FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Thomas Klaus Nindel

Appearance matching and fabrication using differentiable material models

Department of Software and Computer Science Education

Supervisor of the doctoral thesis:	Prof. Dr. Alexander Wilkie
Study programme:	Informatics
Study branch:	Computer Graphics and Game Development

Prague 2022

I am dedicating this thesis to my family, and in memory of my initial supervisor, Jaroslav Křivánek. Jaroslav was the reason I pursued a doctoral degree by rekindling my passion for computer graphics. Although he was with me in the first year, he is unable to see my graduation. Thank you to my supervisor Alexander Wilkie, who continued to guide me through this process, and to all colleagues of the Computer Graphics Group for the thriving atmosphere. Further, my thanks go to Berufsakademie Sachsen for their benevolent concessions that made the pursuit of the degree feasible. Finally, thank you Elena for proof reading.

For Julia and Emil.

This work has received funding from the GA UK projects 1164620 and 148222, Czech Science Foundation under grants numbers GAČR19-07626S and GAČR-22-22875S, and from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 956585. This work was further supported by the Charles University grant SVV-260588.

Title: Appearance matching and fabrication using differentiable material models

Author: Thomas Klaus Nindel

Department: Department of Software and Computer Science Education

Supervisor: Prof. Dr. Alexander Wilkie, Department of Software and Computer Science Education

Abstract: Computing derivatives of code - with code - is one of the key enablers of the machine learning revolution. In computer graphics, automatic differentiation allows to solve inverse rendering problems. There, parameters such as an object's reflectance, position, or the scattering- and absorption coefficients of a volume, are recovered from one or several input images. In this work, we consider appearance matching and fabrication problems, that can be cast as instances of inverse rendering problems. While gradient-based optimization that is enabled by differentiable programs has the potential to yield very good results, it requires proper handling - differentiable rendering is not a shotgun-type problem solver. We discuss both theoretical concepts and the practical implementation of differentiable rendering algorithms, and show how they connect to different appearance matching problems.

Keywords: optimization, automatic differentiation, rendering, inverse rendering, differentiable rendering, appearance, appearance matching, appearance fabrication

Contents

Introduction

Great advances have been made in computer graphics when it comes to realistic image synthesis algorithms. We are now able to accurately *predict* the behavior of light as it interacts with matter, by using light transport simulations built on top of physical principles. The degree of attainable realism no longer depends on the image formation process itself, but now critically depends on good material *models* that describe the light-matter interaction, both in a principled sense, and with respect to the concrete model's parameters.

One approach to finding such a parameter set is the use of analysis-per-synthesis. There, optimization techniques are used to find a solution of the appearance-matching problem and calculate the material model's parameters. Depending on the parameter domain, different optimization strategies can be used, with gradient-based optimization being used often for its simplicity and favorable convergence behavior, especially in the light of the last decades advances in the field of gradient-based optimization algorithms. This, however, requires the calculation of gradients for the light transport simulation algorithms and material models involved, which is a very challenging problem.

Automatic differentiation, a technique invented in the 1960s by Robert Edwin Wengert, allows one to calculate the derivative of an (almost) arbitrary program with respect to its inputs. This is based on the assumption that the numerical computation performed by every program can be deconstructed into its elementary arithmetic operations. By calculating the derivative of each of these operations and applying the chain rule, the derivative of the entire program is obtained. One popular application of this technique is in machine learning. There, the gradient of the neural network's outputs with respect to the network's weights is used together with a gradient-descent optimizer find a (locally) optimal parameter set. This process of calculating the network's derivatives, which is called "back-propagation" in the machine learning field, is an instance of reverse-mode automatic differentiation.

In contrast to "top down" tabulated material models, where the reflectance properties of an object's surface are essentially scanned and stored, procedural material appearance models aim to describe the surface reflectance "bottom up" using a computational graph based on noise and hashing functions and transformations. This offsets the extreme storage requirements of the tabulated data approaches with an increased computational cost for the evaluation of the procedural model.

Using differentiable procedural appearance models allows to employ gradient-based optimization to automatically home in on a suitable parameter set that is locally optimal for approximating the intended material-light interaction. Matching procedural appearance then becomes critically dependent on the expressive power of the model, with generalized procedural models being able to cover a significant range of materials, but not all of it. This necessitates using material-specific models that may even provide information about the underlying object anatomy, if the model was built from first principles.

Computational fabrication using PolyJet 3D printers poses a similar appearance matching situation. In PolyJet 3D printers, scattering ink droplets are deposited one by one using jetting technology and rapidly hardened using ultraviolet light. The droplets form a heterogeneous participating medium. Using a differentiable volume renderer allows one to calculate the sensitivity of the printed objects' appearance to changes of the volume's ink composition. Optimizing these parameters then allows for the calculation of printing instructions that result in the best possible reproduction of the input appearance. If the input was acquired using material scanning, this closes the loop and allows for a ”replicator” type system that can, essentially, copy objects.

Exemplified by these two use-cases, this work discusses the challenges of appearance matching using gradient-based optimization. Specifically, the contributions include

- A inverse rendering approach to automatically matching the appearance of an anatomically meaningful procedural wood model
- A loss function for coarse-structure matching based on signal phase in gabor space
- A novel tree-ring detector with applications in dendrochronology
- A method to calculate derivatives of discontinuous procedural solid textures
- A end-to-end differentiable pipeline for polyjet appearance optimization
- A novel parameterization of 3D print volumes based on material mixtures
- A metric that allows controlling the volume optimization to favor specific visual stimuli

1. Context

1.1 Automatic Differentiation

Many problems in science and engineering not only require calculation of some function, but also of its derivative, in code. These gradients can then be used to drive optimization algorithms, where some of them even require higher-order derivatives, such as Newton's method. This is then used to solve problems like

$$\min f(x). \tag{1.1}$$

A tempting solution is the use of finite differences, where the function is evaluated at nearby points and the definition of the derivative is applied. Using central differences with some small value ε yields

$$\frac{\partial f}{\partial x} \approx \frac{f(x-\varepsilon)+f(x+\varepsilon)}{2\varepsilon}. \tag{1.2}$$

Using this approach is a bad idea for several reasons: The result is only a biased approximation of the derivative. Evaluating it can be expensive, because for a function of N variables, it requires at least N+1 evaluations. Further, choosing a suitable value of ε can be difficult and leads to issues with numerical stability.

Computer-algebra systems provide the option to compute the derivative of a function, and some can even export source-code for the obtained expressions, giving ready-to-run solutions. The generated code, however, is not human-readable for all but the most simple expressions, and thus not maintainable.

In his dissertation, Wengert [1964] introduced automatic differentiation, and proposed what is now called **forward-mode** automatic differentiation. The key idea is decomposing the program into elementary calculations. By calculating the derivative of each of these calculations and successively applying the chain rule, the derivative of the whole program, or more precisely its Jacobian matrix, can be obtained.

If the program is a map $\mathbb{R}^N \to \mathbb{R}^M$, then its derivative is the $M \times N$ Jacobian matrix $J = [j_{mn}]$ with elements $j_{mn} = \frac{\partial f_m}{\partial x_n}$.

Let's start with an example. Suppose we want to calculate the derivative of a program that implements the function $f(x_1, x_2) : \mathbb{R}^2 \to \mathbb{R}, y = x_1 sin(x_2)$. Decomposed into elementary computations, this can be written as a computational graph shown in Figure 1.1.

In the graph, the intermediate results (outputs of each node) are shown as the edge labels w_i. Their respective derivatives are denoted $w'_i = \frac{\partial w_i}{\partial x}$. To calculate $\frac{\partial f}{\partial x_2}$, we can do a forward-mode traversal of the graph, seeding $w'_1 = 0$ and $w'_2 = 1$. The intermediate result w'_3, according to the chain rule, is

$$w'_3 = \cos(w_2)w'_2 \tag{1.3}$$

and further

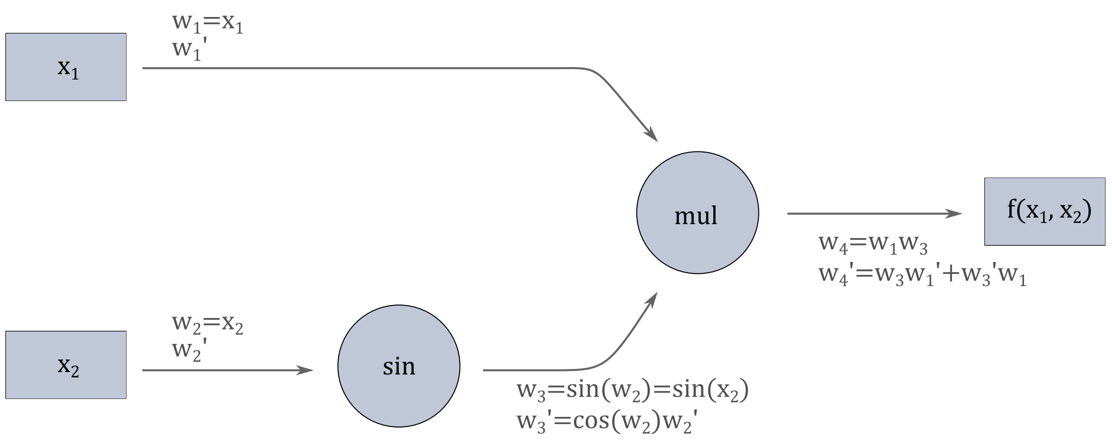


Figure 1.1: Example graph for forward-mode automatic differentiation

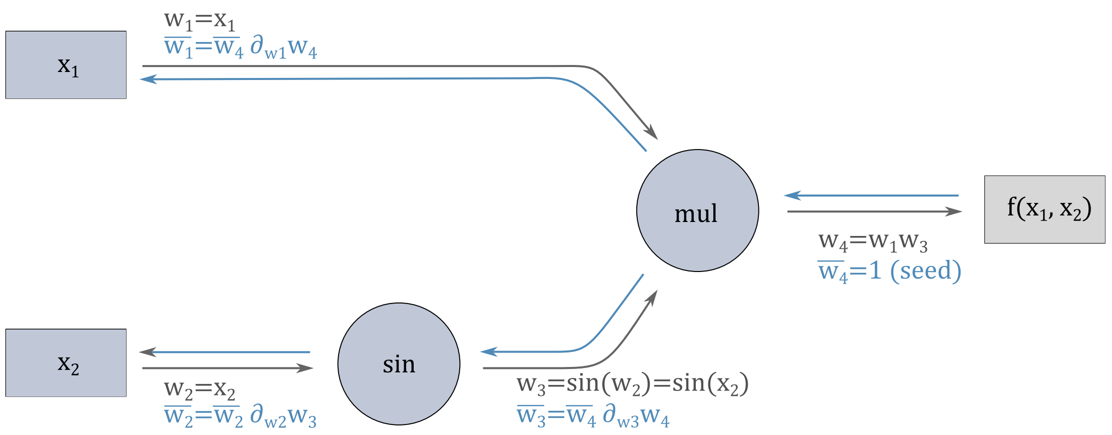
$$w_4' = w_3 w_1' + w_3' w_1 = \cos(w_2) w_2' w_1 = \cos(x_2) x_1. \tag{1.4}$$

What we have essentially done is constructing the derivative $\partial_{x_2} f$ by successively applying the chain rule

$$\frac{\partial f}{\partial x_2} = \left(\frac{\partial f}{\partial w_3} \left(\frac{\partial w_3}{\partial w_2} \left(\frac{\partial w_2}{\partial x} \right) \right) \right). \tag{1.5}$$

It is important to note that in practice, these calculations are evaluated numerically, not symbolically.

Calculating the Jacobian Matrix using forward-mode requires one graph traversal for each of the N input parameters. This can get inefficient in cases where $N \gg M$. For these programs a reverse traversal of the graph is much better. This is done by seeding the output, $\bar{w}_4 = \partial_{w_4} f$ in our example, with 1 and going backwards through the graph.



x_1
$w_1=x_1$
$\overline{w_1}=\overline{w_4}\ \partial_{w1}w_4$
mul
$f(x_1, x_2)$
$w_4=w_1w_3$
$\overline{w_4}=1$ (seed)
x_2
sin
$w_2=x_2$
$\overline{w_2}=\overline{w_2}\ \partial_{w2}w_3$
$w_3=\sin(w_2)=\sin(x_2)$
$\overline{w_3}=\overline{w_4}\ \partial_{w3}w_4$

Figure 1.2: Example graph for reverse-mode automatic differentiation. The values in the adjoint graph (in blue) flow in the opposite direction of the primal values

Figure 1.2 shows the intermediate steps with the primal values w_i, and their respective *adjoint* values $\bar{w}_i$. The intermediate adjoint values are obtained by successive substitution using the chain rule, i.e.

$$\bar{w}_3 = \frac{\partial f}{\partial w_3} = \bar{w}_4 \left(\frac{\partial w_4}{\partial w_3} \right) = \frac{\partial f}{\partial w_4} \frac{\partial w_4}{\partial w_3}, \tag{1.6}$$

and so forth. The chain-rule is effectively nested from back to front, like so:

$$\frac{\partial f}{\partial x_2} = \left(\left(\left(\frac{\partial f}{\partial w_3} \right) \frac{\partial w_3}{\partial w_2} \right) \frac{\partial w_2}{\partial x_2} \right) \tag{1.7}$$

This is called **reverse-mode** automatic differentiation, Bartholomew-Biggs et al. [2000]. It requires only one graph-traversal per output parameter, making it much more efficient than forward mode for functions with $N \ll M$. For optimization problems driven by some scalar loss function, this is the case ($N = 1$), and a similar reason is behind its popularity in machine learning, where it is called back-propagation. Many neural networks have way more parameters than they have outputs.

1.2 Implementing automatic differentiation

Implementations of automatic differentiation can either rely on compile-time code augmentation, or on maintaining a transcript of the performed computations and intermediate results at run-time. This transcript can later be traversed in a derivative pass. Different limitations in terms of time and memory complexity arise, depending on the implementation strategy.

Compile-time forward-mode accumulation relies on a transformation of the primal program, adding derivative computations, and running this derivative pass once for every input variable the program's derivative is to be calculated with respect to. The result is a time-complexity of $O(N)$. Most implementations are based on operator overloading. Using *type* overloading, the required multiple passes through the derivative code can be transformed to a single pass using operations on gradient vectors instead of scalars, with the resulting linear space complexity. An advantage to this strategy is its ease of implementation and the possibility for the compiler to optimize both the primal and derivative computations (Schroeder [2022]), a disadvantage is its poor scalability.

Runtime implementations of automatic differentiation are usually based on recording a transcript, called a "gradient tape" or Wengert tape. This actually allows to later apply the chain rule in an arbitrary manner, with forward-mode and reverse-mode traversals being the ends of a whole spectrum of traversal orders. This leads to the interesting question of the optimal traversal strategy, a problem known as the optimal jacobian accumulation problem (i.e. Naumann [2008]). Compiler optimization is not available for this runtime implementation, except when involving just in time compilation.

Numerous frameworks using this particular way of implementing automatic differentiation are available. Machine learning frameworks such as TensorFlow (Abadi et al. [2016]), Pytorch (Paszke et al. [2017]) or Jax (Bradbury et al. [2018]), and computer graphics focussed libraries such as Dr.Jit (Jakob et al. [2022a]) and DiffTaichi (Hu et al. [2019]) are popular examples. A notable technique here is just-in-time compilation, that can greatly improve on the efficiency and portability of the derivative computations, and allows targeting a broad range of compute devices such as wide-SIMD many-core CPUs and GPUs. With the application of automatic differentiation on inverse rendering problems, a range of domain-specific consequences arise. These are discussed in Section 1.6.

1.3 Physics of light

1.3.1 Radiometry

Light is both an electromagnetic wave and a particle, due to the wave-particle duality in quantum physics. In computer graphics and light transport simulation, the particle model is commonly used, as its propagation can be easily described using geometric ray optics. Wave optical effects like polarization and interference are negligible in many applications, but can play a role in certain situations and hence require either approximative or accurate treatment.

Light particles, or *photons*, carry a specific amount of energy E, depending on their frequency or wavelength,

$$E = hf \tag{1.8}$$

,

with the Planck constant $h \approx 6.626 \cdot 10^{-34} J \cdot Hz^{-1}$. Light in the visible spectrum has frequencies of approximately $790 \ldots 400 \cdot 10^{12} Hz$, or wavelengths of around $\lambda = 380 \ldots 750 \cdot 10^{-9} m$, respectively.

Measuring the total number of photons emitted by a hypothetical ideal point light source, per unit time, gives **radiant flux** $\Phi[Js^{-1} = W]$, Figure 1.3 left. **Radiant intensity** $I(\omega)$ $[Wsr^{-1}]$ then is a portion of this radiant flux emitted in a certain direction ω on the unit sphere $\mathbb{S}$, measured in steradians, Figure 1.3, middle. Integrating radiant intensity over all directions calculates the radiant flux, $\Phi = \int_{\mathbb{S}} I(\omega) d\omega$.

Irradiance is the radiant flux per unit area, $E(x)[Wm^{-2}]$, Figure 1.3 right image. It is important to note that to calculate radiant flux E through an area element dA that is not oriented orthogonally to the flux's direction, the factor $\cos\theta$ appears. θ measures the angle between the normal of the area element and the direction of the radiant flux, which is called the Poynting vector.

Radiance then is the radiant flux per unit area and solid angle, $L(\omega, x)$, its unitis $[Wm^{-2}sr^{-1}]$ and its relation to the radiant intensity

$$L(\omega, x) = \frac{\partial I(\omega)}{\partial A_\perp} = \frac{\partial I(\omega)}{\partial A \cos\theta}. \tag{1.9}$$

In rendering systems, radiance is the natural unit of measurement. Radiance does not change for light propagating in a vacuum in a straight line.

1.3.2 Describing surface reflectance

When light interacts with matter, it is scattered and absorbed to various proportions. This behaviour is dependent on the incident angle ω_i. Formally, a

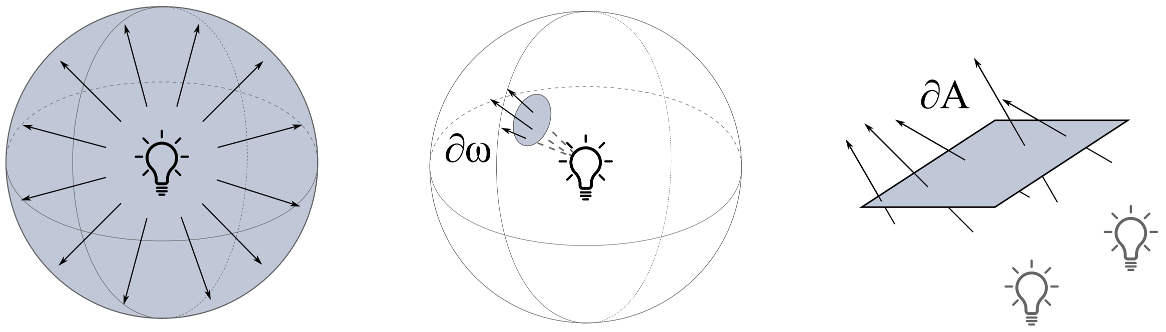


Figure 1.3: Illustration of the radiometric quantities radiant flux (left), radiant intensity (middle) and irradiance (right)

surface interaction can be described by the bidirectional subsurface scattering and reflection distribution function, or BSSRDF, Figure 1.4 right. It is a 6-dimensional function of the incident direction, the outgoing direction ω_o, the incoming surface interaction point x_i and the outgoing surface point x_o. When not considering subsurface light transport, $x_i = x_o$ (or more elaborate cases with refraction, where x_o is a non-stochastic function of the incident direction), the function can be simplified to four dimensions and split into two separate lobes, the bidirectional reflectance distribution function, BRDF, and the bidirectional transmittance distribution function, BTDF, Figure 1.4 left.

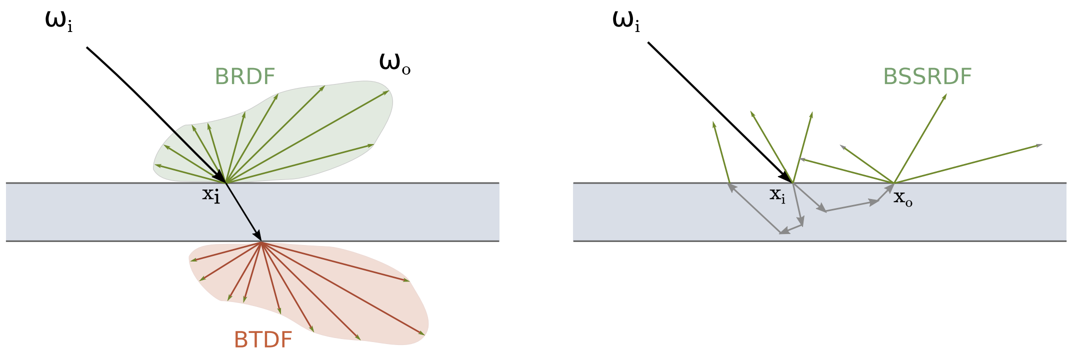


Figure 1.4: Illustration of the surface reflectance distribution function (BRDF, left), the transmittance distribution function (BTDF, left), and the subsurface reflectance distribution function (BSSRDF, right)

For all incident directions ω_i, the B*DFs need to integrate to a value smaller than 1, $\int_{\mathbb{H}} f_r(\omega_i, \omega_o) d\omega_o < 1 \; \forall \omega_i \in \mathbb{S}$, to not violate conservation of energy.

