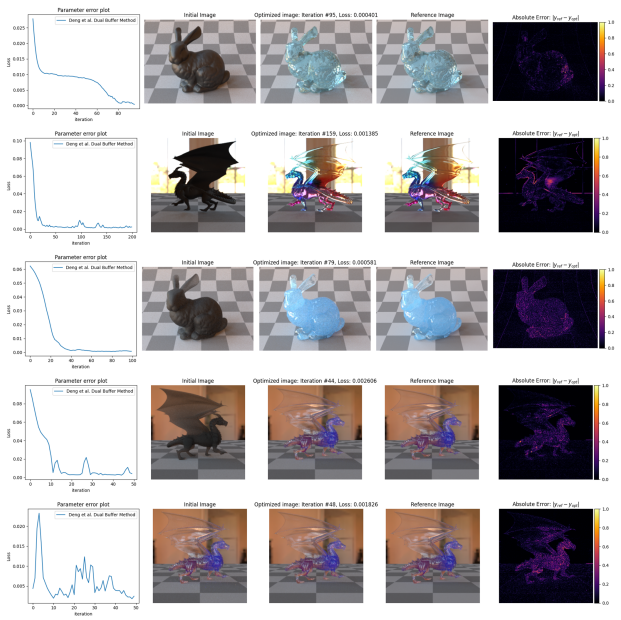

Figure 3: We successfully optimize material parameters for the Principled (first and second row) and Rough dielectric BSDF (third to fifth row). Columns: (1) convergence plot, (2) initial state, (3) optimized result, (4) reference image, and (5) the absolute error between the reference and optimized image. For corresponding parameter values please refer to Table 1.

in Fig. 3 and 4 respectively). While both approaches result in acceptable visual and numeric results, the single-image version, however, required additional constraints. Most notably, we clamp η to a maximum of 1.55 to enable convergence. This restriction was *not* necessary when using the dual buffer method. Note that the minimum loss in the L2 error was 0.005608 (in iteration 121), whereas the dual buffer version obtained 0.0000401 (in the 95th iteration).

On our second test case, the Dragon using a surface BSDF, we compare the optimization behaviour for different numbers of reference images (4 vs. 1) and samples per pixel (4×16 spp vs. 1×8 spp), as well as relaxing the minimum clamp value for the specular transmission (σ_s) parameter. As shown in Fig. 4 (second row), the lower sample count

Table 1: Results corresponding to Figure 3; DBM = dual buffer method, ic = iteration count. Unless specified otherwise, default values are used. Please note that the optimization results that are shown in the last row are achieved by separating the optimization procedure into two parts.

Case	parameters	initial	optimized	reference	hyperparam.	opt. param.
Bunny (Principled)	c	0.01, 0.01, 0.01	0.4192, 0.8531, 0.9990	0.412, 0.824, 0.999	*spp*=8, DBM	Default
	α	0.5	0.001	0.01		
	σ_t	0.02	0.9184	0.9		
	η	1.54	1.4737	1.49		
Dragon (Principled)	c	bitmap	bitmap	bitmap	*spp*=16, ic = 200, DBM	Default
	α	0.7	0.0022	0.001		
	σ_t	0.1	0.9749	1		min=0.11
	η	1.64	1.5135	1.49		Default
Bunny (Rough dielectric)	c	0.01, 0.01, 0.01	0.4810, 0.8495, 0.9990	0.412, 0.824, 0.999	*spp*=16, min. err.=0.0001 DBM	Default
	α	0.5	0.001	0.01		
	σ_t	0.98	0.4841	0.4		
	η	1.544	1.5169	1.49		max=1.55
Dragon (Rough dielectric)	c	varying volume	varying volume	varying volume	*spp*=4, DBM	Default
	α	0.7	0.0098	0.01		
	σ_t	0.98	0.349	0.4		
	η	1.544	1.4869	1.49		

