Translucent Material Parameter Estimation

Saip-Can Hasbay[(1)]
Supervised by: David Hahn[(2)]

(1) University of Vienna, (2) TU Wien

Abstract

In this paper, we present a workflow for optical material parameter estimation based on real-world images, numerical optimization, and differentiable rendering (using *Mitsuba 3*). We primarily focus on *translucent* materials and demonstrate our approach both for synthetic and real-world data. 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. Furthermore, we also propose an experimental workflow to acquire the necessary reference images and potentially the 3D scene geometry. 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

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.

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 [26, 2, 27]. The driving force behind many rendering methods was artistic expression rather than physical accuracy. However, with more powerful hardware supporting modern ray tracing algorithms [23, 14], photo-realistic and physics-based rendering has become not only feasible but de-facto standard in many fields, both within and outside of computer graphics.

Given a virtual scene description, the main goal of physics-based rendering is to create images matching the way our eyes perceive the world. 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.

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. Lastly, we present and analyze our results from both synthetic and real-world data. In summary, our contributions are:

- A software tool[1] 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.
- 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 [12], formulating the now well-known rendering equation. Veach [32] introduced the path-integral form, which provides the theoretical foundation of modern Monte Carlo (MC) ray tracing methods [25]. In order to arrive at a photorealistic rendering, an accurate description of how various materials interact with light must be formulated. Different analytical material models have been introduced to account for the light-scattering effects from surfaces [34, 9, 11]. Similarly, models for participating media describe volumetric scattering effects as light passes through the material [25, 11]. Physics-based rendering itself (which we also refer to as *forward* rendering in this context) is concerned with computing accurate images using 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 [19, 15, 28], 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 physics-based 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 [22, 33, 21, 16, 37] 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 [22, 33]. 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 [16, 36, 17, 38].

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. [36] analyzed numerous potential approaches and mathematically classified various estimators for differential light transport. Prior to the work by Vicini et al. [33], 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 by Gkioulekas et al. [8] is one of the initial inverse rendering research regarding translucent material reconstruction. They introduced an optimization framework to measure the bulk scattering properties of homogeneous materials. Their primary focus was describing homogeneous materials 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. [35] proposed another inverse rendering approach for heterogeneous translucent materials from a single input photograph. Their method can obtain the material distribution and estimate material parameters from the provided reference image. Yang et al. introduced an iteration process for optimizing the scattering and absorption 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). 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).

[1] Access at https://github.com/sapo17/BachelorThesis.

3 Parameter estimation

In this section, we briefly recap the underlying theory of the rendering process and the core principle of differentiable ray tracing. Next, we present the material parameters of interest, and finally state the problem we address.

3.1 Background and theory

Let us start with the well-known rendering equation as formulated by Kajiya [12], also referred to as the light transport equation (LTE). As we are primarily interested in translucent materials, we consider the angular form of the LTE, see also [25, 12]:

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

Here $L(p,\omega_o)$ describes the radiance (power per area, solid angle, and color channel) leaving a point p in direction ω_o 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 [25].

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).

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

$$\min g(y(x)), \text{ subject to } h(x) \leq 0, \quad (2)$$

where g defines the optimization objective (or loss) function on the rendered image y given scene parameters x, and h defines additional constraints (e.g., 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 the specified objective [38]. In the simplest form, the objective takes the L2 norm of the difference between the rendered and the reference image, 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. 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 back-propagation method [22, 33]. Formally, we take the derivative of Eq. (1) with respect to the scene parameters x:

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

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 [13] 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 Microfacet models and Disney BSDF

Modeling surface reflection and transmission often originates from the idea that rough surfaces can be represented

as a collection of small microfacets. The scattering of light from a group of microfacets is statistically modelled by microfacet-based BRDFs [25]. Walter et al. [34] introduced a microfacet distribution function, called GGX, which has been shown to be equivalent to the microfacet distribution introduced by Trowbridge and Reitz [30]; see also [24]. Microfacet models not only require the distribution of facets, but also need to account for some facets being obscured by others (also known as *masking*) and some will be *shadowed*. Smith's masking-shadowing function [25] addresses these effects. More recently, Walter et al. [34] reviewed microfacet theory and extended it to simulate transmission through rough surfaces. *Mitsuba 3* [10] implements this variant in its Rough-dielectric BSDF model, which we make use of in this project. Furthermore, *Mitsuba 3* allows two types of participating media in the scene: homogeneous and heterogeneous, which can be used to simulate translucent materials such as milk or skin, but also clouds and fog [10]. Therefore we combine a Rough-dielectric BSDF and a homogeneous interior medium to simulate and optimize volumetric scattering effects in translucent materials.