1.3.3 Volumetric light transport

Light propagation in participating media can be intuitively described as a random walk where simulated photons interact with the medium at certain points. These collision events can either be pure scattering (the simulated particle changes direction) or purely absorbing (the simulated particle vanishes). More formally, we introduce the **scattering coefficient** $\sigma_s[m^{-1}]$ and the **absorption coefficient** $\sigma_a[m^{-1}]$. Their sum is called the medium **optical density** or extinction

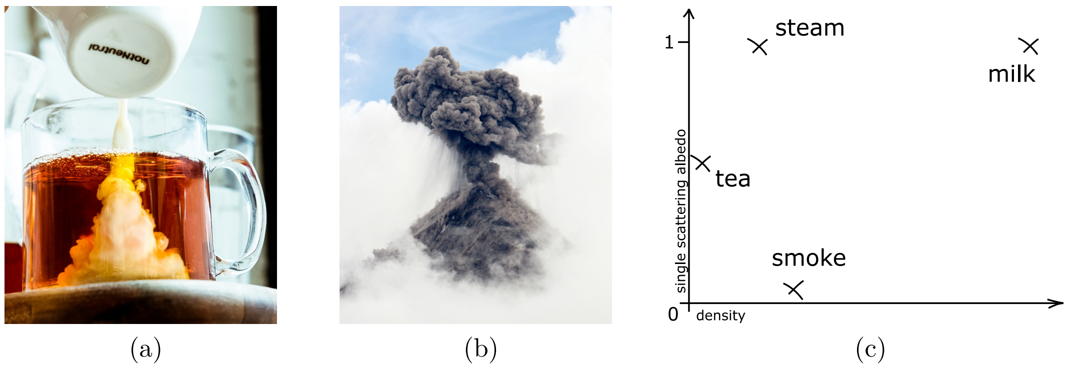


Figure 1.5: The Tea in image (a) is an absorbing medium (spectrally varying) with a very low scattering coefficient and thus has a low single scattering albedo corresponding to its dark color, and a low density. The milk in image (a), on the other hand, is low absorbing but high scattering, giving it a high single-scattering albedo and high scattering coefficient; clouds or steam (b) consists of small drop of water, a scattering but almost non-absorbing medium, whereas smoke consists of small soot particles that absorb and sometimes scatter light. Image (c) shows a qualitative sketch of the properties in the albedo/density plane

coefficient, $\sigma_t = \sigma_s + \sigma_a$, with the inverse $\sigma_t^{-1}[m]$ being the **mean free path** length of the simulated particle. The fraction of scattering events compared to all medium interactions is called the **single scattering albedo**, $\alpha = \frac{\sigma_s}{\sigma_t}$. Figure 1.5 illustrates how these properties map to the appearance of participating media.

Beer's law describes the radiant intensity I at a distance t, starting with an initial radiant intensity L_0, as

$$I(t) = I_0 \; e^{-\sigma_t t} \tag{1.10}$$

The change of direction in the case of a scattering event can be described by a probability density function, called the scattering **phase function**, $f_p(x, \omega, \omega')$, with ω and ω' being the incoming and outgoing directions, respectively. Phase functions need to integrate to 1 in order to adhere to the principle of energy conservation, $\int_{\mathbb{S}} f_p(x, \omega, \omega')d\omega') = 1$.

In the simplest case, the scattering phase function is a uniform probability density function with the value of $f_p(x, \omega, \omega') = \frac{1}{4\pi}$. Anisotropic phase functions are commonly approximated by the Heney-Greenstein function, an approximation of the more accurate (but more complex) Mie scattering phase function Mie [1908]

$$f_{p,HG}(g, \omega, \omega') = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g \;\cos(\omega, \omega')^{3/2}}. \tag{1.11}$$

The asymmetric factor $-1 \leq g \leq 1$, adjusts the probability density function between dominantly back-scattering $g < 0$ and dominantly forward scattering $g > 0$.

A point in a heterogeneous medium can now be parameterized by either the combinations of parameters $(\alpha, \sigma_t, f_p)^T$, or the triplet $(\sigma_s, \sigma_a, f_p)^T$. Only the second parameterization using σ_s and σ_a allows to aggregate material mixtures into one using linear combinations, a property that we used to parameterize the material model in our appearance fabrication contribution.

The radiative transfer equation,

$$(\omega \cdot \nabla)L(\omega) = -\sigma_t L(x,\omega) + \sigma_s \int_{\mathbb{S}^2} f_p(x,\omega,\omega')L(x,\omega')d\omega' \tag{1.12}$$

written in its integral form (called the volume rendering equation),

$$L(x,\omega) = \int_x^y \tau(x,y)\sigma_s(y)L_s(y,\omega)dy \tag{1.13}$$

with the in-scattered radiance

$$L_s(y,\omega) = \int_{\mathbb{S}^2} f_p(y,\omega,\omega')L(y,\omega')d\omega' \tag{1.14}$$

and the transmittance

$$\tau(x,y) = exp(-\int_x^y \sigma_t(s)ds) \tag{1.15}$$

describes the radiance field in a participating medium.

1.4 Light transport simulation

Computer graphics is currently split into two approaches to the generation of images: Real-time online rendering, and offline rendering, Figure 1.6 has an example of both. Both employ, at least until the time of this writing, distinctively different techniques to the image generation, adapted to the goal of the respective approaches: Real-time rendering must obey tight timing restrictions of a few milliseconds per frame, and is commonly building on rasterization as the fundamental image generation process. The available time for offline rendering is considerably longer, in the order of several minutes to hours per image are common. This allows the use of precise, physically based light transport simulations based on Monte Carlo integration. With recent developments in de-noising low sample count Monte Carlo renderings, the application of MC seems to becomes feasible also in the real time domain and could lead to a convergence of real-time and offline rendering.

Monte Carlo rendering is capable of capturing all effects of ray-optical light propagation such as subsurface scattering or caustics, providing for great realism, and has been widely adopted across many industries, from movies to architectural simulations. As an example, about half of the images on the IKEA website are renderings, not photographs (Figure 1.7).

The basic idea behind light transport simulation is to simulate the propagation of individual photons through the scene by sampling paths that connect a light

[1]Screenshot by User Asdfiel on the flightsimulator.com forums, "Asdfiel" [2021] Image "Porsche 911 Carrera T" by Gustavo Coutinho, via the Corona Renderer Gallery, Coutinho [2021]

source to the camera, via a number of scattering events at object surfaces. This is formalized in the rendering equation, Kajiya [1986]:

$$L(x,\omega) = L_e(x,\omega) + \int_{\mathbb{H}} f_r(x,\omega_i \rightarrow \omega_o) L(x,\omega_i)(\omega_i \cdot \hat{n}) d\omega_i \tag{1.16}$$

Here, L is the incident radiance at point x from direction ω, L_e is the radiance emitted by some light source, f_r describes the throughput of a surface interaction at x for incoming radiance from direction w_i towards direction w_o. The surface normal is denoted with $\hat{n}$, the vector inner product is denoted with $\cdot$.All of the functions here can be wavelength-dependent, which we will omit without loss of generality.

Path space formulation Note that this formulation requires recursively evaluating incident radiance at surface interaction points. Using a change of variables, the recursive integral can be expressed as a non-recursive integral over all *paths* $\bar{x} \in \omega$, the so-called path space integral formulation of light transport:

$$L(x,\omega) = \int_{\Omega} f(\bar{x}) d\mu(\bar{x}) \tag{1.17}$$

$$f(\bar{x}) = W_i(x_0 \leftarrow x_1) L_e(x_{l-1} \leftarrow x_l) \prod_{k=1}^{l} G(x_k \leftrightarrow x_{k+1}) M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}) \tag{1.18}$$

$f(\bar{x})$, the measurement contribution function, describes the radiance contribution of a path $\bar{x}$, calculated from emitted radiance L_e and the product of the throughput of all scattering events M (material term) and free flight G (geometry term). M models the camera pixel response (usually a box filter). Paths $\bar{x}$ consist of k segments $x_0 ... x_k$, the arrows in the notation denote individual path segments, i.e. $x_1 \rightarrow x_2$ is a path segment from x_1 to x_2, see Figure 1.8 for an sketch.

This formulation generalizes to participating media, where G then is a function of the path throughput according to Beer's law for free-flight propagation between medium scattering-events (Equation 1.20), and M (Equation 1.19) describes the scattering-event itself. Scattering events can be either at surface interactions where M is the bidirectional scattering distribution function (BSDF),

(a)

(b)

Figure 1.6: Screenshot from Microsoft Flight Simulator 2020, showing state of the art realtime graphics (a), and a off-line rendering of an outdoor scene(b)[1]

Figure 1.7: Rendered kitchen scene from the IKEA catalog, Image from Visualization [2017]

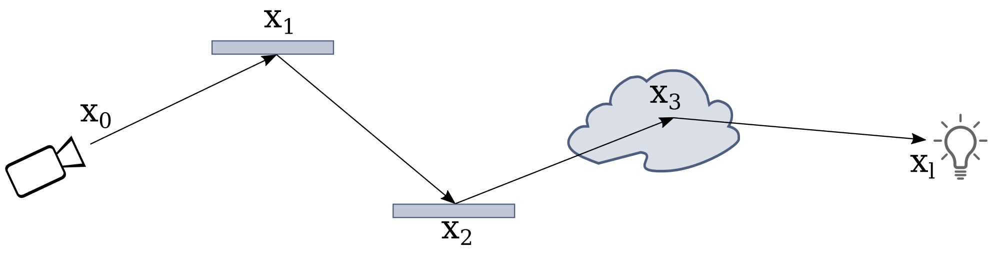


Figure 1.8: A path of length $l = 4$ with two surface interactions x_1 and x_2 and a medium interaction x_3

or at medium interactions. For medium interactions, M is a product of the single scattering albedo α, the medium density σ_t and the phase function f_p. M and G can be spatially varying for heterogeneous media.

$$M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}) = \begin{cases} \alpha(x_k)\sigma_t(x_k)f_p(x_k, \omega(x_k \leftarrow x_{k+1})) & \textit{medium} \\ f_r(x_k, \omega(x_k \leftarrow x_{k+1})) & \textit{otherwise} \end{cases} \tag{1.19}$$

$$G(x_k \leftrightarrow x_{k+1}) = \begin{cases} \tau(x_k, x_{k+1}) & \textit{medium} \\ V(x_k \leftrightarrow x_{k+1}) & \textit{otherwise} \end{cases} \tag{1.20}$$

Here, V is the binary visibility function, that is 0 when the path segment contains an occlusion.

Since path space can have very long paths, this is especially true for dense participating media, tracing them becomes intractable and a upper limit for the path length is imposed. To avoid bias resulting from truncating path space to this limit, Russian Roulette is used to terminate paths at random when they exceed a certain number of interactions.

To form an Monte Carlo estimator, the integral is converted into a sum over a certain number of evaluations of the primary estimator $\hat{X}$, which is constructed by dividing the integrand by the probability of sampling the path $\bar{x}$,

$$\int_{\Omega} f(\bar{x}) d\mu(\bar{x}) \underset{MC}{\approx} \frac{f(\bar{x})}{p(\bar{x})} = \hat{X}(\bar{x}) \tag{1.21}$$

Repeating this process N times gives the Monte Carlo estimate of the integral,

$$\int_{\Omega} f(\bar{x}) d\mu(\bar{x}) \approx \sum_{i=1}^{N} \hat{X}^{(i)}(\bar{x}) \tag{1.22}$$

with the variance decreasing with increasing N. Monte Carlo is not always necessary. For certain simple cases, analytical solutions exist (i.e. inverse adding-doubling Prahl et al. [1993]).

1.5 Sampling path space

The Monte Carlo method requires sampling paths $\bar{x}$ from path space Ω. A general strategy is to start all paths at the observer, and on each material interaction continue the path by sampling from the respective probability density function for a new outgoing direction, until a light source is hit. Depending on the scene, this sampling strategy can result in a high variance of the estimators, which is especially visible with effects caused by changes in the index of refraction (i.e. caustics), or when light sources are obstructed by geometry. Other path-sampling strategies can be used Veach [1998], such as light tracing (starting at the light source and terminating paths when they hit the camera sensor) or bidirectional path tracing, where paths are started both at light sources and the sensor, and get subsequently merged.

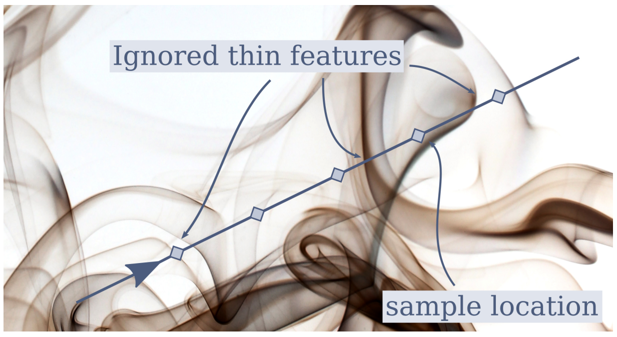


Figure 1.9: Regular tracking leads to bias

Rendering participating media adds the question of how to sample free flight distance between interactions to the path sampling strategies. Analytical solutions exist for volumes with (piece-wise) homogeneous density. In the heterogeneous case, a straightforward, albeit biased solution is sampling the volume in regular intervals (“raymarching”), Figure 1.9. An unbiased estimator can be constructed by using null-collision algorithms such as Woodcock Tracking Woodcock et al. [1965], a technique from the neutron transport community. A modification of Woodcock Tracking can also be used to importance-sample free flight distances in heterogeneous media. Miller [1967] provided a proof, Galtier et al. [2016] found a useful integral formulation.

1.6 Differentiable Light Transport Simulation

Within the path space integral framework, material appearance is represented by the M-term, Equation. 1.19. To solve material appearance matching problems, we are interested in finding an appropriate model for M, and its parameters. We want to solve this *inverse rendering problem* in an analysis-by-synthesis fashion, where from a set of observations of a scene the parameters are to be reconstructed. To that end, we will differentiate both light transport and the material model, and plug the obtained gradient values into a gradient descent optimizer.

Gradient descent has many favorable properties, such as a convergence guarantee if the parameter space is convex, and is very easy to implement using automatic differentiation. Applied on a physically based light transport simulation based on Monte Carlo integration, this creates an estimator for the derivative of the path space integral, Equation. 1.17 with respect to the material models' parameters. For the sake of simplicity, we will, without a loss of generality, only consider one differentiable parameter in the following.

Applying automatic differentiation to a Monte Carlo path tracer leads to many challenges, of which two are relevant for this thesis. First, the potentially very high computational cost when computing derivatives with respect to many input parameters, which is the case, for example, in heterogeneous volumetric appearance matching. Second, in the correct handling of discontinuities in the integrand that depend on the differential parameters. Domain-specific solutions have been proposed to address these challenges.

1.6.1 Differentiating the rendering equation

In a simplified form, the Leibniz integral rule can be applied to calculate the the derivative of the rendering equation by simply moving the differential operator inside the integral,

$$\frac{\partial}{\partial \pi} \int_\Omega f(x, \pi) d\mu(x) \underset{(1)}{=} \int_\Omega \frac{\partial}{\partial \pi} f(x, \pi) dx \tag{1.23}$$

This simplification comes with the condition that the integrand does not contain discontinuities that depend on π. Before discussing a more general form next, note the following.

Non-Commutativity of Monte Carlo estimation and the derivative operator Applying Monte Carlo and then taking the derivative

$$\int_\Omega f(\bar{x}) d\mu(\bar{x}) \approx \frac{f(\bar{x})}{p(\bar{x})} \underset{\partial_\pi}{\mapsto} \frac{\partial}{\partial \pi}\left[\frac{f(\bar{x})}{p(\bar{x})}\right] \tag{1.24}$$

is not the same as taking the derivative and then applying Monte Carlo estimation (non-commutative)

$$\int_{\Omega} f(\bar{x})d\mu(\bar{x}) \underset{\partial_\pi}{\mapsto} \int_{\Omega} \frac{\partial}{\partial \pi} f(\bar{x})d\mu(\bar{x}) \underset{MC}{\approx} \frac{\frac{\partial}{\partial \pi} f(\bar{x})}{p(\bar{x})} \tag{1.25}$$

Zeltner et al. [2021] have a detailed discussion of the implications in the context of differentiable rendering.

1.6.2 Differentiating the rendering equation in the presence of parametric discontinuities

If the integrand contains discontinuities that depend on π, the simplification in Equation 1.23 would lead to an inconsistent, biased Monte Carlo estimator. The Reynolds Transport Theorem, a generalization of the Liebniz integral rule, Vennard [2011], adds what is called the *boundary term* (second summand in Equation 1.27) in order to consider the parametric discontinuities as well. Let $D(\pi)$ be a parametric integration domain with a boundary $\partial D(\pi)$ and

$$D'(\pi) = D(\pi) - \partial D(\pi) \tag{1.26}$$

the domain's interior. The Reynolds transport theorem then states that

$$\frac{\partial}{\partial \pi} \int_D f(\omega, \pi) d\omega = \int_{D'(\pi)} \frac{\partial}{\partial \pi} f(\omega, \pi) d\omega + \int_{\partial D(\pi)} (\frac{\partial \omega}{\partial \pi} \cdot \hat{n}) f(\omega, \pi) d\partial D'(\pi) \tag{1.27}$$

Here, $\hat{n}$ is the boundary's outward pointing normal vector with the boundary movement velocity $\partial_\pi \omega$. With zero boundary velocity (or a zero boundary), this identity reduces to the simplified Leibniz integral identity of Eqn. 1.23.

Examples of parametric discontinuities include changing the shape or position of objects that can lead to changes in visibility, or changes in discontinuous procedural textures, like thresholded Perlin noise. Computer graphics made good progress towards correctly calculating derivatives of light transport in these cases.

Edge sampling, introduced by Li et. al. Li et al. [2018], computes the boundary term by explicitly sampling the silhouette edges of the scene's geometry. While being a straightforward solution, finding relevant silhouette edges can be quite difficult. In the presence of millions of triangles that make up the scene, only a small fraction of them contribute to the gradient. The search for these silhouette edges becomes a significant performance bottleneck. A more efficient solution to the problem was found by Loubet et al. [2019], followed by Bangaru et al. [2020], who both apply reparameterizations to the integral, effectively removing the discontinuity in the integrand for the parameter π, and in the process alleviating the necessity for explicit edge sampling.

This can be interpreted in two ways, as stated by Loubet et al. [2019] in their paper.

> ”Instead of integrating a function with a discontinuity whose position depends on [π] , we integrate in a space where the discontinuity does not move when [π] changes. This is equivalent to importance sampling the integral [...] using samples [...] that follow the discontinuity”

Figure 1.10 has a sketch that illustrates the concept.

Applying the Divergence Theorem gives rise to an unbiased version of these reparameterizations. The theorem relates the boundary integral to an area integral, which for the resulting Monte Carlo estimators translates to relating *edge samples to area samples.*

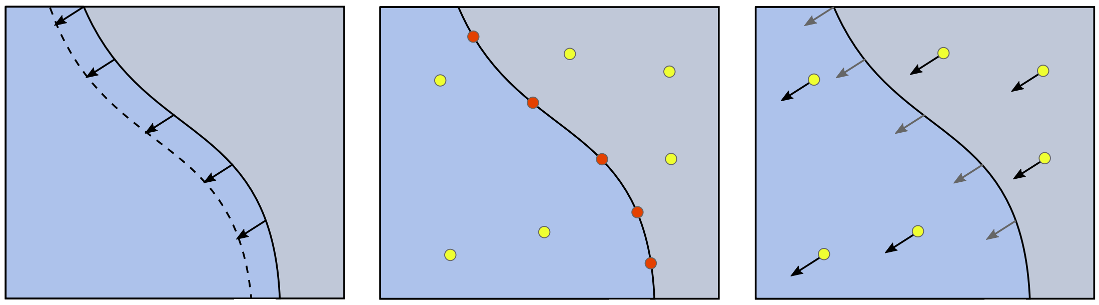


Figure 1.10: Sketch of a discontinuity that moves with some parameter (left), area samples (yellow) and edge samples (orange) when using edge sampling Li et al. [2018], illustration of reparameterization, a technique that does not need to explicitly sample the discontinuity

It's equation

$$\oint_{\partial D} (f \cdot \hat{n}) dx = \iint_{D - \partial D} (\nabla \omega \cdot f) d\omega \tag{1.28}$$

allows us to rewrite the boundary term from Equation 1.27 as

$$\oint_{\partial D} f(\omega, \pi)(\partial_\pi \omega \cdot \hat{n}) d\omega = \iint_{D'} \nabla_\omega \cdot \Big(f(\omega, \pi) V_\pi(\omega) \Big) d\omega. \tag{1.29}$$

$V_\pi(\omega)$, called a *warp field*, is a smooth interpolation of the boundary velocity $\partial_\pi \omega$. For a discontinuous solid texture (i.e. thresholded Perlin noise), constructing the warp field is straightforward and involves only a projection of the boundary velocity onto the desired geometry, see Section 2.4. Using a convolution-based approach allows for the construction of a warp field for discontinuities caused by positional changes of silhouettes of mesh-based geometry. Vicini et al. [2022] show how to obtain a warp field while sphere tracing a signed distance field. The primary application for the latter two examples are in the inverse rendering of shapes.

