




# Translucent Material Estimation

23.02.2023, Saip-Can Hasbay  
(01428723)



Hello everyone.

Today, I will discuss the project I have been working on for the last two to three months.

So since presentations are not something that I do quite often, I would appreciate if you be patience with me.

We have been told that you can ask questions after the presentation.

However, please feel free to interrupt me if something needs to be clarified.

# Outline

- Introduction: Essential ideas
- Problem and Goals
- Method
- Results
- Hopefully: Live example or a video

Let us begin.

Here is the outline for the presentation.

First, I will briefly introduce the essential ideas utilized in this project and this section will also contain some information regarding previous work.

Next, I will discuss the problem and goals of this project.

Then, I will briefly introduce our approach and specifically talk about the procedure we implemented.

And lastly, we will show our results.

Hopefully, depending on the remaining time, I will also show a live example using our tool.

# Introduction

Physics-Based Differentiable Rendering  
A Comprehensive Introduction. SIGGRAPH  
2020 Course, Zhao et al.[1]



Rendering



$$f(x) = y$$



"Inverse Rendering"

$$x = f^{-1}(y)$$



Scene description: geometry,  
materials, lights, etc.

Before we start, regarding the background material to physics-based differentiable rendering, a fantastic SIGGRAPH 2020 course from Zhao et al. provides a great introduction to the field.

With that let me begin with the topic.

Rendering is a relatively well-known problem.

So imagine you are trying to render a scene.

The procedure requires inputs such as objects, light positions, material information, etc.

We often describe all this input as a high-dimensional vector  $x$  that contains all that information.

And then rendering is a function  $f$  that map this input into another vector  $y$ , where  $y$  is the generated image.

In today's presentation, I'd like to talk about the opposite of this procedure, which can be called inverse rendering.

In inverse rendering our goal is to invert this function and obtain the scene description  $x$  from an image  $y$ .

Our focus will be on physics-based inverse rendering.

Which incorporates many different difficulties.

# Difficulties

- Physics-based (inverse) rendering
  - Scattering effects, complex materials, global illumination

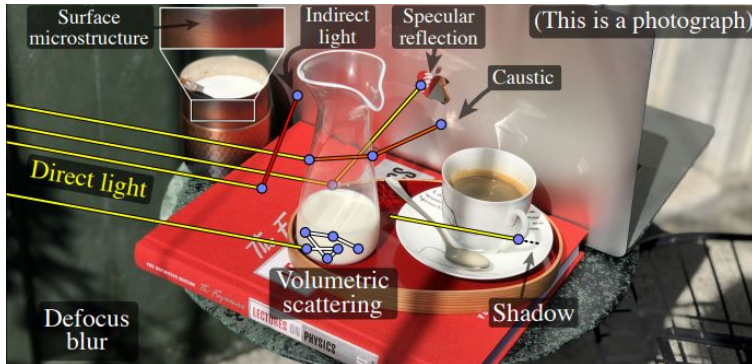


Image from Zhao et al. [1]

Consequently, the task of inverse rendering is not that easy since scattering effects from complex materials play a vital role.

Intuitively, one could maybe think this is not a complicated problem.

For example, if we have an image of a red book, one would instinctively argue that the scene description should contain a book object with red material.

However, we should remember that our goal is to describe reality in a physically accurate manner.

The red book in the image is actually a combination of red pixels, and there could be several reasons why the image contains red pixels.



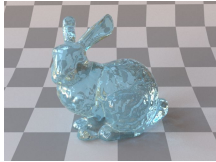
In the simplest case, the red pixel might result from the red-colored diffuse material.

However, it might also be the result of an indirect global illumination effect, such as a reflection of a red object on the computer.

So in summary, since the given scene description is high-dimensional and complex and the generated image depends on the complexity of that, there is often not a direct way to find the inverse function to retrieve parameters from an image.