Burley et al. [3] proposed a general-purpose BRDF (also known as Disney Principled BRDF). They compared their new model with measured materials in terms of the microfacet models [29, 5, 34]. Their BRDF blends metallic and dielectric BRDF models and includes a microfacet reflection with anisotropic roughness and an optional secondary clearcoat reflection. The Disney BSDF [9] extends the dielectric BRDF with integrated subsurface scattering and blends in a specular BSDF based on a new specular transmission parameter, arriving at a unified BSDF model including refraction and subsurface scattering effects. For the specular BSDF, they follow the GGX model and extend the microfacet reflection to refraction. *Mitsuba 3* [10] implements the model proposed by Burley et al. [9] in its Principled BSDF material, which we also use in this project.

3.3 Material parameters

Following the summary of material models, we now briefly list the parameters of these models we use in our optimization pipeline. For surface-only materials, i.e. the *Disney Principled BSDF*, we have:

- 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 reflection effects such as clear-coat materials, or metallic reflections here. For volumetric rendering, i.e. *Rough Dielectric BSDF* with homogenous medium [34, 11], we additionally consider

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

3.4 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.

To solve this problem, we combine differentiable rendering with a gradient-based optimization procedure. To use our approach, however, we first need to meet the above-mentioned requirements, which we discuss in the first part of the next section.

4 Method

In this section, we first introduce the experimental and then the computational part of our approach.

4.1 Scene and image acquisition

Here, we focus on 3D scene construction and reference image acquisition. For validation tests where the Bunny [31] model is in use, we utilized the readily available Mitsuba scene from their documentation [11] and modified the parameters we introduced in §3.1. However, for synthetic data, users may also construct a scene in a 3D modelling tool such as Blender [4]. For example, for the Dragon [6] model test case in §5.1, we constructed a scene in Blender and exported it using Mitsuba's Blender Add-on [20]. For validation tests, we then rendered (using Mitsuba) the acquired scenes to get the reference images.

The real-world case is more complicated. During imaging, a controlled environment is crucial to accurately determine all scene parameters that could affect the resulting image, particularly scene geometry and light position(s).

We propose multiple approaches, *first*, users may reconstruct the imaged environment manually. This task can get seriously complicated and requires relatively adequate skills in modelling. For the alginate [18] material test cases, we only partially had this difficulty. We were provided with a "material scanner"—a small light-proof container with fixtures for camera and light placement—where we photographed the alginate samples. Fortunately, we were also provided with the corresponding 3D model that we used in Blender. However, we still had to approximate the model of the sample alginate materials, camera positions, and the radiance values of the lights. Please note that we processed images from the real-world to remove the background.

Our *second* approach utilizes a photogrammetry tool Metashape [1]. Using Metashape, we not only acquired the geometry of the imaged bird statue (third row in Fig. 5) but also the camera positions from photographs. Using Metashape, (1) we first loaded images and used the *Align*

Images functionality, (2) then used the *Build Mesh* functionality, (3) next, exported mesh and camera locations with the X3D format, (4) and lastly, imported the X3D file into Blender. Once we had a scene in Blender, we utilized the Mitsuba Blender Add-on [20] 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 fourth row of Fig. 5, from a scaled-sphere object the optimization recovers a *rough* bird statue. 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, 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 will be used in the objective function (selecting either the L2 norm or the dual buffer method as introduced earlier). Next, we select the material parameters of interest (x_0), which get assigned to a newly initialized ADAM optimizer by our tool. Optionally, we also 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; 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); and *minimum and/or maximum clamp values*, which define box constraints for specific parameters. For example, an RGB color value must be in the interval $[0,1]$ per channel; by default, most parameters[2] are constrained to $[0,1]$. Note that virtual camera and reference image resolutions are overridden by our tool according to the aspect ratio of the loaded image and restricted to $(256 \times AspectRatio, 256)$. This restriction originates mainly from memory limitations.