Not all problems with discontinuities require the accuracy of handling the derivatives as shown here. By convolving the integrand with a small gaussian kernel, parametric discontinuities vanish at the cost of a small bias of the resulting estimator. This is not always possible, e.g. convolving the visibility change introduced by an object's silhouette when differentiating with respect to its position is not meaningful. In our work on inverse rendering procedural wood appearance in Section 2, we apply a similar strategy by differentiating the signal phase of the original periodically discontinuous solid texture in Gabor space, leading to improved convergence of the optimization algorithm[2].

[2]Gabor filters kernels are sinusodials multiplied by a gaussian

Implementing reparameterization into a differentiable renderer that is based on automatic differentiation requires overriding the default gradient calculations. Depending on the concrete implementation of automatic differentiation, this can require substantial changes to the renderer, including changing the core path tracing loop.

1.7 Implementing differentiable renderers

We will now turn to the discussion of implementations of differentiable rendering algorithms, that need to address both the challenge of correctly handling discontinuities, and the challenge of gracefully scaling with the number of differentiable parameters.

Many differentiable, physically-based light transport simulations are available. Redner (Li et al. [2018]), which also implements the edge-sampling approach (Section 1.6.2) to handle parametric discontinuities. Interestingly, Redner can use either TensorFlow or PyTorch as a framework for automatic differentiation, and also has a CUDA backend implementing the derivative computations very efficiently. The Mitsuba3 renderer (Jakob et al. [2022b]) is a successor to the very popular Mitsuba physically based renderer (Jakob [2010]). The system is based on a custom library for automatic differentiation and vectorization with a strong rendering focus, Dr.Jit, and contains an implementation of the reparameterization approach to handling discontinuities based on the divergence theorem. Mitsuba3 and also contains very efficient algorithms to compute derivatives with respect to a large number of scene parameters. TEG (Bangaru et al. [2021]) is a differentiable programming language that contains an integral primitive, making it possible to *analytically* compute derivatives of some integrals with parametric discontinuities, based on automatic elimination of the Dirac terms that emerge within. DiffTaichi (Hu et al. [2019, 2020]) is a language developed to implement differentiable physics simulations that has applications in computer graphics. Beyond physically-based light transport simulations, there exist differentiable rasterizers that are used in machine learning, such as Liu et al. [2019].

In a very basic form, a differentiable renderer can be created by applying automatic differentiation to an unidirectional path tracer (see the algorithm in Figure 1.11 for a simple example). Using an object oriented language that supports operator overloading, this can be realized quite elegantly by implementing a few essential differentiable data types (e.g. float, vectors, matrices and so on) and overloading their operators such that they also compute the gradients with respect to the differentiable parameter(s) π.

The gradient computation can be, for forward-mode automatic differentiation, realized by storing a gradient vector with each differentiable variable, that contains the intermediate derivative values with respect to the differentiable parameters. For large numbers of these parameters, this can quickly become very expensive, both computationally (because we need to perform the derivative-computation once per participating parameter), and in terms of memory use. The latter can sometimes be improved by using a sparse representation of the gradient vector for scenes where only a small subset of all differentiable param-

Algorithm 1 $\partial_\alpha Volpath$

```
 1: $L_e \leftarrow 1.0$
 2: $r_o, r_d \leftarrow castRayThroughPixel()$
 3: $t \leftarrow intersect(r)$
 4: $x_i \leftarrow r_o + t * r_d$
 5: $r_o \leftarrow x_i$
 6: $\tau \leftarrow 1.0$
 7: $\partial_\tau \leftarrow 0.0$
 8: $\partial_\alpha \leftarrow 1.0$                                   ▷ seed
 9: while in medium do
10:     $t \leftarrow sampleDistance(r)$
11:     $t_{max} \leftarrow intersect(r)$
12:     if $t \geq t_{max}$ then               ▷ sampled distance outside medium
13:         $x_{i+1} \leftarrow r_o + t_{max} * r_d$
14:         $\tau \leftarrow \tau * get\tau(x_i, x_{i+1})$
15:         break
16:     else
17:         $x_{i+1} \leftarrow r_o + t * r_d$
18:         $\tau \leftarrow \tau * \alpha$
19:         $\partial_\tau \leftarrow \tau * \partial_\alpha + \partial_\tau * \alpha$          ▷ product rule
20:         $r_o \leftarrow x_{i+1}$
21:         $r_d \leftarrow sampleDirection()$
22:         $x_i \leftarrow x_{i+1}$
23:     end if
24: end while
25: return $\tau * L_e, \partial_\tau$
```

Figure 1.11: Algorithm to estimate radiance derivatives of a homogeneous volume with single scattering albedo α seen from some point outside. The volume density σ_t is hidden inside the *sampleDistance* and *get*τ functions. *sampleDistance* is assumed to perfectly importance sample Beer's law, Equation 1.10. *sampleDirection* samples the phase function f_p for a scattered direction, and is again assumed to perfectly importance sample. These simplifications allow to write the algorithm without explicitly considering the probability density function, because all of the factors in the denominator of the primary estimator cancel out while inside the medium, and only the last path segment requires spectial treatment. *get*τ calculates the transmittance between two points using Beer's law. The volume is further assumed to have a null-interaction at its boundary (so no change of direction due to a change in the index of refraction, for example), the only light source is an environment map that emits radiance $L_e = 1.0$.

eters are actually involved during rendering, depending on what subset of Path Space is being sampled. This allows to exploit the sparsity when involving partially visible or highly undersampled textures, or in grid-based heterogeneous volumes. Our initial work on optimizing PolyJet printouts using differentiable rendering, Sec. 3, was based on this technique.

Reverse-mode automatic differentiation is commonly implemented by recording a computational graph while performing the primal pass through the path tracer. A traversal of the graph in reverse (or any other order) to accumulate derivatives, as discussed in Sec. 1.2, then calculates the gradient. A compile-time implementation of reverse-mode is also possible, Schroeder [2022]. In a physically-based renderer, this graph can become very large, especially in the presence of participating media, which requires simulating hundreds of medium interactions *per path sampled.* Every interaction requires the evaluation of the phase function, recursively estimating in-scattered radiance and estimating transmittance between interactions. Hundreds of these paths need to be traced for an image *per pixel* to converge to a sufficiently small level of variance. In-memory storage of the Wengert tape then is no longer feasible.

1.7.1 Efficient handling of large gradient vectors

To meet the demand for high-performance differentiable physically-based rendering, domain-specific methods have been developed to address the memory-hardness of storing the Wengert tape in large-scale automatic differentiation. Jakob et al. [2022a] write:

“Efficient methods in this area turn the derivative of a simulation into a simulation of the derivative.”

Instead of recording and traversing the tape, the approach of Nimier-David et al. [2020] propagates ”derivative radiation” through a scene, that is emitted by the sensor and eventually absorbed by differentiable scene parameters. The rendering is split into two passes, a primal and an adjoint one, that share the same basic propagation algorithms, and both passes being linear in space complexity. This adjoint technique makes calculating derivatives with respect to millions of scene parameters possible, as is required for the optimization of grid-based textures. Evaluating the adjoint pass requires, however, recursive queries for in-scattered radiance at each scattering event, leading to quadratic time complexity. Remarkably *Path replay backpropagation*, proposed by Vicini et al. [2021], reduces this to a linear.

A note on the interpolation of textures Some material models in computer graphics are based on storing some tabulated data as a bitmap texture, such as a displacement map that models small surface height variations. Rendering these requires evaluating the spatial derivative of these displacement to calculate a local normal vector used for lighting. Using linear interpolation to lookup texture values then results in piece-wise constant derivatives (and as such, piece-wise constant normal vectors), and the delivery in unsatisfactory rendered results as a consequence. Higher-order texture interpolation needs to be used in that case.

1.7.2 Testing derivatives calculated using Monte Carlo estimation

In the process of implementing a differentiable rendering algorithm, the validation of the calculated derivative values is sometimes necessary. Comparing derivative estimates to their finite-differences approximations is a natural choice, but these also come with their own variance. This translates to the question of what value for ε to use, Equation 1.2.

Too large values for ε lead to significant bias error. Too small values result in machine round-off errors and lose any meaning if they get too small compared to the variance of the primal estimator. Using a large number of samples to compute estimates is essential to increase the coverage level of the confidence intervals, which in turn allows for the use of smaller values of ε. Formally, the variance $\mathbb{V}$ of an derivative estimator $\widehat{Y}$ that uses the primary estimator $f \approx \widehat{X}^{(i)}$ and uses finite differences,

$$\begin{aligned}\widehat{Y} &= \frac{f(x-\varepsilon)+f(x+\varepsilon)}{2\varepsilon} \approx \frac{\partial f}{\partial x} \\ &= \frac{1}{2N\varepsilon}\left(\sum_{i=1}^{N}(\widehat{X}^{(i)}(x+\varepsilon)-\widehat{X}^{(i)}(x-\varepsilon))\right)\end{aligned} \tag{1.30}$$

is

$$\begin{aligned}\mathbb{V}[\widehat{Y}] =&\left(\frac{1}{2N\varepsilon}\right)^2\sum_{i=1}^{N}(\mathbb{V}[\widehat{X}(x+\varepsilon]+\mathbb{V}[\widehat{X}(x-\varepsilon]) \\ =&\frac{1}{2N\varepsilon^2}\mathbb{V}[f],\end{aligned} \tag{1.31}$$

if $\widehat{X}^{(i)}(x-\varepsilon)$ and $\widehat{X}^{(i)}(x+\varepsilon)$ are independent random variables (different random seeds used for the sampler for each of the primal renderings f). The variance thus gets very large for $\varepsilon \ll 1$. It is much better to use the same set of random inputs for the sampling process (same seed). If $\widehat{X}^{(i)}(x)$ is differentiable with respect to x, we can use the relation

$$\widehat{X}^{(i)}(x+\varepsilon)-\widehat{X}^{(i)}(x-\varepsilon) \approx 2\varepsilon\frac{\partial \widehat{X}^{(i)}}{\partial x} \tag{1.32}$$

and thus

$$\mathbb{V}[\widehat{V}] \approx \frac{1}{N}\mathbb{V}[\frac{\partial \widehat{X}}{\partial x}]. \tag{1.33}$$

For some functions, an interesting technique is fitting a suitable model (i.e. using polynomial regression) to a sampling of the parameter domain of x. This gives access to an approximate analytical proxy function that can be differentiated symbolically and used for derivative validation.

1.8 Inverse rendering as an optimization problem

Inverse rendering aims at recovering scene parameters from one or multiple views of the target scene appearance. We frame inverse rendering as an optimization problem, where differentiable rendering and a differentiable material model will supply the gradients needed for the optimization to work on. Because the differentiable renderer is predictive in the sense that its outputs closely match physical reality, we can use actual photos as an input to the inverse rendering problem, bridging the gap from physical reality to a *digital twin.* The process also works in the opposite direction, by using the recovered material parameters as instructions for a fabrication device to create physical objects.

Formally, the optimization problem is given by the minimization of some loss function E between one (or several) reference images I_{ref} and *renderings* that are obtained by the (predictive) rendering function R.

$$\underset{\pi}{\operatorname{argmin}}\ E(R(\zeta(\pi)), I_{ref}) \tag{1.34}$$

The rendering operator R takes the scene ζ and its (differentiable) parameters π to an image $I \in \mathbb{R}^{W \times H}$ with a resolution of $W \times H$ pixels. Using some initialization π_0 and given the differentiability of all functions involved, the minimization problem can be solved using the gradient descent algorithm with the learning rate β:

$$\pi_{n+1} = \pi_n - \beta \frac{\partial E}{\partial \pi} \tag{1.35}$$

Using the chain rule, we can string together the required derivative $\partial_\pi E$ from the derivative of the rendering operator and the derivative of the scene, which in turn contains the material model we are interested in,

$$\frac{\partial E}{\partial \pi} = \frac{\partial E}{\partial R} \frac{\partial\ R}{\partial \zeta} \frac{\partial \zeta}{\partial \pi}, \tag{1.36}$$

with the second and third factor, $\partial_\pi R$, being supplied by the differentiable rendering system that implements the differentiable material model.

1.8.1 Importance of initialization

Finding a good initialization π_0 can be (but does not always have to be) very important. The solution surface the optimization is being performed on is often non-convex, containing many local minima that can have hugely different loss values. Depending on the initialization, the algorithm might converge to different solutions. Finding the global minimum thus can be very difficult, especially in the presence of many parameters, but this is not always necessary.

As a simple example, consider a scene as shown in Figure 1.12, where a light source is observed through two translucent slabs that have different absorption coefficients that we are interested in, and identical (and small) scattering coefficients that are fixed. The observed color is invariant of the stacking order of the slabs, so there are at least two different solutions for the free parameters that result in an identical appearance. This is called a non-local ambiguity in volumetric light transport, that plays an important role in the context of full-color 3D printing (See Part II, Sec. 3, also Babaei et al. [2017]).

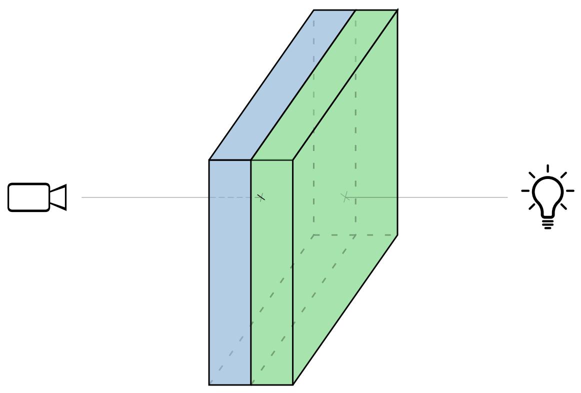


Figure 1.12: Sketch of a light source observed through two translucent slabs, illustrating a non-local ambiguity in volumetric light transport

It can be helpful for the convergence of the optimization to transform the problem into a space that has more favorable properties. We show examples of such a transformation in the contributions sections of this thesis: For the optimization of polyjet printouts, we optimize in *material mixture space*, instead of the space of physical parameters of heterogeneous volumetric light transport. For appearance-matching a procedural solid wood texture, we fit coarse structure by optimizing in Gabor space for it gives better convergence due to the separation of the ring structure from the rings color variations.

1.8.2 Optimization algorithms

Choosing a suitable optimization algorithm and its hyper-parameters can also have a big impact on the convergence. Some parameters can require different learning rates than others, due to having different loss gradient magnitudes. Further, some of the parameters can be (non-)linearly dependent, creating ambiguity that can negatively affect convergence behavior. In order to effectively navigate such spaces, the machine-learning community has spawned many optimization algorithms with favorable properties in such cases, with Adam (Kingma and Ba [2014]) being the most prominent example.

Due to the variance of gradients obtained from a Monte Carlo estimator, gradient-descent optimization is effectively turned into stochastic gradient descent, Robbins and Monro [1951]. In inverse heterogeneous scattering, the inverse rendering algorithm *literally* stochastically considers a subset of the differentiable parameters during each iteration due to sparse sampling of the participating medium. As long as the gradient estimates are *unbiased*, the convergence guarantees that exist for gradient descent still apply analogously. This creates a tradeoff between iteration time and the number of iterations required for convergance, adjustable by the sample count of the estimator. Eventually, the variance of the estimator will prohibit the optimization to progress further and instead will show a trajectory that jumps around the actual local minimum. Increasing the sample count allows the optimizer to get closer. Adjusting along this tradeoff on the fly during an optimization run can lead to improvements of the overall optimization time, e.g. starting with coarse, low-spp estimates in early iterations, and subsequently switching to higher sample counts. Denoising the primal rendering can used remove some of the variance of the calculated loss values at the cost of some bias. Denoising gradient estimates themselves seems to be an interesting direction for further work.

Parameter ranges need to be constrained to meaningful values so the optimization trajectory does not drift beyond the feasible set. Methods such as the use of Lagrangian multipliers (Bertsekas [2014]) are well known. Other approaches use sigmoids to discourage certain parameter trajectories. Projected gradient descent (Gafni and Bertsekas [1984]) is another standard optimization algorithm developed to improve convergence under some constraints. There, after each gradient descent step, the resulting parameters are projected back onto the feasible set.

1.8.3 Solving inverse volumetric light transport

For simple cases, analytical solutions to inverse volumetric light transport are available, using one of several approaches. The application of Beer's law (Equation 1.10), which is critically limited to non-scattering media; Kubelka-Munk theory, Kubelka [1948], which is still widely used for modeling textiles and paints. There are also methods that replace the radiative transfer equation with the diffusion equation and solve it instead, e.g. d'Eon and Irving [2011], Haskell et al. [1994], Lastly, the adding-doubling method by Prahl et al. [1993] can be used in simplified cases with planar geometry to completely describe radiative transfer.

Even for an homogenous medium of arbitrary geometry, the error surface of an gradient-based approach to the inverse rendering problem aiming at recovering the parameters (σ_s, σ_t) is smooth Gkioulekas et al. [2013]. However, it contains ambiguities for media with non-isotropic phase functions f_p. Similarity theory Wyman et al. [1989] studies these equivalence classes of the parameters of the radiative transfer. While this can be exploited to improve rendering times, Zhao et al. [2014], it can lead to convergence difficulties in the inverse rendering case.

Under the assumption of a band-limited solution radiance L, similarity theory states that there exists alternate medium parameters $(\sigma_t^*, \sigma_s^*, f_p^*)$, such that the corresponding radiative transfer equation

$$(\omega \cdot \nabla)L(\omega) = -\sigma_t^* L(x,\omega) + \sigma_s^* \int_{\mathbb{S}^2} f_p^*(x,\omega,\omega')L(x,\omega')d\omega' \tag{1.37}$$

has a solution equal to the original one in Equation 1.12. Finding such equivalent parameters is not trivial. Some approaches are based on a substitution of radiance L and phase function f_p using spherical harmonics, using either first order approximations or approximations of higher order, as shown by Zhao et al. [2014].

In the heterogeneous case, non-local ambiguities, like the translucent slabs example described above, add to the non-convexity of the solution space (Gkioulekas et al. [2016]). To disambiguation of the parameters, additional views under different lighting conditions can be used as an input to the inverse rendering algorithm. In some cases, however, this disambiguation it is not necessary: In appearance fabrication, finding *some* material parameter instance that satisfies the requirements to the output quality is sufficient. This is discussed in more detail in Section 3.

1.9 Appearance models

Within the framework of the rendering equation, material appearance is described using either the BSDF (see Section 1.3.2), or in the presence of a participating medium, in its respective volumetric properties. The most obvious property, the surface color, is essentially created by varying the BSDF throughput with wavelength, to simulate the absorption that is at the root of subtractive color mixing. Other visual aspects such as gloss or surface roughness are the result of varying the reflectance distribution with the incident angle.

Besides straight-up measuring reflectance distributions using e.g. a photo goniometer and using this tabulated data for image formation in the renderer, approximative models are available. Simple empirical models are usually based on a diffuse component and a specular lobe, the latter of varying width to mimic changing levels of gloss. The more advanced, physically based microfacet BSDFs, Cook and Torrance [1982], models the material surface as a set of very small ideal mirrors (“facets”) and calculate aggregate properties from their orientation statistics. For layered surfaces, approximative BSDFs models can describe i.e. the effect of clear coat, see for example Hanrahan and Krueger [1993]. Production BSDF models (i.e. Disney Principled BRDF by Burley and Studios [2012]) then composite several distinct distributions capturing different appearance aspects (diffuse and specular reflectance, roughness, metallicity, clear coat, sheen) into an aggregate function.

Representations of the underlying data of these functions, we call them appearance models here, can roughly be categorized into four families:

- *texture-based appearance* stores spatially varying material parameters in memory as textures
- *exemplar-based* approaches can synthesize these textures from one or several small examples
- *procedural appearance models* generate this data on the fly while rendering, often by the means of a material graph
- *data-driven appearance models* use neural networks for representation.

Ways of modeling volumetric appearance follow along similar lines. The volume properties are either stored in 3D-textures (grids), procedurally computed on the fly on the basis of some noise function, or encoded in some neural representation.