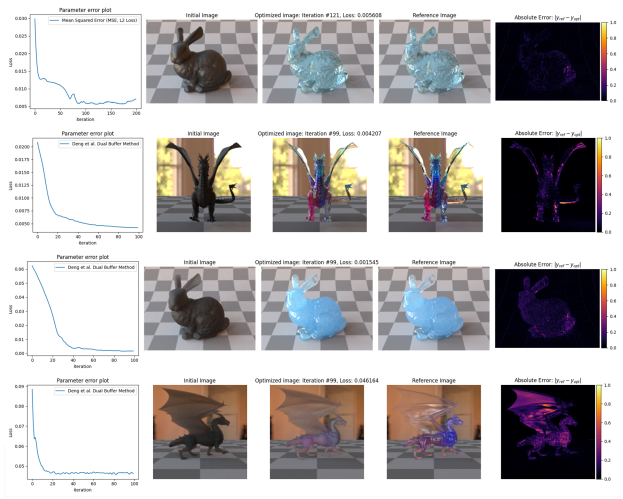

Figure 4: Further results from our comparisons with different optimization strategies and hyper-parameters for the Principled (first and second row) and Rough dielectric (third, fourth row) BSDF Please also refer to Table 2 for details.

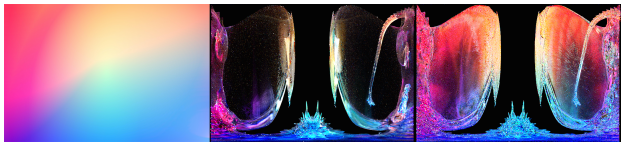

Figure 5: Texture used in the Dragon (Principled) test case. Left: reference bitmap; middle: optimized result using a single camera; right: optimized result from four cameras.

results in visually slightly worse appearance after parameter optimization. However, as shown in Table 2 (second row), the numerical results are not satisfactory. Additionally, as shown in Fig. 5, the single-image optimization encodes a lot of appearance details into the texture image. Using multiple reference images, on the other hand, allows for improved texture recovery (note that the unused areas in the texture will always remain unchanged during optimization).

Using the volumetric material model, we again test the influence of noise on the optimization procedure. In the third row of Figure 4, we observe visually acceptable re-

Table 2: Results corresponding to Figure 4; DBM = dual buffer method, ic = iteration count. Unless specified otherwise, default values are used.

Case	parameters	initial	optimized	reference	hyperparam.	opt. param.
Bunny (Principled)	c	0.01, 0.01, 0.01	0.4119, 0.8450, 0.9990	0.412, 0.824, 0.999	*spp*=16, ic=200	Default
	α	0.5	0.001	0.01		
	σ_t	0.02	0.9002	0.9		
	η	1.54	1.4973	1.49		max=1.55
Dragon (Principled)	c	bitmap	bitmap	bitmap	*spp*=8, DBM	Default
	α	0.7	0.1266	0.001		
	σ_t	0.1	0.001	1		
	η	1.64	1.8079	1.49		
Bunny (Rough dielectric)	c	0.01, 0.01, 0.01	0.5011, 0.8844, 0.9990	0.412, 0.824, 0.999	*spp*=8, DBM	Default
	α	0.5	0.0089	0.01		
	σ_t	0.98	0.5991	0.4		
	η	1.544	1.6042	1.49		
Dragon (Rough dielectric)	c	varying volume	varying volume	varying volume	*spp*=16, DBM	Default
	α	0.7	0.3065	0.01		
	σ_t	0.98	0.2705	0.4		
	η	1.544	1.5499	1.49		max=1.55