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 to 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 [3]. To execute the optimization, users can simply load a Mitsuba 3 scene and a (set of) reference image(s), and select the appropriate material parameters. Our tool automatically selects integrators and scene resolution, sets default hyperparameters, and presents differentiable scene parameters for optimization upon scene loading. At the end of the optimization, our tool provides the necessary visualizations and ability to export the obtained results.

5 Results

In this section, we examine results produced using our software tool and optimization approach. In the first part, 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. Our tool uses the NVIDIA CUDA variant of Mitsuba v3.2.1, and all timings reported in the following have been measured on NVIDIA GeForce RTX 2060 graphics card.

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 Figs. 2, 3, we show successful results using the method described in §4.2. Table 1 also shows the corresponding initial and optimized parameter values and optimization hyperparameters. Please note that we use a single reference image for the Bunny and four images for the Dragon test cases.

In all cases, we observe that the dual buffer method [7] is extremely useful in reconstructing the object's material properties, allowing for *less* restrictive constraints.

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. 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. Result using this procedure is shown in the last row of Fig. 2.

Having established that our method is capable of recovering ground-truth material parameters, we now show

[2] The interested reader may find the default values in the *constants.py* file of our repository.

[3] We expect the use of this tool for academic purposes. The design and HCI related topics are beyond the scope of this paper.

Case	Params.	Init.	Opt. (succ.)	Ref.	Opt. (unsuc.)	Hyper. (succ.)	Hyper. (unsuc.)	Loss / Comp. time (succ.)	Loss / Comp. time (unsuc.)
Bunny (P)	c	0.01, 0.01, 0.01	0.415, 0.853, 0.999	0.412, 0.824, 0.999	0.397, 0.833, 0.999	*spp*=8, DBM, ε=0.0001	*spp*=16, MSE	0.0002 / 48.9s	0.0098 / 35.6s
	α	0.5	0.001	0.01	0.001				
	σ_s	0.02	0.904	0.9	0.912				
	η	1.54	1.495	1.49	1.747				
Dragon (P)	c	bitmap	bitmap	bitmap	bitmap	*spp*=16, DBM, $h(\sigma_s)_{min}$=0.11	*spp*=8, DBM	0.0015 / 231.3s	0.0528 / 180.9s
	α	0.7	0.001	0.001	0.135				
	σ_s	0.1	0.973	1	0.001				
	η	1.64	1.507	1.49	1.815				
Bunny (R)	c	0.01, 0.01, 0.01	0.469, 0.844, 0.999	0.412, 0.824, 0.999	0.599, 0.924, 0.999	*spp*=16, ε=0.0001, DBM, $h(\eta)_{max}$=1.55	*spp*=8, DBM	0.0005 / 237.3s	0.0025 / 154.6s
	α	0.5	0.014	0.01	0.001				
	σ_t	0.98	0.502	0.4	0.585				
	η	1.544	1.503	1.49	1.919				
Dragon (R)	c	volume	volume	volume	volume	*spp*=4, DBM, 2-stage	*spp*=16, DBM	0.0016 / 241.2s	0.0456 / 498.3
	α	0.7	0.011	0.01	0.307				
	σ_t	0.98	0.357	0.4	0.26				
	η	1.544	1.484	1.49	1.549				

Table 1: Results from Figs. 2, 4; DBM=Dual Buffer Method, MSE=Mean Squared Err., P=Principled BSDF, R=Rough dielectric BSDF, succ.=successful, unsuc.=unsuccessful.

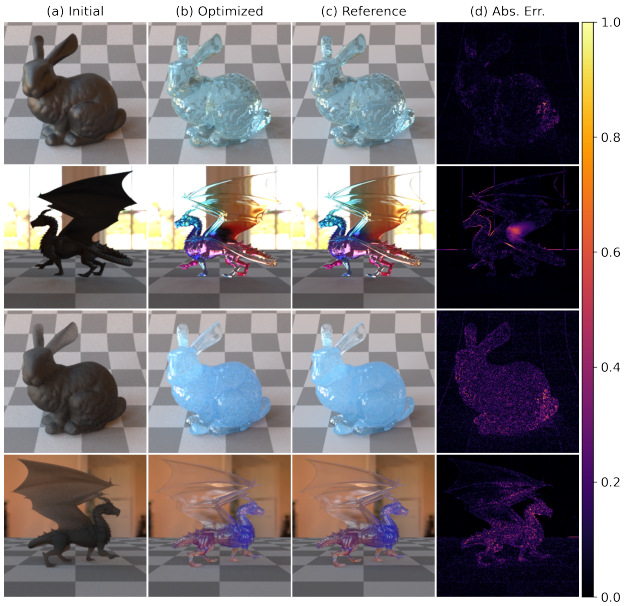

Figure 2: Our successful attempts for the Principled (first and second rows) and Rough dielectric (third and fourth rows) BSDF. For corresponding parameter values please refer to Table 1.