## Objective Function

$$g(\text{img}) = \left\| \text{img} - \text{Target} \right\|^2$$

**Rendering** **Target**

minimize  $g(f(x))$

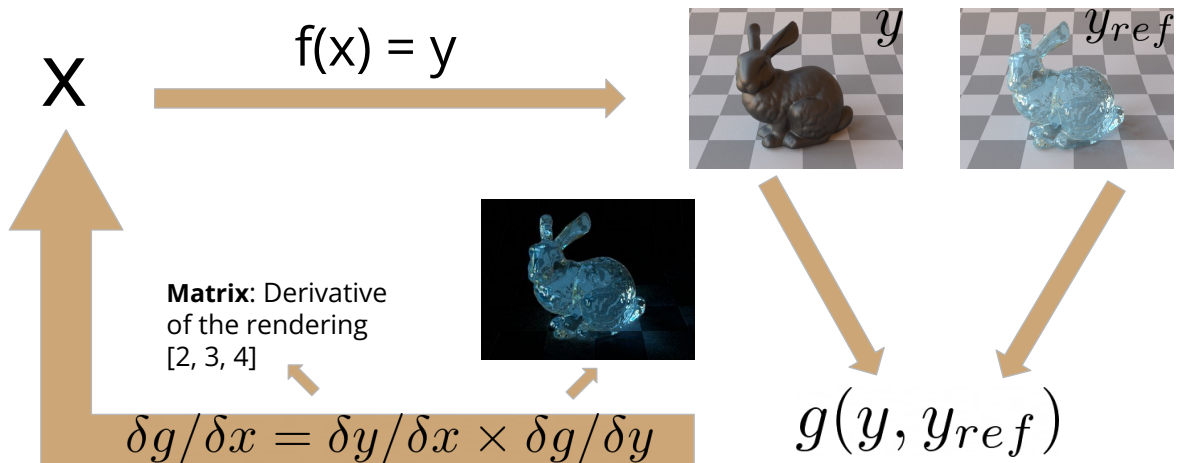
Consequently, the task of inverse rendering is often defined as an optimization problem.

With this, we introduce an objective function that quantifies the quality of our current solution.

In the simplest form, the objective takes the L2 norm of the difference between the rendered and the reference image.

So we basically take our rendering algorithm  $f(x)$  and plug it into this objective function  $g$ , and subsequently, we try the best set of scene parameters  $x$  that brings us closer to the target.

# Differentiable Rendering



But as mentioned previously, the domain we are in is quite high dimensional. Consequently, the problem often is changed into a local optimization problem based on gradient descent.

At each iteration, we evaluate the objective function with our current state and differentiate the scene parameters respect to this value.

Mitsuba--which is the renderer we primarily used in this project, makes things a little easier by using the chain rule.

I will not go into too much detail here, but the idea is differentiating the objective function first and then differentiate the rendering algorithm after and multiply them to get gradients.

So, if we differentiate the objective function, in the case of L2 norm we get an image like this--which is quite simple to understand.

It basically tells us the difference between where we are and our target.

The second part is differentiating the rendering algorithm--which eventually gives us a matrix of derivatives from the rendering algorithm.

And consequently, we multiply these two to get the gradient that we need to apply to the scene and get closer to our goal.

The tricky thing is differentiating the rendering algorithm, and fortunately, Mitsuba 3 handles this quite efficiently using radiative backpropagation algorithm and also an extended version of that that introduced Vicini et al in 2021.

But, for conciseness I will not discuss further.

However, if you are interested I would definitely recommend the referenced papers. With this let me switch to the goals of our project.



## Problem and Goals (1)

- Main task: Translucent material parameter estimation
- But also:
  - A tool for inverse rendering
  - A gradient based optimization algorithm
  - A workflow for data acquisition
  - A naive approach for geometry and material reconstruction



*Optimized material from synthetic (left) and real-world (right) alginate [5] data. Dragon model by Delatronic [14]*

So, our main goal is to estimate translucent material parameters.