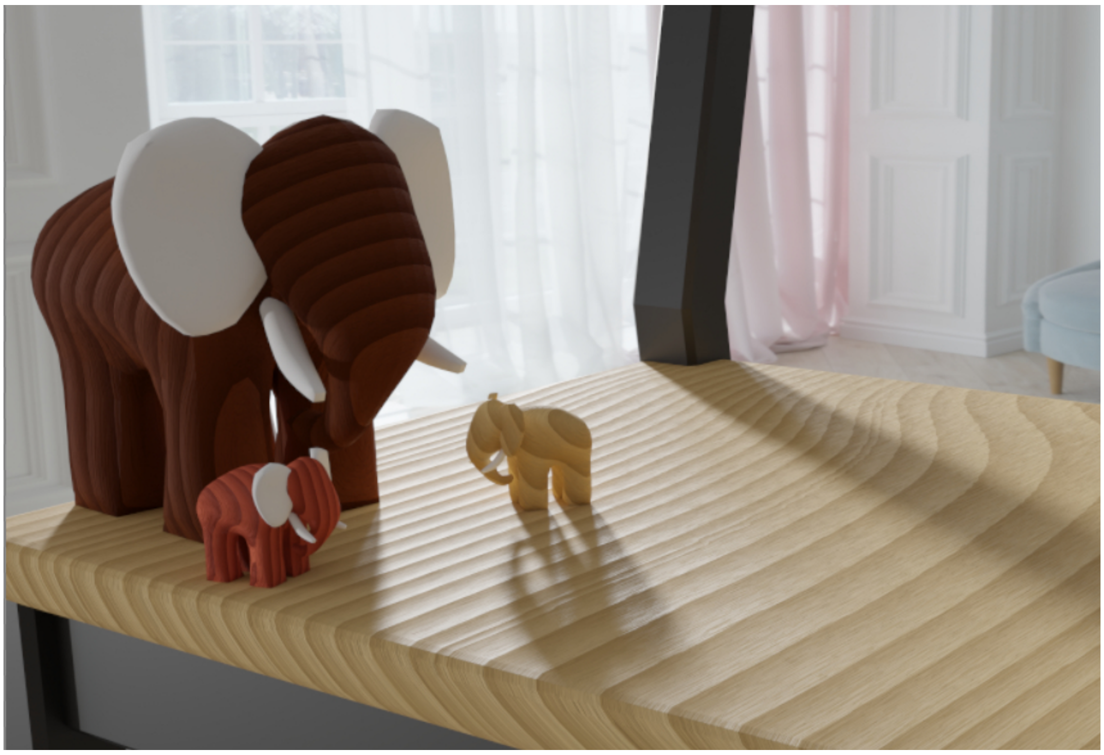


Figure 1.13: Wood rendered using a procedural appearance model, showing varying levels of surface roughness and gloss, that are a result of the underlying volumetric geometry

Texture-based surface appearance can capture most materials and reach very high fidelity if the texture's resolutions are large enough. The textures are created by measurements, such as photography for the case of the diffuse albedo. Other BSDF components can be acquired using commercial-grade scanning hardware that is available to perform these at great detail, and often at great cost. Different implementations of the acquisition process have been proposed (Guarnera et al. [2016]), such as single-image SVBRDF capturing (Deschaintre et al. [2018]).

A major disadvantage of these methods is the memory requirement for storing the textures involved - depending on viewing distance and object size, they can require gigabytes of data to capture the appearance of a single object. In production work, measurements of all BSDF components are sometimes not available, and modeling starts from only a diffuse albedo texture. This leaves the creation of other BSDF parameters to the hands of an artist. A step towards automating this process is the fitting of a *procedural appearance proxy*, see Section 2 for our contribution in this respect.

Data-driven approaches imply an appearance matching process due to their fundamental approach: The neural networks weights are trained on examples of the desired appearance, such as a face, until some convergence criterion is reached. Gradient-based optimization is used to train the network, driven by some loss function. Neural volumes have been a very successful approach here, see Mildenhall et al. [2021], Dellaert and Yen-Chen [2021]. *Procedural appearance models* have seen increasing adaptation in production use. They are usually not limited

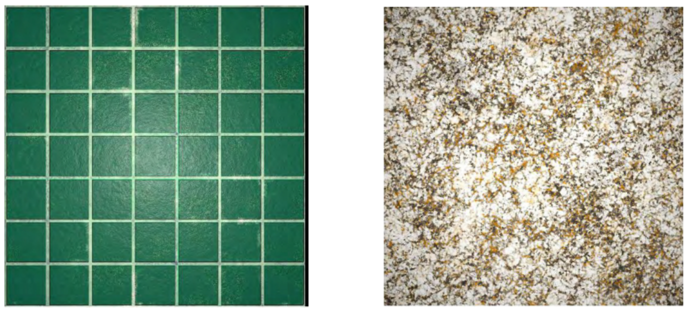


Figure 1.14: The tiled floor material on the left is dominated by the regular grid-like structure, requiring a custom material graph. Stone on the right shows dominantly stochastic appearance, making it a good candidate to model using procedural noise. Images from Shi et al. [2020]

in image resolution, and can, depending on the material graph, express a substantial range of material appearances (Figures 1.13, 1.14). Another clear advantage is the usually negligible storage requirement, which is offset by an increased per-pixel computational cost to evaluate the procedural model. Their bottom-up synthesis approach is based on performing transformations and filtering, often on the basis of elementary noise functions, in a computational graph. This is used for both highly-structured surfaces such as a tiled floor, and unstructured ones that can best be described by its statistical properties.

1.10 Matching procedural models

Procedural appearance models are usually created by a skilled artist. The manual process is quite time-consuming process, requiring many hours of work – both for the creation of the basic material graph, and its adaptation to a specific target appearance. Automating the entire process of finding a good procedural representation of a target appearance is extremely challenging, because it includes the design of a suitable material graph.

This can be tackled in different ways. One way is using the same, general model to render all objects with, and finding appropriate parameters to match the desired appearance. Such a model needs to have a large appearance gamut in order to be universally applicable. Recent work in Neural Radiance Fields Mildenhall et al. [2021], and in solid noise functions (e.g. Galerne et al. [2012]) are two examples of that. Appearance-matching of strictly noise-based procedural appearance boils down to matching its statistical properties, that can be obtained by spectral analysis in their respective bases. Lagae et al. [2010], for example, use a multi-octave wavelet noise and match using principal component analysis and histogram analysis. For Gabor noise, Galerne et al. [2012] show a method for approximating a Gabor noise power spectrum into a sparse sum of Gaussians.

Highly structured materials, (Figure 1.14, left) cannot be approximated well using noise functions. Instead, they require a specific procedural model that is then tied to a certain material class – a function to model a tiled floor would hardly be convincing if being applied to a wooden object. Since the automation of the design of a material graph from scratch is not yet possible, existing approaches instead build on top of a library of material graphs and reduce the design problem to a selection problem. This has been explored by Shi et al. [2020]. If the object's material class is known as a prior, the model can be selected manually.

Given an appropriate material model, an instance of its parameters has to be found such that its rendering closely matches the desired appearance. This begs the question of how to quantitatively measure an appearance difference, a subproblem that we explore further in Sections 3 and 2. With a suitable difference metric available, numerical optimization can be used to find the parameter set in question.

Guo et al. [2020] optimize model parameters under a Bayesian framework. They obtain differentiable models by implementing them using a automatic differentiation framework (PyTorch), and use a loss function based on VGG feature maps (Style loss, Gatys et al. [2015, 2016]). Shi et al. [2020] show a similar approach.

The use of a loss function that considers non-local features for polishing the model parameters is very important here. This is because the selected procedural model may not be able to create a pixel-perfect replica of the target appearance, so using a per-pixel loss function may lead to unsatisfactory solutions. Metrics that consider structural similarity, like the one proposed by Wang et al. [2004], can improve on that, by also considering a small neighborhood around each pixel. Style loss (e.g. Jing et al. [2019]) can connect features that are spatially even more seperated, due to the multiple levels of convolution of a VGG network that are used to calculate these perceptual loss values.

For discrete problems (integer constraints), using gradient descent is not feasible. Solving these combinational inverse design problems is still tractable in some cases, e.g. Ansari et al. [2020, 2022].

1.11 Wood anatomy and optical properties

Wood, as a product of nature, was one of the first building materials known to man. It is highly sought after for its decorative properties that show great variation through the different wood species and even across members of the same species. Processes such as physical surface abrasion through sanding or planing, treatment with compounds such as oil, wax, resins and paint, and more recently chemical alterations of the different compounds wood is made from add to the appearance gamut the material can exhibit.

In computer-graphics, wood surfaces are often created by mapping a texture onto a shape that consists of the diffuse reflectance, and can supply other parameters to complete an approximate description of its reflectance properties. These are commonly derived from the diffuse reflectance by a manual process that is performed by a texture artist. Sometimes, BSDF inputs are acquired by material scanning, which gives accurate information but comes with high demands for storing it. Some procedural, ground-up appearance models are also available that are inspired by the underlying wood anatomy, Liu et al. [2016], Larsson et al. [2022]. The fully automatic matching of a procedural wood appearance model is still an unsolved problem.

If wood grows in an area with pronounced yearly seasons, it shows charac-

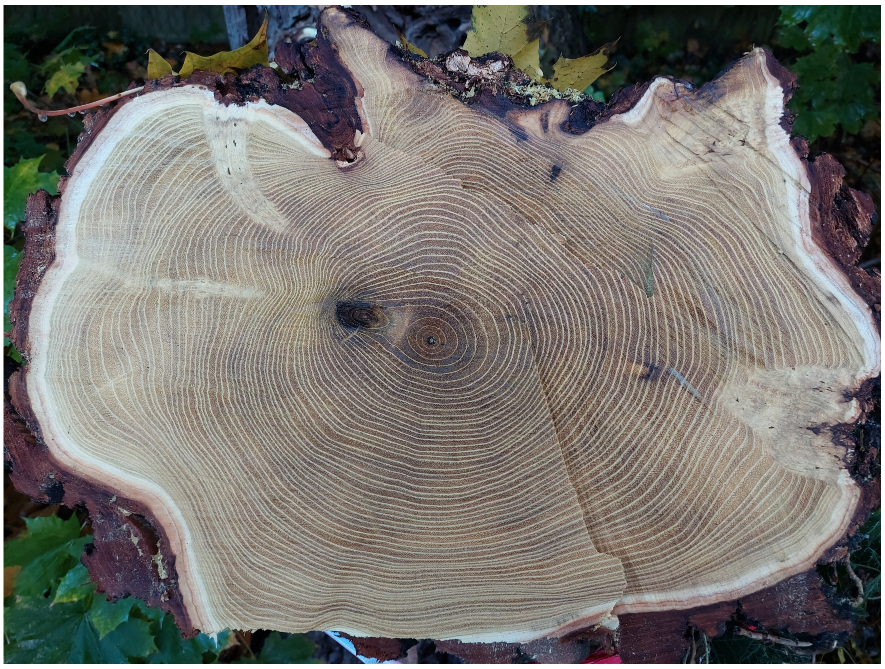


Figure 1.15: Transverse surface of a willow stem. Note the characteristic yearly growth rings, and the double core due to a side-branch starting nearby

teristic, quasi-cylindrical growth zones called growth rings. This is in contrast to trees from geographic regions where the yearly weather variations are small (i.e. the tropics). Their wood can still show phases of faster and slower growth, but they do not correlate with the age of the tree. In temperate climate zones, growth in spring is characterized by large cell lumina [3] and thin cell walls of the tracheids. Later in the year, usually after the summer solstice, growth changes to thicker walls, smaller lumina that is darker and contains more lignin. Wood formed in this second phase is called latewood, see Figure 1.16 for a qualitative overview.

Variations of these growth patterns can be caused by a multitude of environmental factors, including the availability of water and sunlight due to the weather or other locally changing conditions (such as another tree shadowing the specimen due to its faster growth). Dendrochronology is the science of dating the growth rings based on these variations, and vice versa. It allows to assign exact calendar years even in some wood exemplars that have been found in archeological sites, and can also be used to derive climate conditions from the growth variations. The latter is used as primary data in climate research.

From a functional perspective, early growth secures optimal transport of water and nutrients from the root to the treetop; late growth contributes more to

[3]Lumen is the biological name for the inside of a cell

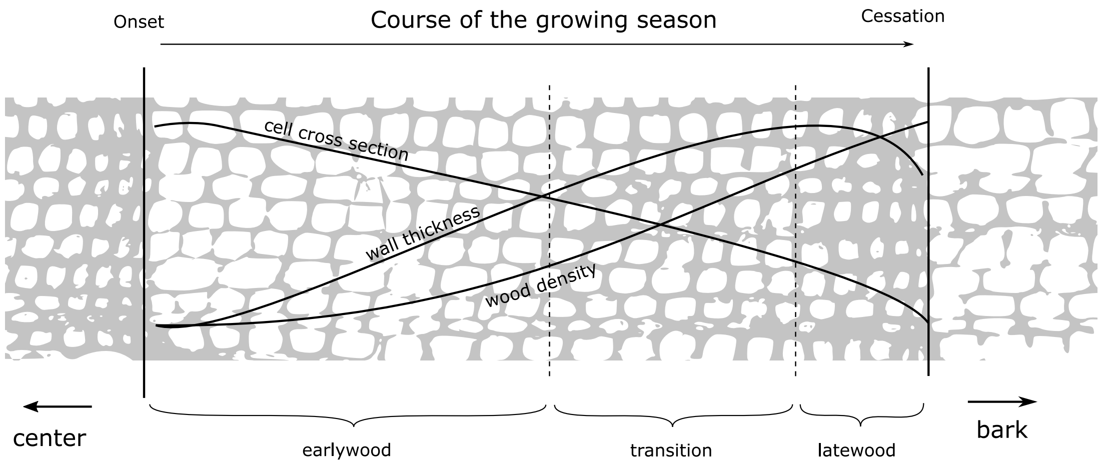


Figure 1.16: Coniferous species' ring patterns are created by environmentally caused variations in cell diameter and wall thickness; Redrawn after Figure 1 in Cuny et al. [2014]

the static strength of the plant. The material between two cells, called the middle lamella, can be characterized as a homogeneous, isotropic gel, consisting of hemicelluloses, pectine and later lignin, Hänsel [2014]. The cell walls themselves consist of so called fibrils that are arranged in a helical fashion around the vertical axis of the tracheid's cell volume. Fibrils are some ten nanometers in diameter, and made of bundles of cellulose molecules. In essence, wood could be described as a naturally occurring, fiber reinforced polymer.

Optical properties Wood displays a wide range of optical features, and we already mentioned the characteristic patterns formed from the growth variations. The surface of wood has spatially varying reflectance, latewood usually being more glossy than earlywood. The surface reflectance is anisotropic, depending mostly on the local fiber orientation. Some species of wood show fluorescence.

Wave-optical effects can be observed when considering transparent wood. Transparent wood is created by a chemical process called de-lignification, where lignin is removed from between the cell walls, and replaced with an artificial, transparent polymer. This leaves the cell walls and the fibrils they consist of mostly intact, creating a highly anisotropic scattering medium with structure-sizes in the order of magnitude of the wavelength of light.

Kienle et al. [2008] performed both spatially- and time-resolved measurements in the near-infrared frequency range, and verified their results using Monte Carlo simulations. For the simulation, they model wood as a grid of infinitely long, hollow, parallel cylinders with a homogeneous scattering medium in between. They established the anisotropic nature of wood reflectance and its dependence on moisture content, and most importantly by their simulations that wood's interaction with light can mostly be explained by the cylinder grid structure the tracheids form. It seems reasonable to assume that at least qualitatively, their results can be transferred to the visible range.

These results were later confirmed by Kitamura et al. [2016], who also performed time-of-flight near infrared spectroscopy measurements in order to determine the optical absorption- and scattering coefficient of wood *cell wall substance*. They modeled the medium as a regular grid of tubes with a square cross-section, with dimensions based on microscopic measurements of the specimen they were recreating.

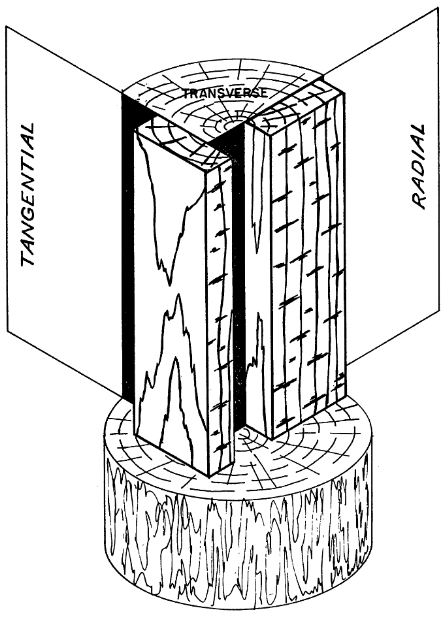


Figure 1.17: "Diagrammatic log section that illustrates the relationship of tangential (t), radial (r), and transverse or cross (x) surfaces. Note patterns and growth- ring orientation on different surfaces.", Beals and Davis [1977], Figure 2. Radial cuts go through the center of the stem, tangential cuts are offset

Sugimoto et al. [2018] performed spatially and spectrally resolved measurements in the optical frequency range. They measured transmittance and reflectance with different fiber orientations. This data is sufficient to plug in to one of the analytical approximations of the radiative transfer equation. We used the inverse adding doubling method with the values shown in Table 1.1.

	radiant intensity			
	reflected	transmitted	σ_a $[mm^{-1}]$	σ_s $[mm^{-1}]$
tangential direction	0.62	0.2	0.0842	16.2
transverse direction	0.38	0.5	0.0573	3.23

Table 1.1: Estimate for the volumetric properties of coniferous wood from data reported by Sugimoto et al. [2018]. The index of refraction was assumed to be $IOR = 1.55$, the sample thickness $d = 0.50mm$, the light beam diameter $d = 1.0mm$. Figure 1.17 for an additional overview

To render wood volumetrically, we assume the optical density $\sigma_t = \sigma_a + \sigma_s$ to be constant with varying wavelength and instead calculate a spectrally varying single-scattering albedo. To reflect density variations due to different cell wall thicknesses, we vary the density in the range of $\sigma_t \in [3.3, 16.21]mm^{-1}$. This rough estimate completely ignores wood's anisotropy and can only serve as a first step towards physically-based volumetric rendering of wood.

The apparent surface albedo of a participating medium does not map trivially to the single-scattering albedo α. This was studied extensively by Elek et al. [2017], who analyzed both the forward and inverse mapping using a sampling of the parameter space and successively regressing a polynomial to find an analytic approximation. They map the single-scattering albedo α to a apparent surface color C using the following polynomial:

$$\begin{aligned}\alpha(C) =& \sum_{k=1}^{K} c_k \left(1 - (1 - \frac{C - C_s}{1 - C_s})^{d_k}\right) \\ c_k =& \{0.163581, 0.391943, 0.029277, 0.316847, 0.098352\} \\ d_k =& \{3.969542, 15.94272, 46.26871, 59.95706, 206.5716\} \\ C_s =& 0.04526\end{aligned} \tag{1.38}$$

We use this mapping to find approximate values for the single-scattering albedo of our wood volumes from their median surface reflectance, resolved in RGB space.

1.12 Color 3D Printing

Much like computer numerically controlled machining revolutionized manufacturing, 3D printing is making its way towards transforming how we manufacture physical objects. While dimensional accuracy, and more recently also other mechanical properties, were the main goal of creating and improving the available 3D printing technologies, additive *appearance* is slowly emerging as a new technique to creating objects with the desired optical properties. The most promising printing technology to control objects' visual properties is PolyJet 3D printing, or simply "UV Printing").

The Polyjet technology works by depositing drops of liquid, colored photopolymers that can be rapidly hardened by applying UV light. This allows for the building of solid, three dimensional structures. Figure 1.18 shows some example objects. Due to the printer's ability to switch between several different materials, on a per-droplet basis, it is possible to control the printout in very fine detail. This essentially allows for the printing of heterogeneous scattering volumes, with a typical voxel-size of about 15 µm, see Figure 1.19 for a sketch of the operating principle.

Controlling the surface appearance of such a printout is difficult, due to the non-trivial light transport happening inside the volume that can lead to washed-out colors, blurring of sharp edges. Being aware of the problem, printer manufacturers made a big range of materials available to improve on the appearance accuracy – primary (CMYWK) colored materials are available with different optical densities. Low-scattering inks can be used to expand on the color-gamut of the printout, while sacrificing texture sharpness. Highly dense/opaque materials do the opposite, reducing color-bleed while limiting the color gamut.