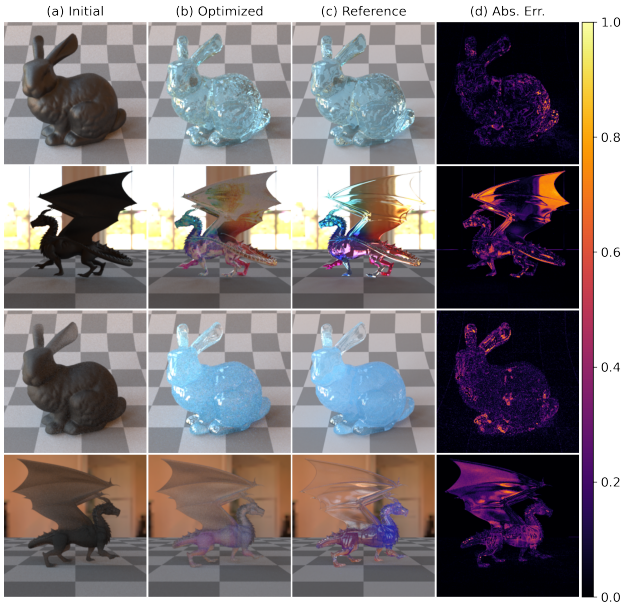

Figure 4: Our unsuccessful attempts for the Principled (first and second rows) and Rough dielectric (third and fourth rows) BSDF. For corresponding parameter values please refer to Table 1.

additional comparisons and discuss potential pitfalls during translucent material reconstruction in a short ablation study, Figs. 4, 3 and Table 1. 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 in Fig. 2 and 4 respectively). Although the visual result might look acceptable, the numerical results (Table 1)—specifically for the η parameter—are not satisfactory. Note also the instability of the convergence compared to our successful attempt (Fig. 3).

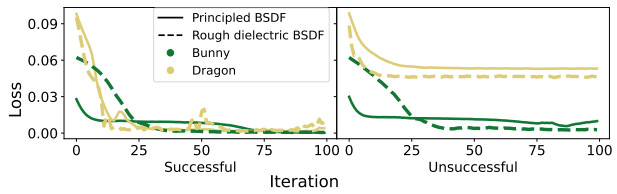

Figure 3: Convergence plots (synthetic-data) from (un)successful attempts. See also Figs. 2, 4.

On our second test case, the Dragon using a surface BSDF, we compare the optimization behaviour for different samples per pixel (16 vs. 8) value, and relaxing the minimum clamp value for the specular transmission (σ_s) parameter. As shown in Fig. 4 (second row), the lower sample count results in a visually worse appearance. As shown in Table 1 (second row), the numerical results are also not satisfactory.

Using the volumetric material model, we again test the influence of noise on the optimization procedure. In the

Case	Params.	Init.	Opt.	Hyper.	Loss / Comp. time
Alginate (G)	c	0.1 0.1 0.1	bitmap	*spp*=8,	0.0135/ 52.7s
	α	0.7	0.699	DBM, 2-stage	
	σ_s	0.1	0.429	$h(\alpha)_{max}$=0.7,	
	η	1.54	1.156	$h(\sigma_s)_{min}$=0.4	
Alginate (B)	c	0.1 0.1 0.1	bitmap	*spp*=8,	0.0033/ 51.1s
	α	0.7	0.501	DBM, 2-stage	
	σ_s	0.1	0.573	$h(\alpha)_{max}$=0.6,	
	η	1.54	1.397	$h(\sigma_s)_{min}$=0.5	
Birdy (R)	c	bitmap	bitmap	*spp*=4,	0.2711/ 347.8s
	α	0.5	0.246	MSE	
	η	1.5	1.435		
Birdy (SS)	c	bitmap	bitmap	*spp*=4,	0.1511/ 516.6s
	α	0.5	0.813	MSE,	
	η	1.5	1.172	lr:p=0.0003	
	p	ss	shape		