But, with this project, we also provide a tool for inverse rendering and data management for that procedure.

We implement a gradient based optimization algorithm that tries to accomplish the task of inverse rendering.

We also provide a couple of workflows for data acquisition, specifically image and scene file acquisition.

And lastly, we propose a naive approach for geometry reconstruction—which I personally find also very interesting.

On the right hand side you can see two images of reconstructed materials of a Dragon object.

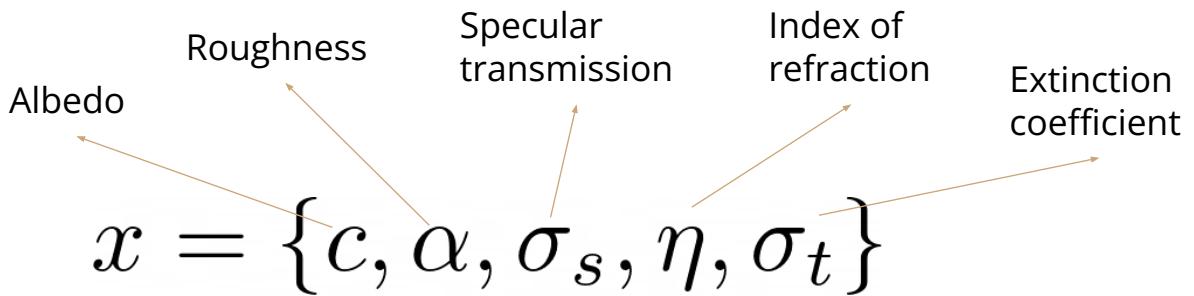
The left one is the optimization result from synthetic data and the right one is from real-world alginate material which were provided to us from Prof. Fersch TU Vienna and Prof. Starvic TU Graz.



## Problem and Goals (2)

Material parameters of interest:

- (1) Disney Principled BSDF with integrated subsurface scattering [7, 8]
- (2) Volumetric rendering: Rough dielectric BSDF [9] with homogeneous participating medium [6]



Previously we mentioned that  $x$  contains scene description information.

We further specify  $x$  with the following parameters for the task of translucent material estimation.

We use two BSDF models, one is the surface only Disney Principled BSDF from Burley et al. 2015.

And the second for volumetric rendering which combines Mitsuba's rough dielectric BSDF from Walter et al 2007, with homogeneous participating medium.

As can be seen here, the main parameter of interests are the albedo which represents the color obviously, surface roughness, specular transmission—which is used and introduced by Burley et al—index of refraction and extinction coefficient.

## Problem and Goals (3)

- Estimate material parameters:  $x = \{c, \alpha, \sigma_s, \eta, \sigma_t\}$
- By defining the task as an optimization problem

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

where  $h$  defines additional constraints and  $g$  is either

- the L2 norm  $g(y) = ||y - y_{ref}||^2$ , or
- the dual buffer method by Deng et al. [13]:

$$g(y_1, y_2) = (y_1 - y_{ref}) \cdot (y_2 - y_{ref})$$

This slide is more or less for recap purposes, so I will kind of repeat myself.

Here is the material parameters that we would like to estimate.

We define our task as an optimization function using the objective function  $g$  and successively repeat to get closer to our target.

What we never mentioned previously is the term  $h$ , which basically provides additional constraints such as minimum and maximum values for material parameters.

For the optimization function, in most simple cases we use the L2 norm, which takes the current image and a target or reference image as input.

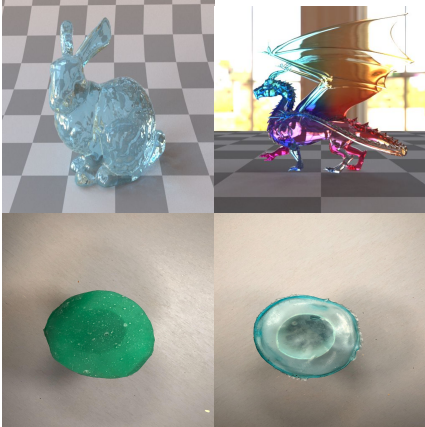
In our tool, for the optimization function we also use the dual buffer method by Deng et al, which takes rather than one differentiable rendering step two independent differentiable rendering steps.