The gamut of printable appearances also depends on the geometry of the printed items. Thin features reduce the attainable opaqueness, and can intro-

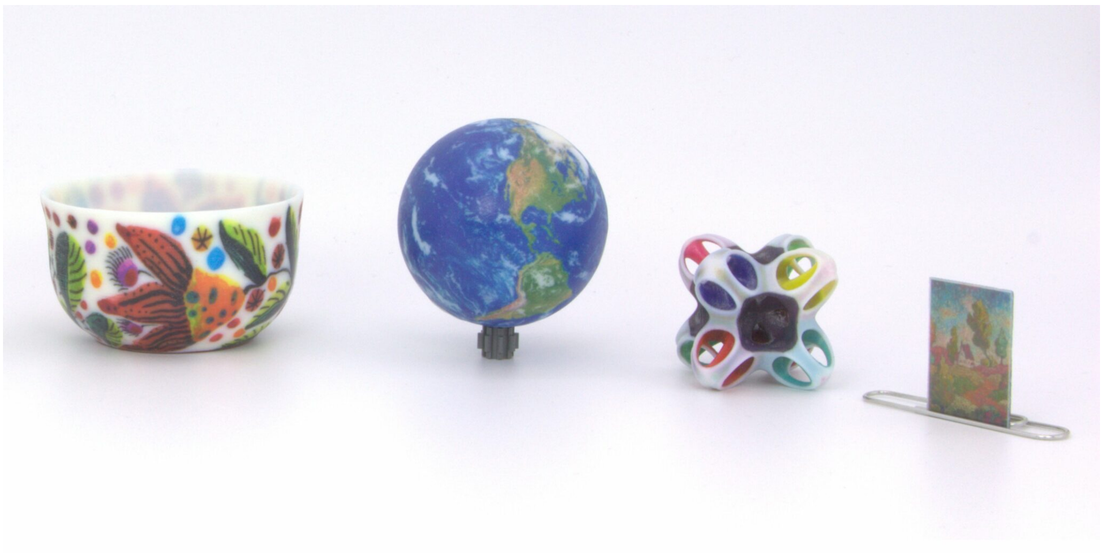


Figure 1.18: Photograph of some Polyjet printouts. The diameter of the globe is 6cm. Image by Sumin et al. [2019]

duce geometric cross-talk. This is highly dependent on the appearance target – a thin slab with similar colors on the opposing sides is less problematic than the same geometry with one white and one black side. On arbitrary geometry, the achievable appearance gamut is a function of local curvature and material thickness. Several publications have been addressing these issues (Elek et al. [2017], Sumin et al. [2019]). In Part II, Section 3 we contribute a novel, general approach to 3D print appearance optimization based on a differentiable material model for the printed volumes.

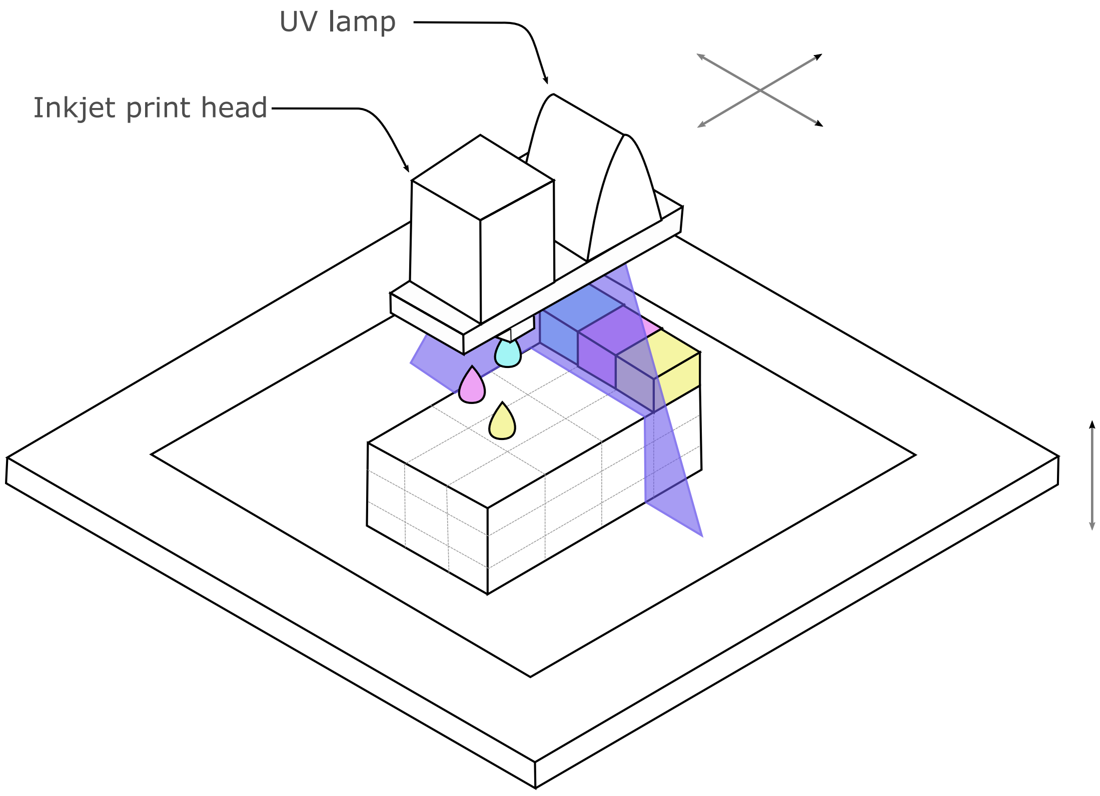


Figure 1.19: Sketch of the principle of a Polyjet UV printer

2. Contribution I: Appearance matching

2.1 Summary

Procedural wood appearance allows for beautiful renderings of a material we all are very familiar with. Liu et al. [2016] proposed the go-to model here, which has a large expressive range that covers many wood species. In their closing remarks, they propose to solve the hard problem of automatically matching their models parameters to a concrete wood specimen automatically. This was the motivation to develop the solution herein.

In production, wood is still mostly modeled using textures acquired by material scanning, with the availability of textures for spatially varying roughness and normal maps being the exception rather than the rule. Using a procedural proxy for the diffuse surface albedo that is *anatomically meaningful* allows for the automatic generation of normal- and roughness-information from a plain photo, and also gives a considerable amount of meaningful artistic control.

The main challenge for us was recovering a plausible map of the wood fibre orientation and the classification of pixels into the earlywood-latewood gamut, to also enable the use of anisotrophic material models. This requires to orient the specimen within the tree stem to get an idea of the sub surface ring structure. We solve this by identifying growth rings on the specimen's surface, and reasoning about the parameters of the cylindrical coarse structure that could have created them. Our method is based on Gabor space, both for ring detection and for recovering a smooth deformation field that allows to accurately match the observed pattern.

Our approach not only gives rise to automatically generating BSDF parameters, but also enables volumetric renderings of thin wood veneer, where the coarse-structures and fibre orientations hafe a significant impact on the rendered appearance. We also found interesting connections to the field of dendrochronology, where the accurate automatic identification of growth ring boundaries is an active research topic.

EUROGRAPHICS 2023 / K. Myszkowski and M. Nießner
(Guest Editors)

Volume 42 (2023), Number 2

Automatic inference of a anatomically meaningful solid wood texture from a single photograph

Thomas K. Nindel[†1,2], Mohcen Hafidi[1], Tomáš Iser[1], and Alexander Wilkie[1]

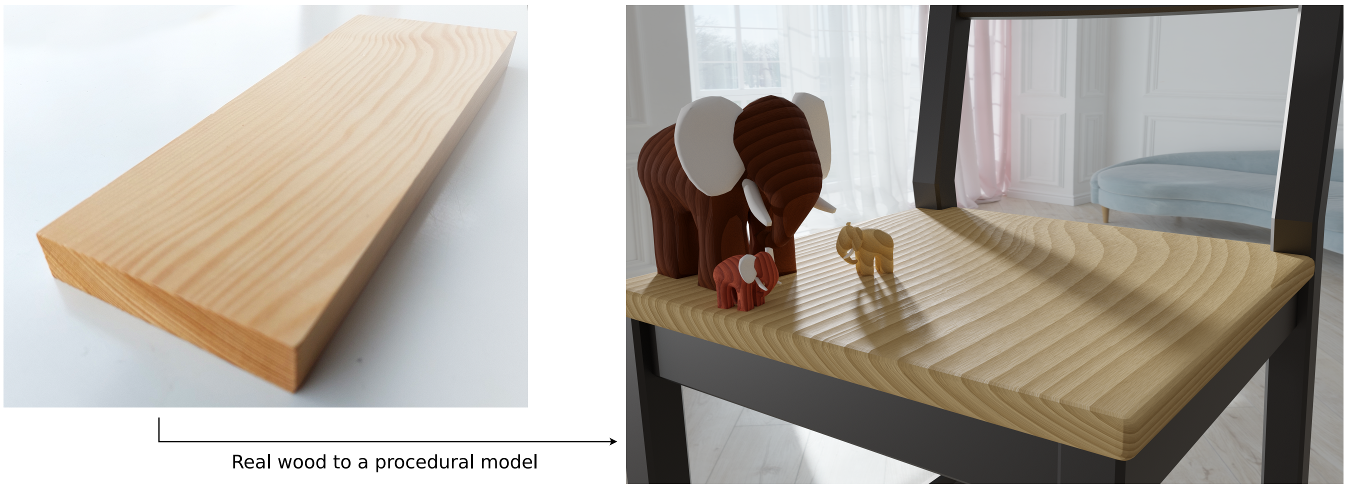


Figure 1: *Our method is capable of taking a 2D scan of a real wood specimen (left, photograph) and then automatically matching corresponding parameters for a procedural solid wood appearance model based on [LDHM16]. Such a model is fully three-dimensional and allows realistic rendering of carved wooden solids (right, Monte Carlo simulated render) based on the input photographs.*

Abstract

Wood is a volumetric material with a very large appearance gamut that is further enlarged by numerous finishing techniques. Computer graphics has made considerable progress in creating sophisticated and flexible appearance models that allow convincing renderings of wooden materials. However, these do not yet allow fully automatic appearance matching to a concrete exemplar piece of wood, and have to be fine-tuned by hand. More general appearance matching strategies are incapable of reconstructing anatomically meaningful volumetric information. This is essential for applications where the internal structure of wood is significant, such as non-planar furniture parts machined from a solid block of wood, translucent appearance of thin wooden layers, or in the field of dendrochronology.

In this paper, we provide the two key ingredients for automatic matching of a procedural wood appearance model to exemplar photographs: a good initialization, built on detecting and modelling the ring structure, and a phase-based loss function that allows to accurately recover growth ring deformations and gives anatomically meaningful results. Our ring-detection technique is based on curved Gabor filters, and robustly works for a considerable range of wood types.

CCS Concepts

• ***Computing methodologies*** → ***Reflectance modeling; Image processing;*** • ***Applied computing*** → *Environmental sciences;*

1. Introduction

In *wood rendering* and *wood appearance modeling*, the current state of the art is based either on highly precise BSSRDF measurements (BTFs, appearance scans), or procedural models.

[†] Corresponding author

Measurement-based approaches yield pixel-perfect matches at the cost of high storage and memory-bandwidth requirements, acquisition complexity, and inability to edit the data after acquisition. On the other hand, procedural wood models allow artistic control and editing, but they are difficult to match to given wood samples, both if an exact match to a given piece of wood is needed, or only with regard to a general wood type. Recent work on using optimisation to match procedural material models to observations [SLH*20] has been quite successful in a broad range of settings, including wood. However, as figure 2 shows, current approaches still fail to properly match a given wood sample *down to its internal 3D structure* – a feature that is needed to make wood grain wrap correctly around a solid 3D object. Specifically, as figure 2 shows, extant techniques are capable of generating an internal ring structure - but not one that really convincingly matches the wood grain pattern seen on the top surface, like in the results we show for our technique in figure 8.

Our main contribution, described in section 3, is that we propose a robust, deterministic method for the automatic inference of a locally fit, procedural 3D material model for wood. Our approach allows one to obtain realistic solid wood textures that can be carved to any geometry (such as the elephants in Fig. 1). Our approach is not based on machine learning and does not rely on any training datasets.

Our work builds on the assumption that by accurately identifying the basic ring structure and its deformations, we can satisfy not only the requirements from wood rendering, that is to attain closely matching structural appearance (Fig. 3, bottom left), but also the requirements of *dendrochronology* for a precisely aligned identification of ring boundaries (Fig. 3, bottom right). In the dendrochronology field (section 2.3), core samples of living trees are inspected for their growth rings variations, and the resulting data is interesting for climate research, archaeology, and art history. Automatic approaches have to be robust to the wide range of wood anatomical features, as certain wood species (e.g., diffuse-porous hardwoods) are very difficult to work with. Our approach performs well even in these settings, and in fact, we used it in dendrochronological contexts as a means of verifying its performance.

2. Related work

2.1. Modelling wood appearance

Wood is a complex material with an intricate volumetric structure. Hartmann et al. [HKRFM17] dive into the complex process that controls the formation of wood cells, whose variance is ultimately responsible for the emergence of the characteristic growth patterns. From a technical standpoint, wood is a bio-composite formed from three different kinds of molecules: cellulose, hemicellulose, and lignin. Lignin is the main chemical responsible for the wood color. It is preferentially found in areas of thick cell walls [LVS*18], which explains why late growth shows a higher color saturation than early growth in coniferous species. Chemically replacing lignin with a transparent polymer results in transparent wood, a material with interesting optical properties [VCL*18].

The basic approaches for modelling wood in photorealistic rendering can be divided into *material scanning* ones, which essentially treat the material as a surface of certain optical properties, and *procedural*, which typically model the whole three-dimensional wood interior. Our method is based on finding the corresponding parameters for a procedural model, such that a scanned surface of a wood can be transformed into a fully procedural and editable 3D model.

Material scanning and BTFs The appearance of a specific specimen of wood can be measured and encapsulated in various ways. Bidirectional Texture Functions (BTFs) [DVGNK99] store the full 6-dimensional surface reflectance using, essentially, lookup tables. Material scanners are available that capture diffuse reflectance, spatially varying roughness and a normal map, which can be fed into an appropriate BRDF implementation, such as the Disney Principled BRDF [BS12]. Both approaches can yield impressive realism for the specimen they were applied to, but suffer from immense storage requirements. Henzler et al. [HMR20] propose an interesting data-driven approach to generate solid textures from 2D specimen, wood being one of their use-cases. Their approach has the advantage of providing an infinite solid texture, but represents the input texture only qualitatively.

General solid noise functions Procedural noise can be a versatile tool to capture the texture of a wide range of materials, Perlin Noise [Per85] and Gabor Noise (e.g. [LLDD09]) being prominent examples. Matching the noise function parameters to obtain a desired input texture is also possible: For example, [GLLD12] show a method to match the power spectrum of Gabor Noise to exemplars, while [BMM*21] show an example of fitting the parameters of a Perlin Noise even in the presence of discontinuities. Since our goal is to closely match the ring structure, using power-spectrum based noise would require additional and hard to control steps, since matching the rings requires a precise match of both the noise signals' amplitude and *phase*.

Domain-specific procedural models A state of the art appearance model of wood is the work of Liu et al. [LDHM16], on whose work

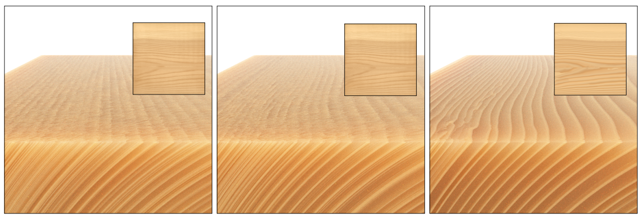


Figure 2: *False-color volume rendering of a solid texture that was fit using a differentiable material graph, (left and middle) compared to our approach. The density of the volume is set to $\sigma_t = 15$ for growth-lateness of greater 0.9, $\sigma_t = 0.5$ otherwise, both with a homogeneous single-scattering albedo $\alpha = (0.805, 0.578, 0.359)^T$ (RGB). The model in the first image was optimized with our board orientation estimate, but without an intial guess for the deformation field as a prior. The second image additionally uses our initial deformation field. The third image shows a visualization of our result; insets are the respective frontal, non-volume renderings. Notice how in spite of the very good match in terms of frontal appearance, the left and middle volumes show blurry ring boundaries.*

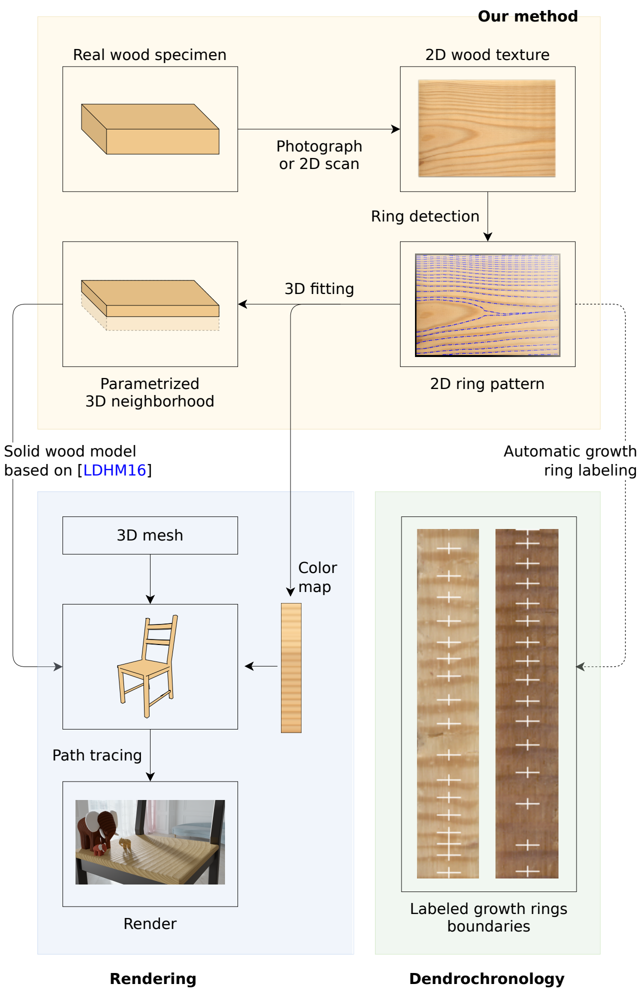


Figure 3: *Method pipeline overview. Our method starts with a real wood specimen (top left) that is scanned or photographed from one side. The 2D image is then processed to detect the wood growth rings (see also Fig. 4). This process is already useful for automatic ring labeling in dendrochronology (bottom right), where the input image is a 2D scan of a flattened drilled cylinder of a tree. For use in solid wood rendering (bottom left), the 2D ring pattern has to be further processed and fitted in a 3D space, since we are interested in parameters of a solid 3D appearance model based on [LDHM16]. As only a 2D scan is available as input, the 3D fitting can only be expected to approximate the actual specimen in its local neighborhood. The final path-traced render is using the fitted parameters and model on an arbitrary user-specified 3D mesh.*

we based our results (Fig. 3). Their model uses various distortion functions to modulate a cylindrical coordinate system. Together with a sawtooth-like function that describes the color saturation changes in the earlywood / latewood transitions, they also provide expressions for diffuse reflectance and the fibre direction. These can be used as base functions to define inputs to a BRDF shader. By the explicit way they model wood “age” on a sub-ring scale, their model can be seen as anatomically informed. Once the model parameters are tuned, a process that requires manual work by an expert, it is very expressive and can be used to model many species of wood. We are directly building on this model, and our contribution is that we are able to automatically recover its main parameters. We also recover a color map that describes coarse-level variations as a function of ring growth time.

Larsson et al. [LIY*22] extend the expressive power of wood-specific procedural solid textures by the integration of wood knots, based on a skeletal description of the branches and a distance-field based formulation of their respective area of influence to the growth rings formation.

2.2. Procedural appearance matching

Differentiable material graphs allow for the use of gradient-based optimization to polish material parameters, after a good initial guess has been found [GHYZ20, SLH*20]. Since the success of appearance matching depends on (and can be limited by) the expressive power of the underlying procedural model, neural loss functions [AAL16] seem to be best suited in finding plausible parameters for a wide range of material graphs, because they evaluate appearance differences in qualitative, perceptual terms. This allows these algorithms to converge to satisfactory results even if the expressiveness of the underlying graph is limited. However, none of these works provide a 3D solid texture that can be used to "carve" objects from or accurately replicate the curvature of the growth rings present in the target.

[LP00] show a half-automatic system that can extract parameters for a procedural 3D solid wood texture. From a thresholded input image, they are able to estimate ring frequency and spatial orientation of a given wood plank, based on image statistics. Ring variations are modelled statistically in terms of turbulence intensity and frequency. Their orientation estimates are based on statistical properties of the thresholded image, while ours is based on accurately identified rings. Our method also accurately recovers the deformation field.

2.3. Dendrochronology

Automated dendrochronological measurements rely on image processing techniques that use either photos [WQZ*10, FDBP16, FDBP17, Len20], or computed-tomography (CT) data [MGSS*21] as their input. Salient points of these methods is the application of 2D image processing techniques to the problem of growth ring identification. They achieve reasonable accuracy for coniferous species, but their accuracy greatly degrades in the presence of pores.

Notable data driven approaches include [FD18], who use a U-Net that was trained on manually labeled data, while [PAH*22] use a Mask R-CNN architecture. Neural networks seem to improve detection rates compared to the approaches based on traditional 2D image processing especially for ring porous wood, a category that our ring detection method shows on-par performance. Data driven approaches create likelihood maps for ring boundaries, that are subsequently thresholded to get exact locations. In contrast, our approach directly pinpoints the ring boundaries without the ambiguity of a likelihood.

Martinez et al. [MGSS*21] extend the idea of tree ring extraction to 3D data and reconstruct tree ring isosurfaces from X-ray computed tomography data based on edge detection. Basing the method on CT data makes the approach very robust, because of strong correlation between X-ray intensities and material density. The limited availability of CT scanners hinders the practical applicability of their approach.

In a more broader context, systems for the identification of tree species from either microscopic or macroscopic images build on segmenting anatomical features of wood, including its rings. Martins [Mar18] contains a survey of the proposed methods. Datasets of dendrochronological measurements are available through the work of [FDBP17], the *WIAD* database [RSB*21], both containing labels. The DendroElevator [oM22] database accumulates data from several sources, containing some very detailed tree ring images (up to microscopic scale) obtained from wood core samples. Some of their data are labelled.