Table 2: Results for real-world data. G=Green, B=Blue, R=Reconstructed (using Metashape), SS=Scaled Sphere, lr=learning rate.

third row of Fig. 4, we observe visually acceptable results, whereas numerical results degrade when reducing the samples from 16 spp to 8. In this experiment, the η parameter initially increases (moving away from the expected reference value), and subsequently, the extinction coefficient (σ_t) remains not recovered accurately. Also note, although the resulting convergence plot and image (Figs. 3, 4) indicate a successful attempt, the numerical results, specifically the η parameter is far from the reference value (third row in Table 1). Consequently, one should note that—specifically in the case of real-world data, where the target value is unknown—one cannot fully rely on the resulting visual representations. Therefore for accurate results, one must also consider the plausibility of the resulting numerical values.

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 (fourth row in Table 1), resulting in a visually noticeable mismatch between the optimization result and the reference. Consequently, in some cases, we suggest separating the optimization procedure into two parts, to acquire acceptable results, as shown in the last row of Fig. 2.

5.2 Real-World applications

In this section, we perform material reconstruction from real-world data. In particular, we aim to acquire optical material properties for two novel alginate specimens [18]. As these alginate specimens are relatively thin, we use the *Principled* BSDF [3, 9], which has proved easier to optimize in the test cases discussed previously.

Please note that we use a single for the alginate and multiple reference image(s) for the bird statue test cases. Furthermore, note that interreflections play a role in the alginate test cases, albeit we kindly remind that they were photographed in the material scanner without the influence of external objects or light sources. On the other hand, interreflections do not play a role in the bird statue experiments. Note also in the bird statue experiments we specifically use Mitsuba's constant emitter as the primary light source.

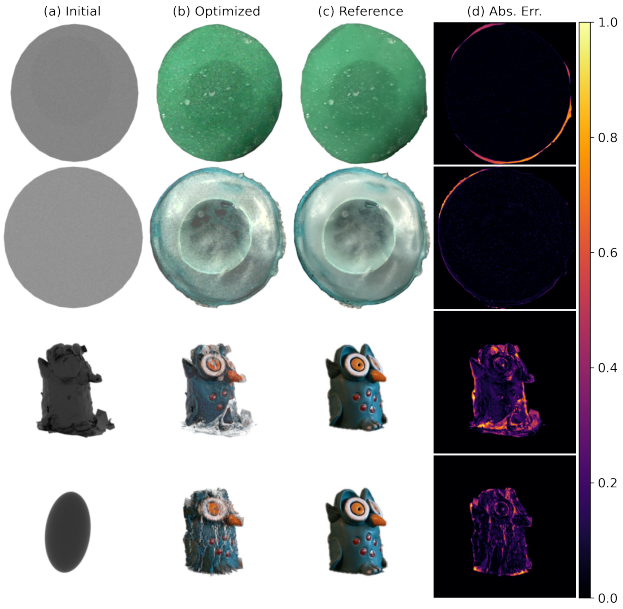

Figure 5: Results from alginate [18] materials (1. and 2. row) and a bird statue (3. and 4. row). For corresponding parameter values please refer to Table 2.

Figures 5, 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. 5. Table 2 summarizes the numerical results corresponding to Fig. 5.

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. Note also how the two-stage strategy is noticeable in the convergence plots (Fig. 6). For both alginate materials, we initialize an RGB texture with the previously optimized base color (c) at the start of the second stage (iteration 50). In the second stage of the optimization, we also observe a

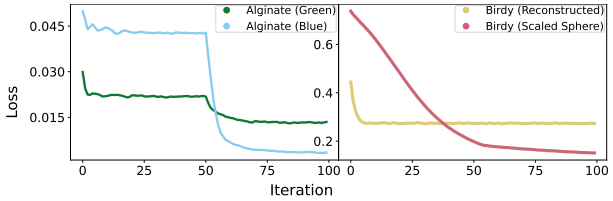

Figure 6: Convergence plots from real-world data.

noticeable decrease in loss. Consequently, the texture plays a significant role in representing the object as it contains a substantial amount of information.

Finally, in Fig. 5 and Table 2, we also show the results of a more complex opaque object. In Fig. 5, the third row showcases results utilizing the geometry obtained through Metashape [1], where only the c, η, and α parameters of the object were optimized. The fourth row features results using a (scaled) sphere object, with optimization of c, α, η, and vertex positions (p) parameters. Note that while we optimize the p parameter of the scaled sphere object, we use a box constraint, namely the bounding box value of the reconstructed object (third row in Fig. 5).