In practice their method provided us with good results to address the noise introduced by monte carlo rendering.

# Problem and Goals (4)

## Requirements

- (1) a (set of) reference image(s)
- (2) a Mitsuba scene



Example reference images we used in our project. Top: Synthetic test cases. Bottom: Real-World algal materials [5].

```
<scene version="3.0.0">
  <!-- Integrator -->
  <integrator type='$integrator'>
    <integer name="max_depth" value="$max_depth"/>
  </integrator>
  <!-- Sensor -->
  <sensor type="perspective" id="sensor">
    ...
  </sensor>
  <!-- BSDFs -->
  <bsdf type="diffuse" id="white">
    <rgb name="reflectance" value="0.885809, 0.698859, 0.666422"/>
  </bsdf>
  <bsdf type="dielectric" id="glass"/>
  <!-- Light -->
  <shape type="obj" id="light">
    <string name="filename" value="meshes/cbox_luminaire.obj"/>
    <ref id="white"/>
    <emitter type="area">
      <rgb name="radiance" value="18.387, 13.9873, 6.75357"/>
    </emitter>
  </shape>
  <!-- Shapes -->
  <shape type="obj" id="floor">
    <string name="filename" value="meshes/cbox_floor.obj"/>
    <ref id="white"/>
  </shape>
  <shape type="sphere" id="glasssphere">
    <ref id="glass"/>
  </shape>
</scene>
```

Example (simple) Mitsuba scene [5].

Before I discuss the method, let me note two requirements for our tool.

The first requirement is a (set of) reference image(s).

On the top you see some scene reference images for the synthetic test cases and on the bottom reference images for the real world algal material specimens.

Left below we show some of the reference images that we used both in the thesis and paper.

The second one is a valid Mitsuba scene, which a simple one is shown on the right.

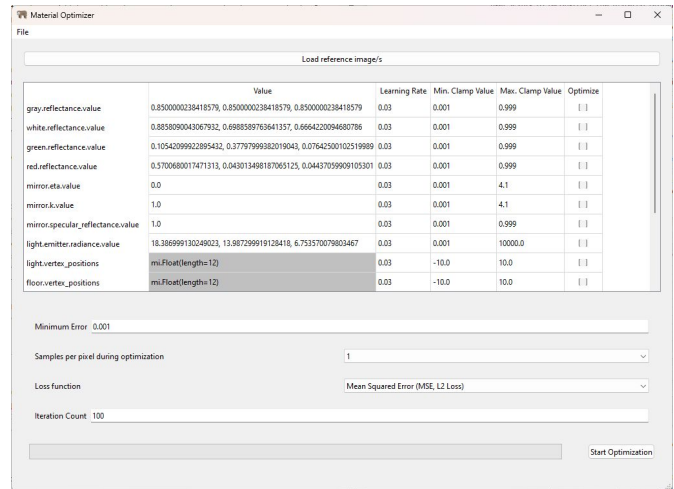
You can see the definition of the integrator, sensor—which refers to the available camera poses—material models/bsdfs, lights/emitters and the objects in the scene.

With that, let me continue with our method.

# Method (1)

Using our tool:

1. Load a scene file which includes material parameters  $x_0$ .
2. Load a (set of) reference image(s).
3. Select  $\mathcal{X}$ , which gets assigned to initialized ADAM optimizer.
4. (Optional) Select optimization hyperparameters (e.g. iteration count).
5. Start the optimization loop.



User interface of our tool.

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

We first load a scene file, which includes the initial material parameters. We then also load the (set of) reference image(s), which will be used in the objective function.