2.4. Finding rings in images

Another field that considers the enhancement, identification, and segmentation of ring-like structures is *fingerprint identification*. The key differences to our domain are the bandwidth of ring frequencies (fingerprint ridges are quite evenly spaced), and the presence of branching (fingerprint ridges can split and merge, which usually is not the case for tree rings). This bandwidth limitation makes transferring techniques that are informed by or based on frequency-domain information (e.g., [THG16, LWL*20]) somewhat more problematic, but not impossible. To our knowledge, the connection between the fingerprint enhancement domain and growth ring identification was first discovered in [Jon08]. We base our approach on *curved Gabor filters* [Got11], which perform very well on fingerprint images. Gabor filters [Gab46] have been used across many disciplines for signal analysis and processing, and are often amongst the top performers from the suitable candidates. A very interesting property is their capability for retrieving phase information [Kon09], further [GR19, AG21].

3. Method

3.1. Overview

Our overall goal (Figs. 1, 3) is to reconstruct a subset of parameters that we can plug into the procedural wood appearance model proposed by Liu et al. [LDHM16]. Their starting point is a cylindrical coordinate system that has the z-axis aligned with the stem of the tree, and they apply distortion functions that work on both the radial distance and the azimuth. They further propose several versions of the spatially varying distortion field, like procedural noise, or using a lookup table ("distortion texture"). This can be either applied in a radially symmetric way, where the texture-space y-coordinates correspond to radial distances and the x-coordinates to values of z in tree space, or using a more elaborate helical wrapping to allow distortions that vary with the azimuth.

Our approach aims at recovering this distortion texture, and a basic color map, from a single photograph of the tangential plane of a wood cut. Please also refer to Figure 4 for a graphical overview.

We recover this data using a three-stage process. First, we calculate both an estimate of the board orientation, and an initial estimate of the distortion field from the locations of growth rings (Sec. 3.3, 3.4). The growth rings are found using curved Gabor filters (Sec. 3.2). Second, we polish the distortion field using a signal-phase informed optimization scheme (Sec. 3.6). Finally, a color map is calculated.

3.2. Curved Gabor filters

The key idea to curved Gabor filters is to calculate them in an image over curved regions that follow local orientation (Fig. 4, left to middle). The initial local orientation is obtained from the 2D image gradient: we use the Scharr operator [Sch00] after smoothing the image with a small Gaussian kernel. The gradient is averaged over a rectangular neighborhood, rotated 90° and normalized to obtain the initial local orientation field.

Once the local orientation is known, we compute a curved region around each point in the image. Each curved region consists of $2p+1$ parallel contours with $2q+1$ points each, in our implementation $p=q=80$. They are constructed by first sampling *perpendicular* to the local orientation for p steps, into both positive and negative directions. From each of the $2p+1$ points, a sampling walk *along* the local orientation in positive and negative directions for q steps, gives the pixel values of the curved region. As the step size is 1 pixel for both, we can look at the result as a pixelated patch of $2p\times 2q$ pixels that follows the local curvature.

Because the patch is rotated so that it follows the local orientation, growth rings will always be aligned horizontally when sampled isometrically. This allows estimating the local ring-frequency of a patch from the number of peaks is contains.

Using the (fixed) ring orientation inside each patch P and the detected frequency, we then calculate the convolution of the patch with a complex Gabor kernel $K=\{K_{\mathrm{re}},K_{\mathrm{im}}\}$ of an appropriate size to get the filtered patch P':

$$P' = P * K, \tag{1}$$

$$K_{\mathrm{re}}(x,y,\theta,f,\sigma_x,\sigma_y) = \exp\left(-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2}\right]\right)\cdot\cos(2\pi f x_\theta), \tag{2}$$

$$K_{\mathrm{im}}(x,y,\theta,f,\sigma_x,\sigma_y) = \exp\left(-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2}\right]\right)\cdot\sin(2\pi f x_\theta), \tag{3}$$

$$\begin{aligned} x_\theta &= x\cdot\cos\theta + y\cdot\sin\theta,\\ y_\theta &= -x\cdot\sin\theta + y\cdot\cos\theta. \end{aligned} \tag{4}$$

We accumulate the resulting signals P' over all patches, giving a complex valued filtered image $I_f=(I_{\mathrm{re}},I_{\mathrm{im}})$. Transforming to polar coordinates gives a magnitude-image $I_{\mathrm{mag}}=\sqrt{I_{\mathrm{re}}^2+I_{\mathrm{im}}^2}$, and a phase-image $I_\phi=\arctan_2(I_{\mathrm{im}},I_{\mathrm{re}})$. From the phase information, we can estimate the location of tree ring boundaries at where $I_\phi(x,y)=\frac{\pi}{2}$, the location of the signals negative zero crossing. The

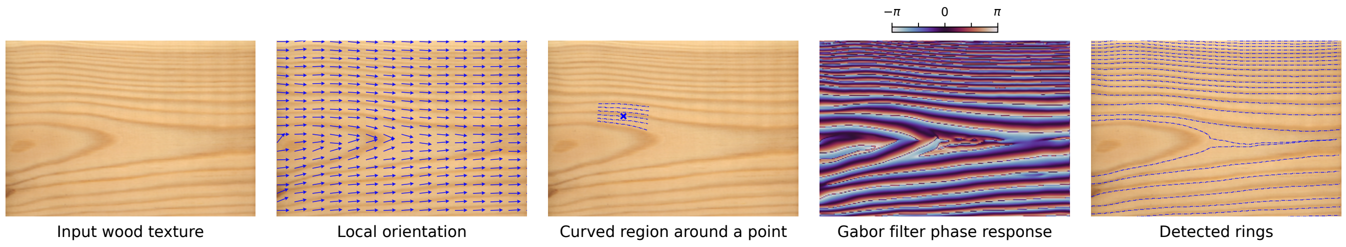


Figure 4: *Ring detection process. We start with an input 2D texture (left) and compute its local orientation. We then compute a curved region around each point in the image and apply a Gabor filter on it. Accumulating over all pixels gives the Gabor filter phase response. We then trace the rings in the phase image to detect their positions (right).*

phase image contains a sign ambiguity that we resolve later (Sections 3.6, 4.3).

As as side-effect to calculating the Gabor filter on curved regions, it is also possible to estimate local curvature and local ring frequency. The local curvature can be computed from the absolute value of the differences of the local orientation of the contours' middle points and their end points. This can be used to identify anomalies such as knots.

The final local orientation field is obtained from the phase image using the same procedure that calculated the initial local orientation estimate, which gives a smooth field that follows along the growth ring's trajectory.

3.3. Ring detection

To detect a growth ring, we seed the algorithm at any point $p = (x,y)$ where $I_\phi(x,y) \approx \frac{\pi}{2}$, and walk along the local orientation field into both positive and negative directions. By following the orientation field instead of just thresholding the phase image, our algorithm can trace rings even across small anomalies.

3.4. Estimating board location

With the detected visible growth rings it becomes possible to reason about the boards orientation within the 3D tree coordinate system. Using this positional information then allows us to calculate an estimate of the distortion field that would lead to the observed pattern. We show this for tangential cut planes, but the approach can easily be extended to other cuts as well.

Our position estimate starts with automatically finding the projection of the tree center (the z axis) onto the plane. It is usually located between the two rings with the largest mean distance from one another, with the remaining inter-ring distances being monotonically decreasing to both sides. This projection of the tree-center also gives the translation of the tangential board along its y-axis. From the median average inter-ring distance, we can induce the average scale of the rings in image-space coordinates. To find the x-translation of the observed plane, we fit the radial scaling factor and translation on the XY-plane using a brute force search in the neighbourhood of the initial estimate, which only takes seconds to compute. The resulting sign of the x-translation is ambiguous since we are only observing a plane.

3.5. Modeling growth distortions and initial guess

In their model, Liu et. al. [LDHM16] model distortions of points in tree space q in the radial and/or tangential direction, magnitudes m_r and m_t. The magnitudes are spatially varying and can be any function $f\colon R^3 \rightarrow R$, such as a Perlin noise, or a texture. The location of the distorted point q' is given by

$$q' = d(p) = q + m_r(q)\hat{r} + m_t(q)\hat{t}. \tag{5}$$

$\hat{r}$ and $\hat{t}$ are unit vectors into the radial and tangential direction, respectively. As the initial guess for the distortion field, we use the difference between the observed and the ideal, undistorted rings in image space and map this displacement to radial distortions $m_r(q)\hat{(r)}$ using the board location estimate. Tangential distortions $m_t(q)$ are not used here due to the ambiguity of radial and tangential distortions when only observing a planar projection of the deformed tree rings.

3.6. Polishing the initial deformation field

A sufficiently accurate initial guess of the deformation field is important for accurate convergence of the final polishing step. The periodic nature of the texture function constrains gradient-based optimization to the interval of the period the starting point is contained in. More precisely, the initial deformation field needs to have a local phase error of less than $\frac{\pi}{2}$ for best results.

Formally, we want to minimize an energy E that is a function of the reference image J and a rendered image I.

$$\underset{m_r,c,s_r}{\arg\min}\, E(J,I) \tag{6}$$

I is obtained from a rendering operator, $\mathfrak{R}$, that takes the boards orientation, given by the linear transformation T, a ring scaling factor s_r, the distortion field $d(q)$ and a color map $c(q_r)$ to an rendered image I:

$$I = \mathfrak{R}(T, s_r, d, c) \tag{7}$$

s_r encodes the mean width of growth rings in radial units, the color map c describes, for each ring individually, the color-changes observable as the tree growth forms early and late growth. The distortion field encodes the ring growth variations. In its simplest form, the rendering function is just a projection of the solid texture to image space by a cutting plane. Note that all functions need to be differentiable in order for gradient descent optimization to be applicable.

Loss function Optimizing using a per-pixel loss, or a feature loss, will result in a good appearance matched *image*, even good estimates for BRDF parameters other than color, through the use of a differentiable renderer as $\mathfrak{R}$. Problems arise due to the co-optimization of both distortion field and color map and will lead to a fit that lacks anatomical meaning (Fig. 2). This is due to the non-orthogonality of the parameters: From a reasonable gamut of colors in the color map, individual surface pixels can be modulated by using the distortion field, making it a proxy to achieve a certain surface color. This does not move the rings as a whole, but rather individual points. The result is a non-smooth, "fuzzy" deformation field that can even lead to ring fold-over.

To overcome this, the optimization needs to be constrained, or the metric improved. We have experimented implementing a monotonic constraint on the distorted radii to discourage foldover, and we also tried enforcing smoothness by convolving the distortion field with a Gaussian kernel in exponentially increasing intervals during optimization. However, we were not able to attain significant improvements. Another potential issue arises from the periodic discontinuity the procedural model contains, which is a direct result of the underlying sawtooth-like function that describes the growth periods of non-tropical wood grain (cf. figure 9). It has been shown that not accounting for discontinuities can lead to convergence issues (shown for example by Loubet et. al. in [LHJ19]) . We implemented an experimental reparameterizing renderer on top of the Mitsuba 3 renderer [JSR*22] to correctly handle texture discontinuities, and also tried smoothing the falling edge of the yearly growth boundary by convolving with a small Gaussian kernel. Neither lead to satisfactory convergence behaviour in our experiments.

We finally opted for improving the optimization metric by introducing using a loss based on signal phase. We also separate the optimization of the deformation field from the color map into a two stage process. Our loss encodes the difference in signal phase between points on the reference image, and points on the rendered image of the current iteration.

$$E = \left(J_\phi - I_\phi\right)^2 \tag{8}$$

The phase-based loss function lets the optimizer fit the locations of entire rings holistically. One could implement a phase-based loss on top of a bank of regular Gabor filters that recover the phase information for both the reference image I and the rendering of the current iteration. Propagating loss gradients back through this filter bank is very costly, however. Due to the constrained nature of the underlying solid texture (tree rings cannot be arbitrarily placed, but follow certain rules), the phase information can be directly obtained: rendering the fractional part of the radial coordinate in the distorted tree's cylindrical coordinate system, multiplied by 2π circumvents this bottleneck. The phase response of the curved Gabor filter calculated from the input image earlier then serves as the optimization target.

The phase ambiguity on the reference image can be resolved by assuming a fixed orientation, and flipping the phase values by 180° beyond the projected tree center, given it is visible in the image. We also note that the loss function of the optimization needs to respect the cyclic nature of phase information when taking differences.

Optimization algorithm and hyperparameters We implemented the fitting procedure using a custom differentiable rasterizer. The rasterizer is built on top of TensorFlow [AAB*15], which is used as a framework for reverse-mode automatic differentiation. We use their implementation of the Adam optimizer [KB14] to drive the optimization loop. The learning rate was set to 0.03, and we run the optimization for 500 epochs. A single run takes less than a minute on a NVIDIA RTX 2080 GPU for a target texture with a resolution of 750x735.

4. Results

4.1. Structurally matching wood appearance

We apply our method to several pieces of coniferous wood, where most rings are identified correctly. Difficulties arise where the plane grazes a growth ring tangentially, where tiny variations of the estimated board location and the ring distortion can create huge changes in appearance. This can be seen in Figure 8, 1st and 3rd column, where mistakenly identified rings add to the problem, leading to a noisy, implausible distortion estimate in that area. In contrast, the specimen in the third column does not suffer from this problem, since it does not contain the projection of the tree center.

Through the fit, we obtain a solid texture (see also Figure 5) that also supplies *semantic information* that we use to derive other BRDF parameters. In Figure 1, we show a rendering that modulates the roughness and displacement as a function to the lateness of growth of each ring, which results in a very realistic appearance with specular highlights that are in alignment with the late growth.

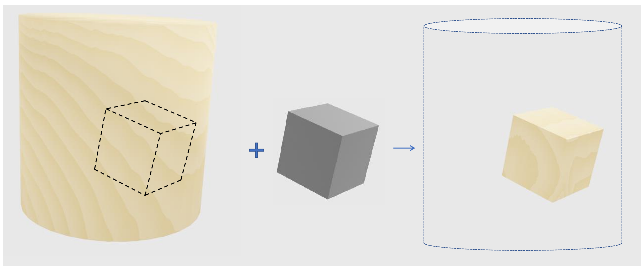


Figure 5: *The procedural wood model is an infinite 3D texture centered around a virtual tree (left). An arbitrary 3D mesh, such as a cube (middle), can be positioned within the 3D texture coordinate system accordingly to where the object was carved from the tree. This gives a textured solid wood object (right) that can be rendered.*

4.2. Volumetric appearance

For small material thicknesses, volumetric light transport becomes significant for the appearance of wood. Examples are decorative use-cases such as veneer lamp shades, or functional usages such as translucent wood touch panels, or architectural use of delignified wood. To illustrate the importance of subsurface reconstruction of the wood grain, we simplify the cellular anatomy into an approximate participating medium. Using our fit, we set the spatially varying density proportionally to the lateness of growth, latewood being more dense than earlywood. The volumetric properties were estimated to a scattering coefficient of $\mu_t \approx 16.0mm^{-1}$ for latewood, $\mu_t \approx 6.5mm^{-1}$ for earlywood, and a (wavelength dependent) single scattering albedo derived from the surface reflectance.

To come up with the density parameters, we applied the Inverse Adding-Doubling Method [Pra11] on data provided in [SKSG18], who performed measurements on Sugi wood, a coniferous subspecies, using micro spectrometer hardware. The heterogeneous single scattering albedo is calculated using the surface albedo mapping function proposed by Elek et. al [ESZ*17], their Eqn. 4. Fig. 6 compares the translucent appearance of both a model that was fit using our pipeline, and a simple extrusion of the surface ring pattern into the depth of the volume. The appearance difference is explained by the orientation of the latewood shells, which is perpendicular in the extruded case, but follows wood grain in the fit case. In Fig. 7 we show a volumetric rendering of a lampshade. The characteristic warm-colored translucent appearance wood shows when it is backlit is captured very well.

4.3. Dendrochonology

We confirm the robustness of our ring detection method by running it on a dendrochonological dataset. There, accuracy of detection is very important, since it is used as primary data for various other fields, including climate research.

To detect rings in dendrochonolgy images, the images are first being scaled to uniform width, and then converted to HSV colorspace, from which only the V-channel is used. The detection procedure is executed as detailed above. The phase sign ambiguity is resolved by assuming an image orientation with the youngest rings at the bottom of the image. The recovered rings are then compared to ground-truth by measuring the closest distance between the ground-truth label (one x/y coordinate pair for each ring). If the distance is smaller than 3 pixels, conforming with the evaluation shown in [FDBP17], the ring is counted as a match. Table 1 summarizes the performance of our approach. Performance for coniferous woods is good. For ring-porous woods, an anatomical variety that has shown to be problematic in previous work, our results are exellent. The high accuracy of our fit means that it can also supply additional data for dendrochronological evaluations. Fig. 9 shows a partial color map, superimposed with the inverse lightness, which correlates with the time the latewood transition occurred in the respective year.

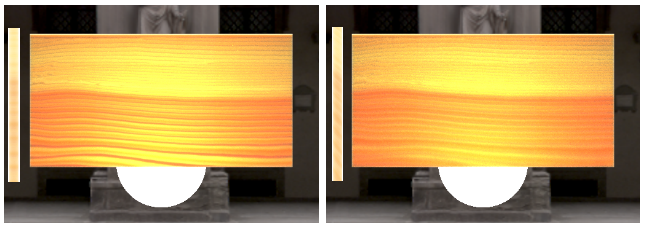


Figure 6: *Volumetric appearance is highly dependent on grain orientation - shown here are renderings of 1.8mm thick veneer sheets backlit by a spherical light source. The volumetric properties are derived from our fit (right), and extruded along the surface normal (left). Insets show side-views to illustrate grain*

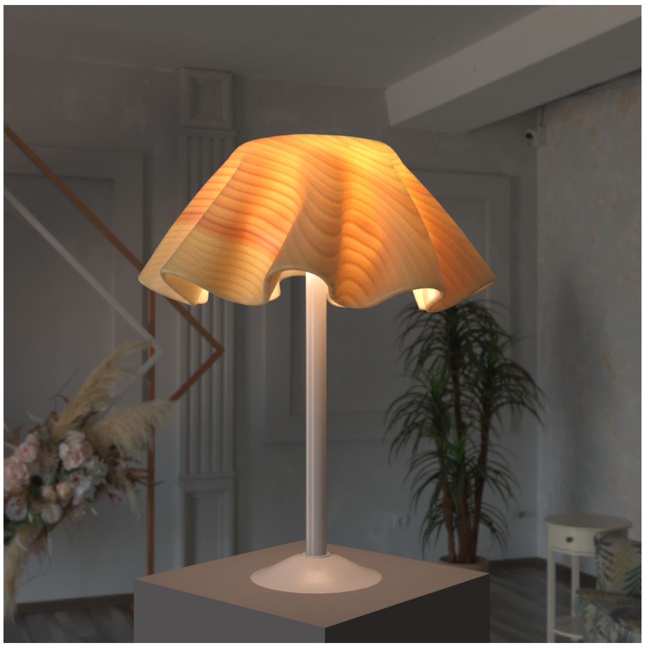


Figure 7: *Volumetric rendering of a lampshade that was cut from a fit model using our method (Fig. 8, 2nd column). Both single scattering albedo and volumetric density are heterogeneous.*

		Ours		[FD18]		[FDBP17]	
species	rings	sen	pre	sen	pre	sen	pre
larch	99	0.77	0.83			0.90	1.00
spruce	115	0.93	0.96			0.99	1.00
pine	108	0.79	0.91			0.93	0.97
ash	114	0.98	0.97	0.97	0.98	0.45	0.85
birch	30	0.17	0.06			0.86	0.74

Table 1: *Comparison of sensitivity (sen) and precision (pre) of our ring-detection method and two state of the art methods on a dendrochronological dataset. [FD18] was developed especially with ringporous wood in mind. Our method performs especially well on ringporous wood (ash) and gives decent results on confierous species (first three rows). Diffuseporous woods such as birch are problematic, where our approach produces many false positives.*

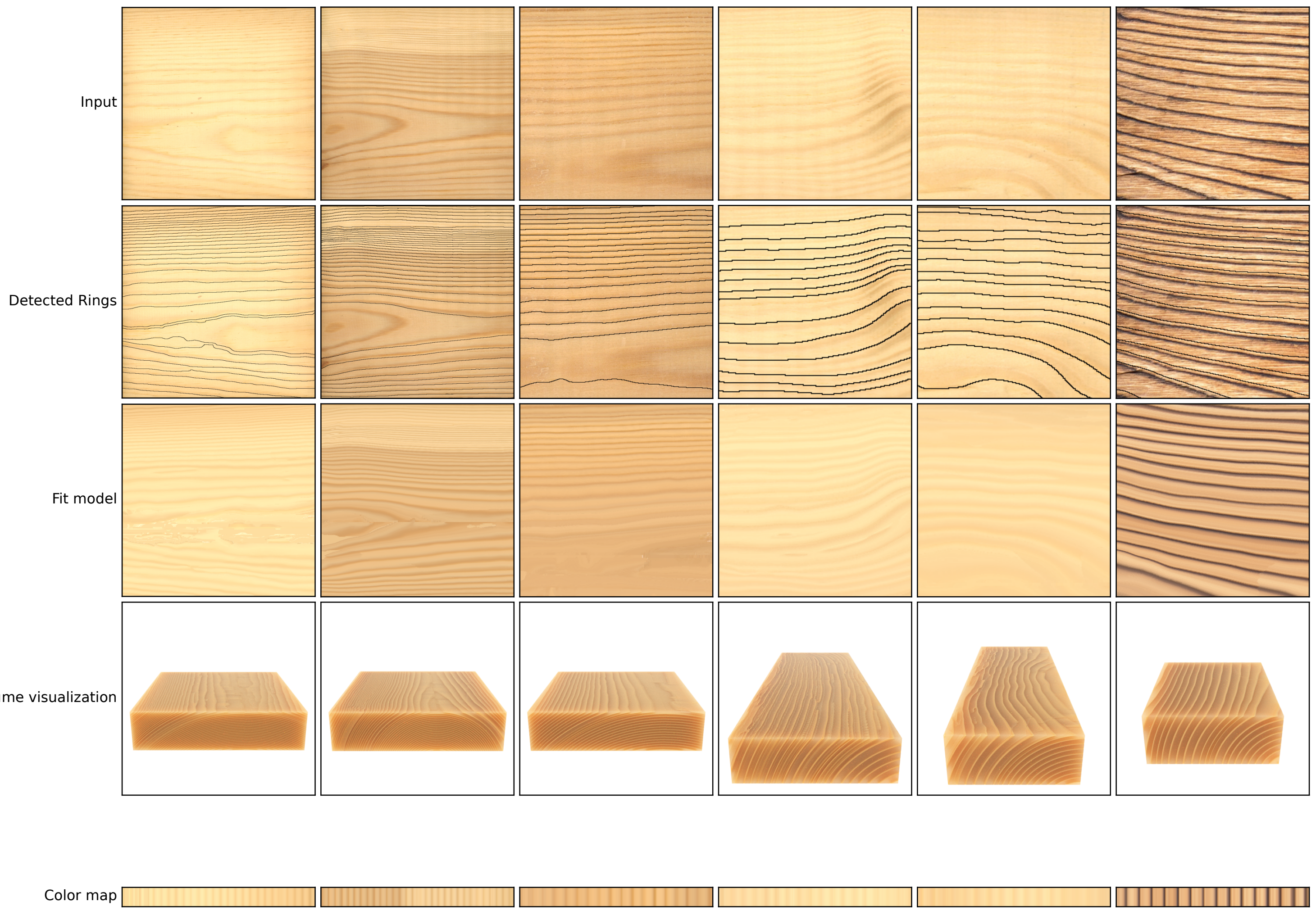


Figure 8: *Results obtained with our approach. Rows from top to bottom: Input texture, detected rings, frontal view of fit model (diffuse reflectance only), volumetric visualization (false color), fit color map.*

5. Conclusion

In this paper, we demonstrated curved Gabor filters to be an efficient technique for the extraction of tree ring information from single images of planar wooden boards. This information can then be used to build a realistic, anatomically meaningful 3D procedural texture that closely matches the structure of the log the original board was cut from.