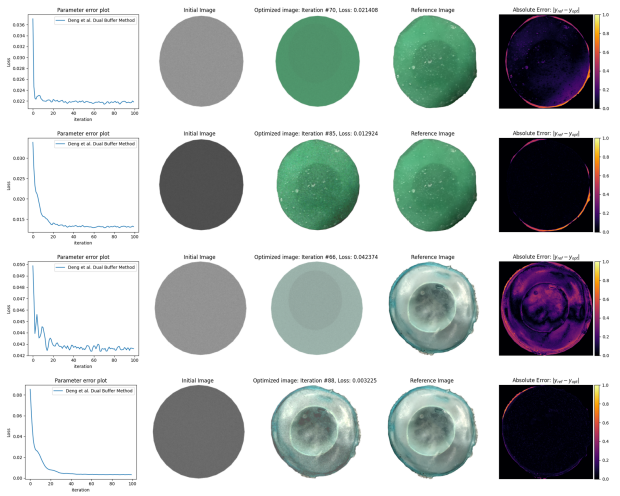

Figure 6: Results from the green (first and second row) and blue (third and fourth row) alginate [20] materials. Note that in both cases Principled [4, 9] BSDF is in use. For corresponding numerical results please refer to Table 3.

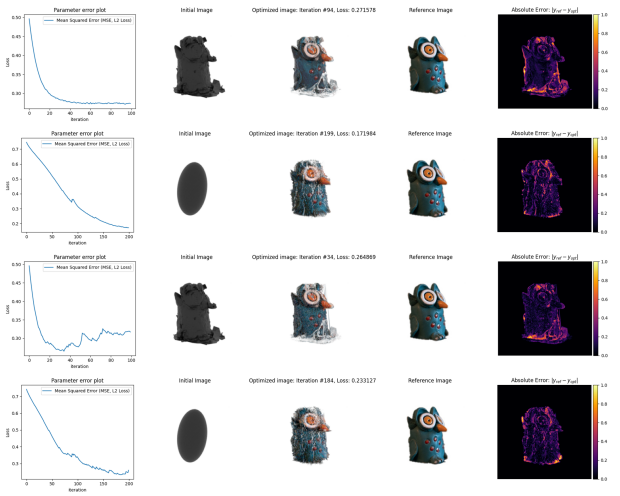

Figure 7: Results from a real-world bird statue. First row: Material reconstruction using the proposed procedure from Section 4. Second row: Geometry reconstruction and base color optimization. Third row: Material reconstruction as in the first row, including vertex position optimization. Fourth Row: Geometry and material reconstruction. For corresponding numerical results please refer to Table 4.

sults, whereas numerical results degrade when reducing the samples from 16 spp to 8. In this experiment, the index of refraction η initially increases (moving away from the expected reference value), and subsequently, the extinction coefficient (σ_t) remains not recovered accurately.

Finally, applying the volumetric material to the more complex Dragon scene in the fourth row of Fig. 4, we attempt to optimize all parameters of interest simultaneously (as opposed to the successful case presented earlier, where we optimize in two stages). Interestingly the material's roughness (α) fails to reduce sufficiently from an initially high value, resulting in a visually noticeable mismatch between the optimization result an the reference. Note that both in this example and in our successful result, we use multiple reference images. Consequently, in some cases, we suggest separating the optimization procedure into two parts, as discussed earlier, to acquire acceptable results, as shown in the last two rows of Figure 3.

5.2 Real-World applications

In this section, we perform material reconstruction from real-world data. In particular, we are interested in acquiring optical material properties for two specimens made of a novel alginate material [20]. As these alginate specimens are relatively thin, we use the *Principled* BSDF [4, 9] material model, which has proved easier to optimize in the test cases discussed previously. Figure 6 shows results for both (green and blue) translucent alginate specimens. We again employ the strategy to split the optimization into two parts for both results. We first only allow one constant RGB colour, and in the second stage extend the optimization to a bitmap texture image. We obtain visually acceptable results, even though the virtual geometry is not perfectly matched to the real world images, resulting in errors along the outer edge of the specimens, seen in the absolute error images in Fig. 6. Table 3 summarizes the numerical results corresponding to Fig. 6.

For both cases, we apply specific minimum and maximum clamp values for the α and σ_s parameters. Without these additional constraints, we obtained visually identical results, albeit with physically unreasonable parameter values. Therefore we include these constraints in order to

Table 3: Results for the real-world alginate specimens.