Next, we select the material parameters of interest, which get assigned to a newly initialized ADAM optimizer by our tool.

Optionally, we also select the optimization hyper-parameters and then start the optimization loop.

On the right you see the UI of our tool.

In the middle you see a table, where rows refer to the optimizable scene parameters.

On the bottom you see the optimization hyperparameters.

Users can start the optimization loop by simply clicking this button right here.



## Method (2)

### *Optimization loop*

Our tool initializes  $\mathcal{X}_i = \mathcal{X}_0$ , and runs for each camera pose (i.e. reference image):

1. Perform a differentiable rendering step with respect to  $\mathcal{X}_i$  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  $x_{i+1}$ .
7. Repeat until either the loss tolerance or the maximal iterations is reached.

Let me continue with the optimization loop.

Our tool initializes  $x_i = x_0$ , and then runs the following optimization loop for each sensor (i.e.~reference image):

We perform a differentiable rendering step with respect to  $x_i$  resulting in an image  $y_{\{i\}}$ .

Then we evaluate the objective function  $g(y_{\{i\}})$ .

Back-propagate  $\delta g / \delta y$  using Mitsuba~3, to obtain  $\delta g / \delta x_i$ .

Take an ADAM optimization step to find updated parameters  $\tilde{x}_{\{i+1\}}$ .

Ensure legal values for  $x_{\{i+1\}}$  by clamping  $\tilde{x}_{\{i+1\}}$  using box constraints.

Update the scene with  $x_{\{i+1\}}$ .

Repeat until either the loss tolerance or the maximal iterations is reached.

This is basically the computational part of our approach and in practice we obtained relatively acceptable results.

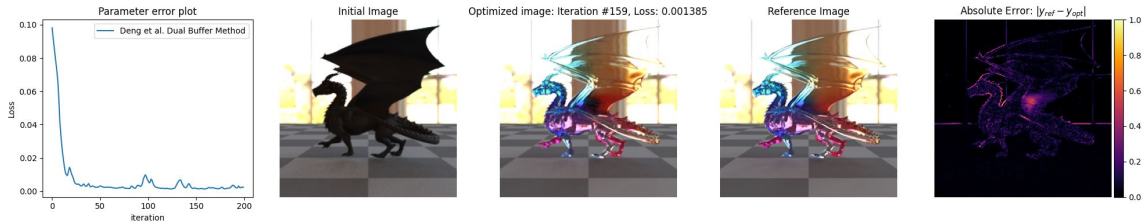
Unfortunately, we do not have too much time to discuss the image and scene file acquisition methods we used.

However, if you are interested you can find more information in the thesis.

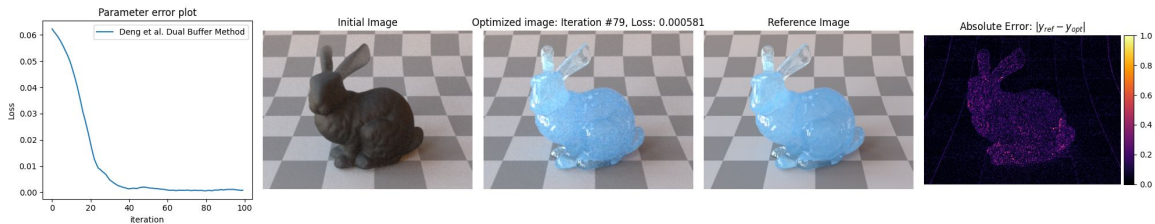
So, with that said, let me continue with our results.

# Results: Synthetic Data

Disney Principled BSDF with integrated subsurface scattering [7, 8]



Rough dielectric BSDF [9] with homogeneous participating medium [6]



OK – I hope everyone feels great—until this point it has been too much information, but now begins the fun part..

In this slide, we show our results from synthetic data.