The main focus of our work was to establish an automatic pipeline that is fast and accurate. The technology we ported and adapted from fingerprint detection not only serves as a tree ring detector with great performance, but also enables the efficient optimization of the ring deformations based on signal phase. The reconstruction of additional BSDF parameters by using differentiable rendering could further improve the model: but as the results of this paper show, the approach we currently use is already suitable for production work. Our pipeline further enables volumetric rendering of wood's subsurface light transport, and can pave a way to finding a faster, approximate BSSRDF, which we see as a interesting direction for further work.

We also see a potential of our robust ring extraction technique to be adopted by the dendrochronology community, where ring labelling is an important task that our method solves as a side effect.

References

[AAB*15] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G. S., Davis A., Dean J., Devin M., Ghe-

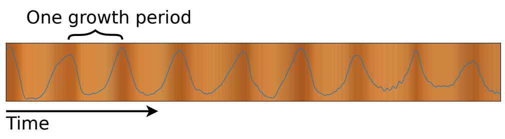


Figure 9: *From the extracted color map, it becomes possible to gain insight into the specifics of each growth cycle, here the latewood-transition. The plot shows the normalized and inverted lightness value of the HSL-transformed colorbar, smoothed by a gaussian kernel. The lightness is proportional to the cell-wall thickness and lignin content.*

MAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: http://tensorflow.org/. 6

[AAL16] AITTALA M., AILA T., LEHTINEN J.: Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (ToG) 35*, 4 (2016), 1–13. 3

[AG21] ALAIFARI R., GROHS P.: Gabor phase retrieval is severely ill-posed. *Applied and Computational Harmonic Analysis 50* (2021), 401–419. 4

[BMM*21] BANGARU S., MICHEL J., MU K., BERNSTEIN G., LI T.-M., RAGAN-KELLEY J.: Systematically differentiating parametric discontinuities. *ACM Trans. Graph. 40*, 107 (2021), 107:1–107:17. 2

[BS12] BURLEY B., STUDIOS W. D. A.: Physically-based shading at disney. In *ACM SIGGRAPH* (2012), vol. 2012, vol. 2012, pp. 1–7. 2

[DVGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG) 18*, 1 (1999), 1–34. 2

[ESZ*17] ELEK O., SUMIN D., ZHANG R., WEYRICH T., MYSZKOWSKI K., BICKEL B., WILKIE A., KŘIVÁNEK J.: Scattering-aware texture reproduction for 3D printing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 36*, 6 (2017), 241:1–241:15. 7

[FD18] FABIJAŃSKA A., DANEK M.: Deepdendro – a tree rings detector based on a deep convolutional neural network. *Computers and Electronics in Agriculture 150* (2018), 353–363. doi:https://doi.org/10.1016/j.compag.2018.05.005. 3, 7

[FDBP16] FABIJAŃSKA A., DANEK M., BARNIAK J., PIÓRKOWSKI A.: A comparative study of image enhancement methods in tree-ring analysis. In *International Conference on Image Processing and Communications* (2016), Springer, pp. 69–78. 3

[FDBP17] FABIJAŃSKA A., DANEK M., BARNIAK J., PIÓRKOWSKI A.: Towards automatic tree rings detection in images of scanned wood samples. *Computers and Electronics in Agriculture 140* (2017), 279–289. 3, 4, 7

[Gab46] GABOR D.: Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering 93*, 26 (1946), 429–441. 4

[GHYZ20] GUO Y., HAŠAN M., YAN L., ZHAO S.: A bayesian inference framework for procedural material parameter estimation. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 255–266. 3

[GLLD12] GALERNE B., LAGAE A., LEFEBVRE S., DRETTAKIS G.: Gabor noise by example. *ACM Transactions on Graphics (ToG) 31*, 4 (2012), 1–9. 2

[Got11] GOTTSCHLICH C.: Curved-region-based ridge frequency estimation and curved gabor filters for fingerprint image enhancement. *IEEE Transactions on Image Processing 21*, 4 (2011), 2220–2227. 4

[GR19] GROHS P., RATHMAIR M.: Stable gabor phase retrieval and spectral clustering. *Communications on Pure and Applied Mathematics 72*, 5 (2019), 981–1043. 4

[HKRFM17] HARTMANN F. P., K RATHGEBER C. B., FOURNIER M., MOULIA B.: Modelling wood formation and structure: power and limits of a morphogenetic gradient in controlling xylem cell proliferation and growth. *Annals of forest science 74*, 1 (2017), 1–15. 2

[HMR20] HENZLER P., MITRA N. J., RITSCHEL T.: Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 8356–8364. 2

[Jon08] JONSSON C.: *Detection of annual rings in wood*. Master's thesis, Linköping University, SE-601 74 Norrköping, Sweden, 11 2008. 4

[JSR*22] JAKOB W., SPEIERER S., ROUSSEL N., NIMIER-DAVID M., VICINI D., ZELTNER T., NICOLET B., CRESPO M., LEROY V., ZHANG Z.: Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org. 6

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6

[Kon09] KONG A. W.-K.: An analysis of gabor detection. In *International Conference Image Analysis and Recognition* (2009), Springer, pp. 64–72. 4

[LDHM16] LIU A. J., DONG Z., HAŠAN M., MARSCHNER S.: Simulating the structure and texture of solid wood. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 1–11. 1, 2, 3, 4, 5

[Len20] LENTY B.: Tree-ring growth measurements automation using machine vision. In *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2020* (2020), vol. 11581, International Society for Optics and Photonics, p. 115810V. 3

[LHJ19] LOUBET G., HOLZSCHUCH N., JAKOB W.: Reparameterizing discontinuous integrands for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia) 38*, 6 (Dec. 2019). doi:10.1145/3355089.3356510. 6

[LIY*22] LARSSON M., IJIRI T., YOSHIDAA H., HUBER J. A., FREDRIKSSON M., BROMAN O., IGARASHI T.: Procedural texturing of solid wood with knots. *ACM Transactions on Graphics (TOG)* (2022). 3

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse gabor convolution. *ACM Transactions on Graphics (TOG) 28*, 3 (2009), 1–10. 2

[LP00] LEFEBVRE L., POULIN P.: Analysis and synthesis of structural textures. In *Graphics Interface* (2000), vol. 2000, pp. 77–86. 3

[LVS*18] LI Y., VASILEVA E., SYCHUGOV I., POPOV S., BERGLUND L.: Optically transparent wood: Recent progress, opportunities, and challenges. *Advanced Optical Materials 6*, 14 (2018), 1800059. 2

[LWL*20] LE N. T., WANG J.-W., LE D. H., WANG C.-C., NGUYEN T. N.: Fingerprint enhancement based on tensor of wavelet subbands for classification. *IEEE Access 8* (2020), 6602–6615. 4

[Mar18] MARTINS A. L. R.: Towards automatic identification of woods from microscopic images. 4

[MGSS*21] MARTINEZ-GARCIA J., STELZNER I., STELZNER J., GWERDER D., SCHUETZ P.: Automated 3d tree-ring detection and measurement from x-ray computed tomography. *Dendrochronologia 69* (2021), 125877. 3, 4

[oM22] OF MINNESOTA U.: DendroElevator website, 2022. URL: https://dendro.elevator.umn.edu. 4

[PAH*22] POLÁČEK M., ARIZPE A., HÜTHER P., WEIDLICH L., STEINDL S., SWARTS K.: Automation of tree-ring detection and measurements using deep learning. *bioRxiv* (2022). doi:10.1101/2022.01.10.475709. 3

[Per85] PERLIN K.: An image synthesizer. *ACM Siggraph Computer Graphics 19*, 3 (1985), 287–296. 2

[Pra11] PRAHL S. A.: Everything i think you should know about inverse adding-doubling. *Oregon Medical Laser Center, St. Vincent Hospital 1344* (2011), 1–74. 7

[RSB*21] RADEMACHER T., SEYEDNASROLLAH B., BASLER D., CHENG J., MANDRA T., MILLER E., LIN Z., ORWIG D. A., PEDERSON N., PFISTER H., ET AL.: The wood image analysis and dataset (wiad): Open-access visual analysis tools to advance the ecological data revolution. *Methods in Ecology and Evolution 12*, 12 (2021), 2379–2387. 4

[Sch00] SCHARR H.: *Optimal operators in digital image processing*. PhD thesis, 2000. 4

[SKSG18] SUGIMOTO H., KAWABUCHI S., SUGIMORI M., GRIL J.: Reflection and transmission of visible light by sugi wood: effects of cellular structure and densification. *Journal of Wood Science 64*, 6 (2018), 738–744. 7

[SLH*20] SHI L., LI B., HAŠAN M., SUNKAVALLI K., BOUBEKEUR T., MECH R., MATUSIK W.: Match: Differentiable material graphs for procedural material capture. *ACM Trans. Graph. 39*, 6 (Dec. 2020), 1–15. 2, 3

[THG16] THAI D. H., HUCKEMANN S., GOTTSCHLICH C.: Filter design and performance evaluation for fingerprint image segmentation. *PloS one 11*, 5 (2016), e0154160. 4

[VCL*18] VASILEVA E., CHEN H., LI Y., SYCHUGOV I., YAN M., BERGLUND L., POPOV S.: Light scattering by structurally anisotropic media: A benchmark with transparent wood. *Advanced Optical Materials 6*, 23 (2018), 1800999. 2

[WQZ*10] WANG H.-J., QI H.-N., ZHANG G.-Q., LI W.-Z., WANG B.-H.: An automatic method of tree-rings boundary detection on wood micro-images. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)* (2010), vol. 2, IEEE, pp. V2–477. 3

2.4 Differentiable rendering of procedural solid textures containing parametric discontinuities

Suppose we sketch a solid texture for wood using a sawtooth wave modulated by a factor p that scales the texture radially. Using cylindrical coordinates $p_{cyl} = (r, \theta, z)^T$, this forms a basic growth rings pattern,

$$f(p_{cyl}, \pi) = frac(\pi r); frac(x) = x - \lfloor x \rfloor \tag{2.1}$$

This solid texture has a periodic discontinuity at the falling edge of the sawtooth function. Fig. 2.1, large image, shows a primal rendering of a lamp shade with this texture applied to the lightness of the basic wood color. Using a differentiable renderer that does not explicitly handle the discontinuity yields incorrect results when calculating the derivative of the pixel intensities with respect to the scaling-parameter (Fig. 2.1, right inset image). To obtain the correct derivative, the

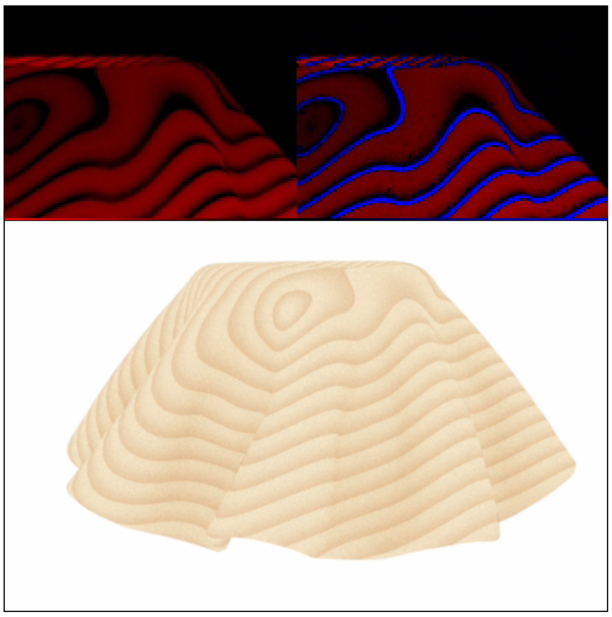


Figure 2.1: Application of the reparameterization technique on a discontinous solid texture. The insets show the image gradient with respect to a radial scaling factor for the growth rings, using reparameterization (left), and naive automatic differentiation (right). Below the insets the color map from negative gradient (red) to positive gradient values (blue) is shown

reparameterization technique discussed in Sec. 1.6.2 can be used on camera rays, Figure. 2.1, left inset. This requires the definition of the warp field V_π. For our example function, this is straight-forward: The local velocity field is obtained by taking the texture function's derivative with respect to the parameter of interest, in cartesian coordinates $v' = (p_x, p_y, 0)^T$. Since the texture changes along the surface of the object it is applied to, the velocity field needs to be projected onto the surface, to obtain the surface warp field. Practically and for arbitrary solid textures, the boundary velocity can itself be calculated using automatic differentiation if no obvious analytical solution is available, at a small performance penalty.

3. Contribution II: Appearance fabrication

3.1 Summary

The technology for Color 3D printing is available for quite a long time, and until 2017 color management was done analogously to the methods that are used in color 2D printing. These phenomenological approaches lead to decent results, but largely ignored the effects of the complex light transport that happens in objects of arbitrary geometry. In their work, Elek et al. [2017] introduced a new way to obtain printer instructions from an appearance specification, which was based on iterative optimization. Their approach did not require the pipeline to be differentiable, but instead used a heuristical refinement operator. This comes with the benefit to getting around the fact that for the printer, a discrete material to voxel assignment is required. This discretization makes the problem of finding an optimal printer recipe an instance of integer programming and thus will no longer be solvable using gradient descent.

Our differentiable approach is based on the idea of taking the halftoning out of the optimization loop and only apply it as the final step, after a (locally) optimal solution for the heterogeneous volume the printout represents is found. We instead optimize in material mixture space, taking advantage of the fact that the $(\sigma_a, \sigma_s, f_p)^T$ - parameterization allows to express any affine combination of the printing materials optical properties with an equivalent aggregate set of values.

Because this allows for the use of gradient-based optimization, it also unlocks the use of differentiable metrics to steer the optimization towards certain visual stimuli, and more use cases such as ink selection and objects with spatially varying translucency.

3.2 Baseline comparison to remapped planar solution

To establish that treating the printing optimization problem requires considering full three dimensional light transport, we try to extend the approach of Elek et al. [2017] to 3D. Their method can be generalized by first optimizing the texture map of the 3D object using their 2.5D-pipeline, and subsequently wrapping the resulting planar volume around the 3D mesh using the UV parameterization of the texture and an internal distance field.

This approach contains certain pitfalls: Texture mapping 3D geometry introduces stretch, which for the remapped solution translates into a change of voxel volume. The resulting changes in light scattering distance affect the appearance of the printed result.

We addressed this by manual pre-processing of the UV map to minimize stretch, and by carefully selecting the resolution and dimensions of the optimized 2.5D slab so that distances in UV space are kept constant during remapping.

We further add enough margin to work around light bleeding occurring at the borders of UV islands, and fill the borders using an inpainting algorithm. This helps the planar optimization by reducing the artificial color contrast between UV-covered texture map regions and the background. After optimizing a slab with the pre-processed texture applied using Elek et al. [2017]'s method, the result gets warped into its final shape of the 3D object. The result can then directly be printed. The transformation requires a mapping of all internal voxels to their respective closest surface point. We iterate over all volume voxels, find the corresponding surface point, which also stores the texture UV coordinates. This identifies the column of the 2.5D-solution, while the distance between the volume voxel and the surface voxel gives the lookup depth inside the column. Figure 3.1 compares printouts with a full 3D solution to this remapped 2.5D

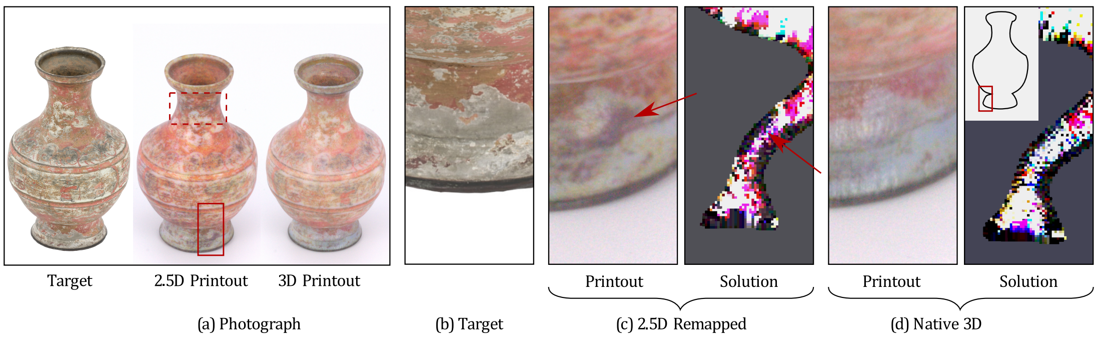


Figure 3.1: Baseline Comparison (a) of results achieved with planar scattering compensation (remapped to 3D using UV + depth coordinates) and our native 3D pipeline without our refinement improvements (the new update step and the content-aware gamut mapping). (c) and (d) show cropouts of a region highlighting problems of the 2.5D method and cross-sections thereof in comparison to the target appearance (b)

approach. To be fair, both images use the same optimization heuristic. The 2.5D vase (a) shows an overly saturated appearance. On thin regions such as the neck (dashed rectangle) and foot (solid rectangle and cropouts (c), (d)), the surface shows unwanted darkening. The CIEDE2000 average (calculated on renderings) is 15 for the 2.5D object, 10 for the 3D one. Depending on surface curvature, 2.5D

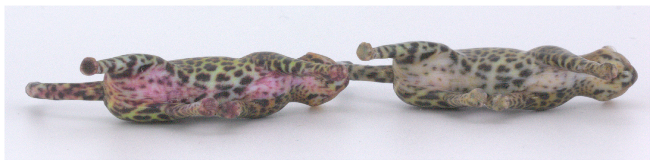


Figure 3.2: Color shifts on the abdomen and darkening on the paws of a 2.5D remapped printout (left) compared to a native 3D printout (right)

colums get their cross-sections expanded or contracted going into the depth of the volume. In concave regions, the colum's voxels expand to cover a greater volume. This in turn leads to a higher absorption and thus a darker surface appearance. Convex regions compress white voxels, likewise increasing the absorptive effect of other colored voxels. This can, in some cases, lead to significant color shifting, as shown in Figure 3.2.