Case	parameters	initial	optimized	hyperparam.	opt. param.
Alginate (Green)	c	0.1 0.1 0.1	bitmap	*spp*=8, DBM	Default
	α	0.7	0.6999		max=0.7
	σ_s	0.1	0.4503		min=0.4
	η	1.54	1.2567		Default
Alginate (Blue)	c	0.1 0.1 0.1	bitmap	*spp*=8, DBM	Default
	α	0.7	0.3828		max=0.6
	σ_s	0.1	0.5132		min=0.5
	η	1.54	1.3406		Default

Table 4: Results corresponding to Fig. 7. Note that min. and max. clamp values used in this example refer to the min. and max. value of the reconstructed object's bounding box from the 1. Row in Fig. 7; lr = learning rate.

Case	parameters	initial	optimized	hyperparam.	opt. param.
1. Row	c	bitmap	bitmap	Default	Default
	specular	0.5	0.4279		
	α	0.5	0.0732		
2. Row	c	bitmap	bitmap	ic=200	Default
	vertex_positions	scaled sphere	shape		lr=0.0008, min/max=(-0.875, -0.933, 9.320)/(0.586, 0.723, 10.944)
3. Row	c	bitmap	bitmap	Default	Default
	specular	0.5	0.999		
	α	0.5	0.1193		
	vertex_positions	reconstructed	shape		lr=0.0008, min/max=(-0.875, -0.933, 9.320)/(0.586, 0.723, 10.944)
4. Row	c	bitmap	bitmap	ic=200	Default
	specular	0.5	0.999		
	α	0.5	0.001		
	vertex_positions	scaled sphere	shape		lr=0.0008, min/max=(-0.875, -0.933, 9.320)/(0.586, 0.723, 10.944)

obtain a plausible *translucent* material result.

Finally, in Fig. 7 and Table 4, we also show the results of a more complex opaque object. In Fig. 7, the first row showcases results utilizing the geometry obtained through Metashape [1], where only the *c*, *specular*, and α parameters of the object were optimized. The second row features results using a (scaled) sphere object, with optimization of *c* and *vertex_positions* parameters. The third row utilizes the reconstructed geometry as in the first row, but with the addition of *vertex_positions* optimization. The fourth row uses a (scaled) sphere object as in the second row but includes optimization of additional parameters that may cause discontinuities, such as *specular*, and α parameters.

In conclusion, we acknowledge the complexity of real-world material and geometry reconstruction and emphasize the importance of a comprehensive data acquisition process that accounts for various factors, including lighting, image acquisition, object and light positions, and geometry reconstruction. However, as shown in Fig. 1, with accurate measurements, multiple reference images, and thorough experimentation, material and geometry reconstruction can be successfully achieved.

6 Conclusion

In this paper, we describe a material reconstruction pipeline built upon the Mitsuba 3 differentiable renderer. Our software tool is capable of reconstructing material properties from a scene description file and multiple images. The main focus of this project is on translucent material reconstruction, which we validated using synthetic data and demonstrated on real-world specimens. We test our appraoch on both a surface-only *Principled* BSDF, as well as a volumetric *Rough dielectric* BSDF with homogeneous participating media.

One difficulty we observed was the complexity of the scene and image file acquisition. The nature of real-world data acquisition can be extremely complex since the measurement process requires a controlled environment and either manual, or automated, geometry reconstruction prior to material identification. Regarding translucent materials, we observed that the noise introduced by MC sampling affected the parameter estimation procedure. Employing the dual buffer method noticably improved our results. Our analysis revealed that certain parameters that introduce discontinuities in the scattering behaviour, such as the index of refraction, caused difficulties in achieving accurate convergence. In some cases, we found that two-stage optimization was necessary. Especially when optimizing a (surface or volumetric) texture for the material's albedo, these highly localized parameters might have a stronger influence than global material parameters, which may result in inaccurate reconstruction. Furthermore, our findings indicated that in certain scenarios, additional constraints, such as imposing a maximum clamp value on some parameters, were necessary. We believe that this limitation stems from slight imprecisions in the image acquisition and modelling phase.