In the first row we show material parameter estimation results using Principled BSDF by Burley et al..

In the second row we show material parameter estimation results using Rough dielectric BSDF by Walter et al.

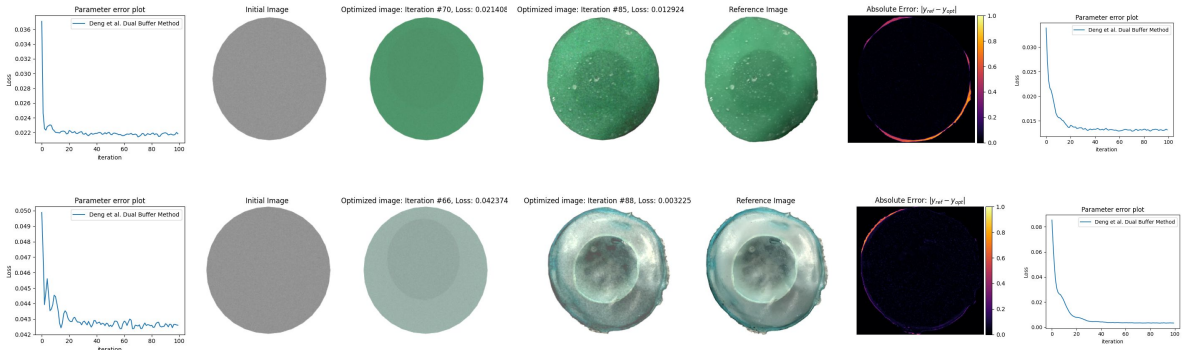
As you can see on both cases we obtained relatively good/acceptable results.

We discuss much more in detail regarding our synthetic test cases, so if interested, I highly recommend reading the thesis.

Specifically in the thesis, the appendix contains extra 15 pages of extra content—which I had a lot of fun while writing it.

# Results: Alginate [5] specimens (real-world)

Disney Principled BSDF with integrated subsurface scattering [7, 8]



*Plots Left/Right: Parameter error plot from the first/second optimization.*

*Images-Left to right: (1) Initial image. (2/3) Optimized image from the first and second part of the optimization. (4) Reference. (5) Absolute error.*

In this slide we see our results from real world alginate specimens.

In both rows we show material parameter estimation results using Principled BSDF by Burley et al..

For both of these examples we divided the optimization procedure into two parts.

First we optimized the material parameters using a constant RGB value.

Then we restarted the optimization by using the optimized material parameters from the first part and changing the representation of the color into a bitmap texture.

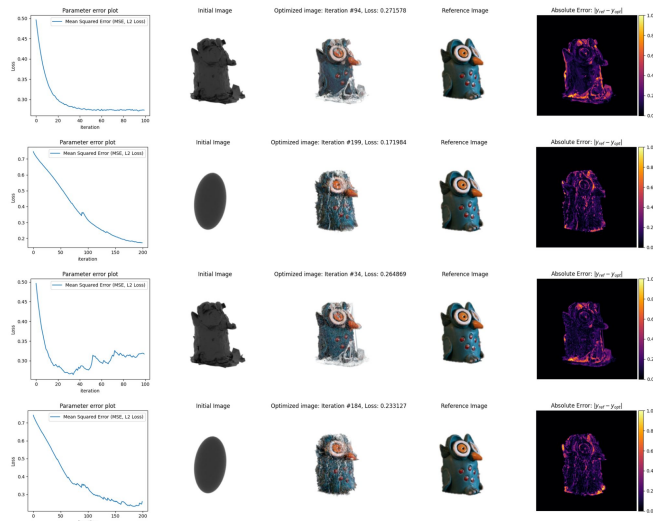
In both cases we get relatively acceptable results.

However, beware that the bitmap texture also contains a lot of information in the obtained results which might not be hundred percent accurate.

For example the shining parts that you see here is contained in the bitmap texture and does not originate from the other material parameters.