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, and thorough experimentation, material and geometry reconstruction can be successfully achieved.

6 Conclusion

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

A difficulty we noted was the scene and image acquisition complexity. Regarding translucent materials, we observed that the noise introduced by MC sampling affected the parameter estimation procedure. Employing the dual buffer method noticeably improved our results. Our analysis showed that certain parameters, like the index of refraction, created discontinuities in scattering behaviour and made it difficult to achieve accurate convergence. In some cases, we found that two-stage optimization was necessary. When optimizing a texture for a material's albedo, highly localized parameters can have a stronger impact than global material parameters, which can lead to inaccurate reconstruction. Our findings showed that additional constraints, such as imposing a max. clamp value on some parameters, were necessary for certain scenarios.

In the future, we aim to improve our geometric modelling and image acquisition procedures, which will allow for a more accurate representation of physics-based reality. 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.

Acknowledgements

We thank Prof. Peter Ferschin (TU Wien) and his former master's student Cheng Shi for providing the material scanner. We thank Prof. Stavric and colleagues at TU Graz for the alginate samples. We also thank Prof. Torsten Möller (University of Vienna), for his help during the topic discovery phase. Lastly, thanks to the Mitsuba team at EPFL in Switzerland, which provided an incredible tool for this author and the whole computer graphics community.

References

[1] AgiSoft. Metashape, 2022. `https://www.agisoft.com/`.

[2] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, jul 1977.

[3] Brent Burley. Physically-based shading at disney. *ACM Trans. Graph. (SIGGRAPH)*, 2012.

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation.

[5] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81, page 307–316, New York, NY, USA, 1981. Association for Computing Machinery.

[6] Delatronic. Dragon. `https://blendswap.com/blend/15891`, 21.08.2015.

[7] Xi Deng, Fujun Luan, Bruce Walter, Kavita Bala, and Steve Marschner. Reconstructing translucent objects using differentiable rendering. In *ACM SIGGRAPH 2022 Conference Proc.*

[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*.

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

[11] Wenzel Jakob, Sébastien 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] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, aug 1986.

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

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

[15] 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):735–745, jul 2006.

[16] 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.

[17] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.*, 38(6), 2019.

[18] Ivan Marjanovic, Elizabeta Samec, Hana Vasatko, and Milena Stavric. Alginate in architecture: An experimental approach to the new sustainable building material. In *Art and Science Applied: Experience and Vision*, volume 2 of *SmartArt*, chapter 21, pages 394–406. 2022.

[19] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, jul 2003.

[20] Baptiste Nicolet. Mitsuba blender add-on. `https://github.com/mitsuba-renderer/mitsuba-blender`, 2022.

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

[22] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.*, 39(4), 2020.

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

[24] Matt Phar. Let's stop calling it ggx. `https://pharr.org/matt/blog/2022/05/06/trowbridge-reitz`, 2023. [Online; accessed 22-February-2023].

[25] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (3rd ed.)*. 3rd edition, oct 2016.

[26] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[27] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. 3rd edition, 2009.

[28] Tanaboon Tongbuasirilai, Jonas Unger, Joel Kronander, and Murat Kurt. Compact and intuitive data-driven brdf models. *The Visual Computer*, 36(4):855–872, 2019.

[29] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces∗. *J. Opt. Soc. Am.*, 57(9):1105–1114, Sep 1967.

[30] T. S. Trowbridge and K. P. Reitz. Average irregularity representation of a rough surface for ray reflection. *J. Opt. Soc. Am.*, 65(5):531–536, May 1975.

[31] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proc. SIGGRAPH '94*, page 311–318. ACM, 1994.

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

[33] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *ACM Trans. Graph.*, 40(4):108:1–108:14, 2021.

[34] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proc. of the 18th EGSR'07*, page 195–206.

[35] Jingjie Yang and Shuangjiu Xiao. An inverse rendering approach for heterogeneous translucent materials. In *Proc. of the 15th ACM SIGGRAPH*, page 79–88.

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

[37] 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.

[38] Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. Physics-based differentiable rendering: From theory to implementation. In *ACM SIGGRAPH 2020 Courses*.