In the future, we aim to improve our geometric modelling and image acquisition procedures, which will allow for more accurate representation of physics-based reality with well-established material models. Based on the evidence obtained from our results, we conclude that our approach will be beneficial for different translucent material reconstruction tasks in various applications.

References

[1] AgiSoft. Agisoft metashape standard, 2022. https://www.agisoft.com/.

[2] Baptiste Nicolet ¡baptiste.nicolet@epfl.ch¿. Mitsuba blender add-on. `https://github.com/mitsuba-renderer/mitsuba-blender`, 2022.

[3] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192198, jul 1977.

[4] Brent Burley. Physically-based shading at disney. 2012.

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022.

[6] Delatronic. Dragon. `https://blendswap.com/blend/15891`, August 21, 2015. [Online; accessed January 30, 2023].

[7] Xi Deng, Fujun Luan, Bruce Walter, Kavita Bala, and Steve Marschner. Reconstructing translucent objects using differentiable rendering. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.

[8] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6), nov 2013.

[9] Stephen Hill, Stephen McAuley, Brent Burley, Danny Chan, Luca Fascione, Michał Iwanicki, Naty Hoffman, Wenzel Jakob, David Neubelt, Angelo Pesce, and Matt Pettineo. Physically based shading in theory and practice. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[11] Wenzel Jakob, Sbastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and

Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org.

[12] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[13] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143150, aug 1986.

[14] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[16] Chris Lattner and Vikram Adve. LLVM: A compilation framework for lifelong program analysis and transformation. pages 75–88, San Jose, CA, USA, Mar 2004.

[17] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Trans. Graph.*, 25(3):735745, jul 2006.

[18] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph.*, 37(6), dec 2018.

[19] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), December 2019.

[20] Ivan Marjanovi, Elizabeta amec, Hana Vaatko, and Milena Stavri. Alginate in architecture : An experimental approach to the new sustainable building material. In and , editors, *Art and Science Applied: Experience and Vision*, volume 2 of *SmartArt*, chapter 21, pages 394–406. , , , 2022. 21.

[21] Evonne Ng* Matthew Tancik*, Ethan Weber*. Nerfstudio: A framework for neural radiance field development, 2022.

[22] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759769, jul 2003.

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99106, dec 2021.

[24] Mitsuba3. Mitsuba 3: Plugin reference. `https://mitsuba.readthedocs.io/en/stable/src/plugin_reference.html`, 2023. [Online; accessed 11-January-2023].

[25] Mitsuba3. Mitsuba 3: Read the docs. `https://mitsuba.readthedocs.io/en/stable/`, 2023. [Online; accessed 14-February-2023].

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), jul 2022.

[27] Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. Unbiased inverse volume rendering with differential trackers. *ACM Trans. Graph.*, 41(4), jul 2022.

[28] Merlin Nimier-David, Sbastien Speierer, Benot Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4), July 2020.

[29] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), December 2019.

[30] NVIDIA, Pter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[31] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (3rd ed.)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, October 2016.

[32] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311317, jun 1975.

[33] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., USA, 3rd edition, 2009.

[34] Tanaboon Tongbuasirilai, Jonas Unger, Joel Kronander, and Murat Kurt. Compact and intuitive data-driven brdf models. *The Visual Computer*, 36(4):855872, 2019.

[35] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 311318, New York, NY, USA, 1994. Association for Computing Machinery.

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

[37] Delio Vicini, Sbastien Speierer, and Wenzel Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 40(4):108:1–108:14, August 2021.

[38] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 195206, Goslar, DEU, 2007. Eurographics Association.

[39] Jingjie Yang and Shuangjiu Xiao. An inverse rendering approach for heterogeneous translucent materials. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, VRCAI '16, page 7988, New York, NY, USA, 2016. Association for Computing Machinery.

[40] Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. Monte carlo estimators for differential light transport. *ACM Trans. Graph.*, 40(4), jul 2021.

[41] 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.

[42] Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. Physics-based differentiable rendering: From theory to implementation. In *ACM SIGGRAPH 2020 Courses*, SIGGRAPH '20, New York, NY, USA, 2020. Association for Computing Machinery.