# Results: Bird statue (real-world)



Geometry and material estimation. Rows: (1) Material estimation using reconstructed mesh from Metashape [16] (2) Texture and geometry estimation. (3) Material and geometry estimation using reconstructed mesh from Metashape. (4) Geometry and material estimation.

Ok, so – In this slide we show results from a real-world bird statue.

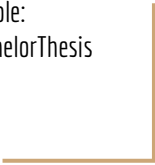
The first and third rows uses the geometry reconstruction acquisition procedure we mention in the thesis.

The second and fourth row shows the naive geometry procedure that I mentioned at the beginning, which tries to reconstruct the statue from a scaled sphere object and also the material.



# Thank you for your attention

Thesis/paper/code available:  
<https://github.com/sapo17/BachelorThesis>



It must have been too much information for 15 minutes, but I tried to give a glimpse of what I have been working on for the last months.

Not sure if we have time for a live-example from our tool—but I could also show another time.

Please let me know if I can help you with any of your questions.

You can find the thesis, paper and tool in this repository.

Thank you very much for your attention

# References (1)

[1] Physics-Based Differentiable Rendering: A Comprehensive Introduction. SIGGRAPH 2020 Course, Zhao et al.

[2] Nimier-David, M., Speierer, S., Ruiz, B., and Jakob, W. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. Transactions on Graphics (Proceedings of SIGGRAPH) 39, 4 (July 2020).

[3] Zeltner, T., Speierer, S., Georgiev, I., and Jakob, W. Monte carlo estimators for differential light transport. ACM Trans. Graph. 40, 4 (jul 2021).

## References (2)

[4] Vicini, D., Speierer, S., and Jakob, W. Path replay backpropagation: Differentiating light paths using constant memory and linear time. Transactions on Graphics (Proceedings of SIGGRAPH) 40, 4 (Aug. 2021), 108:1–108:14

[5] Marjanovi, I., amec, E., Vaatko, H., and Stavri, M. Alginate in architecture : An experimental approach to the new sustainable building material. In Art and Science Applied: Experience and Vision, . and . , Eds., vol. 2 of SmartArt. , , , 2022, ch. 21, pp. 394–406. 21.

## References (3)

[6] Jakob, W., Speierer, S., Roussel, N., Nimier-David, M., Vicini, D., Zeltner, T., Nicolet, B., Crespo, M., Leroy, V., and Zhang, Z. Mitsuba 3 renderer, 2022.  
<https://mitsuba-renderer.org>.

[7] Burley, B. Physically-based shading at disney.

[8] Hill, S., McAuley, S., Burley, B., Chan, D., Fascione, L., Iwanicki, M., Hoffman, N., Jakob, W., Neubelt, D., Pesce, A., and Pettineo, M. Physically based shading in theory and practice. In ACM SIGGRAPH 2015 Courses (New York, NY, USA, 2015), SIGGRAPH '15, Association for Computing Machinery.

## References (4)

[9] Walter, B., Marschner, S. R., Li, H., and Torrance, K. E. Microfacet models for refraction through rough surfaces. In Proceedings of the 18th Eurographics Conference on Rendering Techniques (Goslar, DEU, 2007), EGSR'07, Eurographics Association, p. 195206

[10] AgiSoft. Agisoft metashape standard, 2022. <https://www.agisoft.com/>.

[11] Community, B. O. Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022.

## References (5)

[12] <baptiste.nicolet@epfl.ch>, B. N. Mitsuba blender add-on. <https://github.com/mitsuba-renderer/mitsuba-blender>, 2022

[13] Deng, X., Luan, F., Walter, B., Bala, K., and Marschner, S. Reconstructing translucent objects using differentiable rendering. In ACM SIGGRAPH 2022 Conference Proceedings (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery.

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

## References (6)

- [15] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization, 2014.
- [16] AgiSoft. Agisoft metashape standard, 2022. <https://www.agisoft.com/>.