A Gradient-Based Framework for 3D Print Appearance Optimization

THOMAS KLAUS NINDEL, Charles University, Czech Republic and Berufsakademie Sachsen, Germany
TOMÁŠ ISER, TOBIAS RITTIG, and ALEXANDER WILKIE, Charles University, Czech Republic
JAROSLAV KŘIVÁNEK, Charles University, Czech Republic and Chaos Czech a.s., Czech Republic

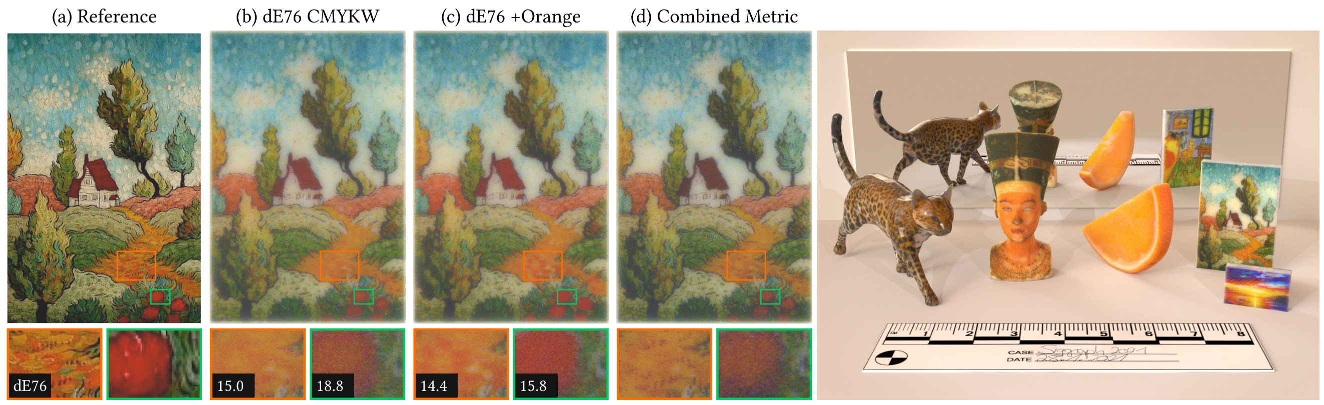


Fig. 1. Comparing optimization metrics for target (a) on a 2.5 mm slab. (b) was optimized for CIE dE76 using CMYKW materials. (c) adds a sixth orange material, improving color reproduction. (d) uses our combined metric with weights $\gamma_1 = 1$, $\gamma_2 = 9$, CMYKW. (e) shows the scale of all models used.

In full-color inkjet 3D printing, a key problem is determining the material configuration for the millions of voxels that a printed object is made of. The goal is a configuration that minimises the difference between desired target appearance and the result of the printing process. So far, the techniques used to find such a configuration have relied on domain-specific methods or heuristic optimization, which allowed only a limited level of control over the resulting appearance.

We propose to use differentiable volume rendering in a continuous material-mixture space, which leads to a framework that can be used as a general tool for optimising inkjet 3D printouts. We demonstrate the technical feasibility of this approach, and use it to attain fine control over the fabricated appearance, and high levels of faithfulness to the specified target.

CCS Concepts: • **Computing methodologies** → **Rendering**; *Volumetric models*; • **Applied computing** → *Computer-aided manufacturing*;

Additional Key Words and Phrases: Large-scale optimization, 3D Printing, volumetric light transport, volumetric optimization, computational fabrication, appearance reproduction, image metrics

Authors' addresses: Thomas Klaus Nindel, thomas@cgg.mff.cuni.cz, Charles University, KSVI MFF, Malostranské náměstí 25, Prague 1, 118 00, Czech Republic and Berufsakademie Sachsen, Hans-Grundig-Strasse 25, Dresden, 01307, Germany; Tomáš Iser, tomas@cgg.mff.cuni.cz; Tobias Rittig, tobias@cgg.mff.cuni.cz; Alexander Wilkie, wilkie@cgg.mff.cuni.cz, Charles University, KSVI MFF, Malostranské náměstí 25, Prague 1, 118 00, Czech Republic; Jaroslav Křivánek, Charles University, KSVI MFF, Malostranské náměstí 25, Prague 1, 118 00, Czech Republic and Chaos Czech a.s., Karlovo Námesti 288/17, Prague, 120 00, Czech Republic.

0730-0301/2021/8-ART178 $15.00
https://doi.org/10.1145/3450626.3459844

ACM Reference Format:
Thomas Klaus Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A Gradient-Based Framework for 3D Print Appearance Optimization. *ACM Trans. Graph.* 40, 4, Article 178 (August 2021), 15 pages. https://doi.org/10.1145/3450626.3459844

1 INTRODUCTION

Resin-based, inkjet 3D printers create colored objects by carefully jetting microscopic droplets of base inks (e.g., CMYKW, cyan, magenta, yellow, black, and white) to form the object's volume. Due to the inks' translucency, the printouts exhibit complex volumetric light transport. On one hand, this is required for spatial color mixing and curing with UV light, but also leads to edge blurring and color-bleeding. This significantly complicates the reproduction of a desired appearance, i.e., finding the volumetric ink arrangement that leads to the best possible printout fidelity. Early approaches based on naïvely placing colored inks on the surface of an object, or using extrusion-based techniques, lead to blur, reduced contrast, and poor color reproduction.

Recently, new approaches have been presented to tackle these issues, and aim at closing the gap between the intended appearance and the printed result. These approaches solve different aspects of the problem: high color reproduction fidelity [Babaei et al. 2017; Brunton et al. 2015; Shi et al. 2018], alpha channel translucency printing [Brunton et al. 2018; Urban et al. 2019], or compensating for lateral scattering to increase texture sharpness [Elek et al. 2017; Sumin et al. 2019]. The latter techniques employ highly-accurate Monte Carlo light transport simulation inside the printouts, in conjunction with a heuristic-based operator to form an iterative optimization method.

However, all these methods are inherently local: They only consider a small neighborhood on the surface at a time, and do not fully account for the potentially global influence each voxel has on the objects appearance.

We formulate the problem of finding a suitable material arrangement as an *inverse volume rendering problem.* Given the appearance of a volume under specified viewing conditions, inverse volume rendering tries to recover its physical properties, so that a forward rendering pass would closely match the original views. Previous applications of inverse volume rendering have already lead to impressive results. They include reconstructing a voxel model of a smoke plume from still images [Gkioulekas et al. 2016], obtaining volumetric models of planetary nebulae [Magnor et al. 2004], or recovering spatially-resolved scattering and absorption properties of living tissue [Hochuli et al. 2016].

Using this approach in the setting of inkjet 3D printing, the entire optimization problem can be formulated under a common, gradient-based framework. Previously discussed use-cases follow naturally from this, lifting some of the inherent restrictions the existing forward-only, or purely heuristic approaches inhibit. For example, our method can reproduce translucent appearances while also matching previous scattering-compensation work in quality.

Leveraging a modern GPU renderer, our method runs on desktop-grade hardware with reasonable timings and can thus also be used for meta-optimizations such as per-target ink selection from an extended set of printer materials. Our contributions for optimizing the appearance of full-color inkjet 3D printouts are:

First, we present a flexible and robust framework based on numerical minimization. The method captures the global interdependence between changes in the voxel configuration and their result on the target appearance. It is general with respect to the optimization goal, the optical properties of available inks, and can handle arbitrary geometries. Our model is physically based, incorporating all parameters of volumetric light transport.

Second, we propose using a volume parameterization that implicitly confines the optimization to the physical limits of inks and the model's manufacturability, without the need for precomputed color tables. This allows the runtime to be (almost) independent of ink count, in turn opening up applications such as ink-set selection.

Third we design a combination of 3D error metrics that can express a range of visual stimuli to optimize for, while allowing for intuitive control over the process.

2 BACKGROUND AND RELATED WORK

2.1 Inverse volumetric light transport

Implementations of inverse light transport commonly employ a gradient-based optimization algorithm. The required gradients can be computed from finite differences or obtained by differentiable rendering, with Gkioulekas et al. [2013] being the first to suggest Monte Carlo estimates of derivatives for inverse rendering. Li et al. [2018] presented a differentiable renderer that allows the calculation of derivatives with respect to arbitrary scene parameters. In Zhang et al. [2019], a general, operator-based theory of radiative transfer was presented, along with an unbiased estimator for calculating derivatives with respect to arbitrary scene parameters, adding those of volumetric light transport. Nimier-David et al. [2019] showed a retargettable renderer that can be configured to estimate the derivatives of light transport. Nimier-David et al. [2020] also recently proposed an adjoint method for back-propagating gradients, which greatly improves performance in settings with a large number of differentiable parameters. Since our approach requires calculating derivatives with respect to millions of volume parameters, this so-called "radiative backpropagation" technique is essential for us.

In Che et al. [2020], a learning-based technique is proposed that can be used for homogeneous inverse scattering applications; one of their networks was trained using a differentiable renderer, MitsubaDR, that was constructed by auto-differentiating a physically based renderer. Finally, Hašan et al. [2010] optimize material assignments for specified subsurface scattering properties using *discrete* optimization.

Khungurn et al. [2015] match the appearance of fibrous materials to several photographs using an inverse volumetric rendering pipeline. Papas et al. [2013] use a database of the scattering properties of pigment emulsions to iteratively find an optimal mixture that most closely matches a target appearance, which directly leads to a fabricable recipe.

2.2 Full-color fabrication

Color and beyond. While various publications tackled fabricating spatially-varying optical properties using 3D printing, e.g., subsurface scattering [Dong et al. 2010; Hašan et al. 2010], the first to show faithful *full-color reproduction* in inkjet 3D printing were Brunton et al. [2015]. They propose a geometry-adaptive error diffusion approach based on existing algorithms from 2D printing. Mapping 2D error diffusion filters onto the object's surface, they halftone the target layer by layer. They also discuss printer color management and building a color profile to allow gamut mapping.

Later, Brunton et al. [2018] extended their method to also support spatially varying translucency. Using lookup tables to map RGBA target colors to printer tonal values, they probabilistically insert transparent voxels into the volume. Translucency fabrication was further formalized by Urban et al. [2019], who provide a physically and perceptually meaningful definition of the alpha channel to create a device-independent standard for 3D printing translucency. Work was also done in controlling the surface reflectance of 3D-printouts, either by directly printing microgeometry [Luongo et al. 2020; Rouiller et al. 2013], or based on a controlled application of varnishes [Piovarči et al. 2020]. Recently, Zheng et al. [2020] used neural networks and end-to-end optimization to create static 4D light fields that can then be fabricated with inkjet 3D printers.

Halftoning alternatives. Error diffusion halftoning is not the only method to obtain a discretized volume for printing. Arguing that there are inherent limitations of halftoning, Babaei et al. [2017] propose a solution they call *color contoning.* In contrast to halftoning, which works on the object's surface, their method drives "stacks" of inks below the object's surface. Doing this substantially removes halftoning artefacts.

Shi et al. [2018] take this approach even further. They use a neural-network model for predicting color spectra of an ink stack. Their model can also be used in reverse, so that ink stacks, composed from their extended material library of 10 custom inks, can be used to achieve significantly improved color reproduction.

Full-color fabrication with Monte Carlo predictions. All the previously mentioned works are based on local color mixing, ignoring the contribution of global light transport. Elek et al. [2017] showed that by measuring the optical properties of basic inks, they can then use a full Monte Carlo simulation to faithfully predict the appearance of a 3D printout given its internal structure. These predictions are used as feedback in an optimization loop, heuristically pushing inks into the depth of the volume. Lateral scattering and color bleeding are reduced, leading to higher texture sharpness.

The original approach was limited to 2.5D planar slabs, but was later extended to full 3D geometry by Sumin et al. [2019]. They pay special attention to thin object geometries, where the colors of opposite faces create cross-talk. By computing a lookup table of color pairs achievable on opposing sides of variable-thickness slabs, they can perform content-aware gamut mapping. Their results show significant improvements of contrast and saturation for arbitrary object geometries.

Our method is inspired by Elek et al. [2017] and Sumin et al. [2019]. In contrast to their heuristic optimization, we employ a gradient-based optimization algorithm. By using end-to-end gradient estimates, our proposed method can determine to what degree every single voxel of the volume impacts the surface appearance, which is illustrated in Fig. 2, and improve the solution to lower the appearance error.

2.3 Image quality metrics

Numerical optimization methods are driven by quality metrics that compare intermediate results to a given target. We also require such metrics to be differentiable in order to be applicable within gradient-based optimization. Measuring the difference in appearance is a task similar to assessing image quality. This is a well-studied, yet challenging topic, mainly due to the non-linearity and subjectivity of the human visual system.

Beyond trivial measures like the mean squared error (MSE), *perceptual* quality metrics were designed to consider the human visual system. Color fidelity can, for example, be measured as the Euclidean distance in the perceptually uniform CIELAB color space (CIE76, McLaren [1976]), which was later expanded to CIEDE2000 [Luo et al. 2001]. Discontinuities in the latter prevent its usage in gradient optimization [Sharma et al. 2005].

Wang et al. [2004] proposed a structural similarity metric (SSIM) that also considers contrast and structure. The metric can be efficiently implemented using differentiable convolutions [Orihuela and Ebrahimi 2019], making it a good candidate for the use in gradient optimization and neural networks [Zhao et al. 2016a].

Due to its low sensitivity to a uniform bias, Zhao et al. proposed to combine SSIM with a color metric for an improved color accuracy. Chen et al. [2006] also noticed that SSIM favors blurred images over noisy ones, and suggested a modified edge similarity (ESSIM) metric. There are other SSIM alternatives available [Wang et al. 2003; Xue et al. 2013; Zhang et al. 2011], further improving its behavior. Notably, Preiss et al. [2014] propose a metric that is sensitive to differences in lightness, color and structure.

Recent work [Zhang et al. 2018] has shown that deep neural networks originally trained for various computer vision tasks can also be used for image quality judgement. Their performance has shown to be favorable over metrics such as the previously disussed ones, when comparing to the quality assessments of the human observer.

3 PROBLEM STATEMENT

Overview. The method takes a textured 3D object on its input. The object has a certain *target appearance*, which is usually a diffuse color texture that defines how the object should look like when printed. This object is geometrically discretized into a voxel grid, where each *surface* voxel (texel) has a given target color based on the original texture. The goal is now to find a configuration of the 3D printer's inks that, when printed, would give a 3D printout that resembles the input as well as possible. In order to do that, an optimization loop is used that predicts how a candidate 3D ink arrangement would look like, and then compares it to the target appearance. An error is calculated using image quality metrics, and is backpropagated through to the ink arrangement via gradients, allowing its iteratively refinement. After convergence of the optimization, the results are quantized into an printable volume.

Formalization. We will now formalize the problem using the notation detailed in Table 1 such that it is solvable by numerical, gradient-descent optimization.

Given the geometrical discretization of a textured object as an input, we call the voxels on the object's surface "texels.[1]" The set of all texels forms the target appearance, $T = (t_1, \ldots, t_I)$. Taking a given configuration of voxels V, we can predict the appearance C of the surface as

$$C = \mathfrak{R}(V), \tag{1}$$

where $\mathfrak{R}$ represents the *volume rendering function*. Our goal is to find a configuration of voxels V_{opt} that minimizes a loss function $e = E(C, T)$:

$$V_{opt} = \arg\min_{V} E(C, T). \tag{2}$$

To find V_{opt} using gradient-descent optimization, we require the gradient of the loss function w.r.t. the volume's parameters V. We can obtain the gradient by applying the chain rule:

$$\frac{\partial e}{\partial V} = \frac{\partial e}{\partial C}\frac{\partial C}{\partial V}. \tag{3}$$

As C is a vector of the appearance of all texels c_i,

$$\frac{\partial e}{\partial C} = E'(C, T) = \left(\frac{\partial e}{\partial c_i}\right)_{i=1}^{I} \tag{4}$$

represents the gradient of the loss function with respect to the appearance C.

Using then, for example, the sum of squared differences as the loss function:

$$E(C, T) = \sum_{i=1}^{I} (c_i - t_i)^2, \tag{5}$$

[1]Mind that these are not *directly* connected to the fragments of the object's texture maps, but are volumetric elements obtained after UV mapping and resampling.

Table 1. Mathematical Notation

Symbol	Usage
I, J, N	number of voxels at the volume's surface / total voxels of the volume / available printing materials
i, j, n	indices for surface voxels / volume voxels / materials
α_j	single scattering albedo of voxel j, $\alpha = \frac{\mu_s}{\mu_t}$
$\mu_{t,j}$	extinction coefficient of voxel j, $\mu_t = \mu_a + \mu_s$
$\mu_{s,n}$	scattering coefficient of material n
$\mu_{a,n}$	absorption coefficient of material n
ϕ	phase function
$\rho_{j,n}$	portion of the n-th material in the mixture of the j-th voxel
π	any volume parameter, e.g. $\mu_{t,1254}$ or $\rho_{421,2}$
$\mathbb{S}$	space of the voxel parameters, formed by (μ_t, α, ϕ)
$\mathbb{M}$	space of the voxel parameters, formed by the material mixture ρ_n
M_n	parameters of printing material n, $M_n = (\mu_{a,n}, \mu_{s,n}, \phi)$
c_i	appearance of the i-th surface voxel
C	appearance of the current solution, $C = (c_i)$
t_i	target appearance of the i-th surface voxel
T	target appearance, $T = (t_i)$
V	volume parameters in space $\mathbb{S}$, $V = (\alpha_j, \mu_{t,j}, \phi_j)$
V'	volume parameters in material space $\mathbb{M}$, $V' = (\rho_{j,n})$
$\mathfrak{R}(V)$	volume rendering function, $C = \mathfrak{R}(V)$
$z(V')$	mixing function $V = z(V')$
e	error value
$E(C, T)$	loss function, $e = E(C, T)$

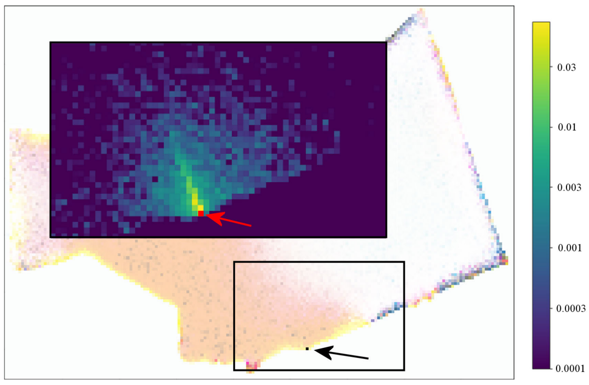


Fig. 2. Illustrating how voxels inside an object influence the color visible on the surface. A 2D slice through the profile of Nefertiti shows the single-scattering albedo in RGB per voxel. The inset visualizes the magnitude of Monte Carlo estimated $\partial c_i/\partial \alpha_j$ in one channel for the marked surface texel, i.e., how much each voxel influences the surface color in normal direction.

its gradient, consisting of the partial derivatives of E with respect to all volume parameters (here called π) can be obtained with:

$$\begin{aligned} \left(\frac{\partial e}{\partial \pi}\right) &= \sum_{i=1}^{I} \left(\frac{\partial e}{\partial c_i}\frac{\partial c_i}{\partial \pi}\right) \\ &= 2\sum_{i=1}^{I} \left((c_i - t_i)\frac{\partial c_i}{\partial \pi}\right), \forall \pi \in V \end{aligned} \tag{6}$$

Derivatives of the form $\partial c_i/\partial \pi$ encapsulate how the change in the appearance of one texel depends on the change of the parameter π. In other words, it is the first derivative of the rendering function with respect to this parameter. A visualization of these derivatives can be seen in Fig. 2.

By choosing an appropriate error function, the optimization is general with respect to the target appearance specification. It allows us to control whether the optimizer should emphasize, for example, texture sharpness, or optimize for highest possible color fidelity. We discuss this trade-off further in Sec. 4.5.

In order to embed this optimization task into the setting of inkjet 3D printing, a few extra steps are required: The discretization of the optimization results in a printable specification, which is discussed in Sec. 4.7, the voxelization of the objects geometry as an input to the pipeline, which can be obtained using distance fields (domain-specific discussions by Brunton et al. [2015]; Sumin et al. [2019]), and the optical properties of a set of printing materials can be obtained by various measurement setups, detailed in the state of the art report [Frisvad et al. 2020].

4 FORMING THE OPTIMIZATION LOOP

Using the relationship between the rendering function, the loss function and their derivatives as a starting point, we will now describe how to form a closed loop for the optimization of the volume configuration. Fig. 3 describes this loop graphically.

4.1 Derivative estimation

In practice, partial derivatives of the form $\partial c_i/\partial \pi$ can be obtained using a Monte Carlo estimator. Mathematically, it can be constructed from the path-space integral of light transport, as shown by Khungurn et al. [2015] and Zhang et al. [2019]. Implementations of these estimators usually work in what is called *forward-mode*: Parallel to computing radiance values, these algorithms also estimate the radiance derivatives alongside. Forward-mode refers to the order in which the graph of computations created by the radiance estimator is traversed for obtaining derivatives, in the context of rendering from scene parameters to radiance estimate. Forward-mode derivatives can be obtained automatically by a process called *auto-differentiation* (AD). We kindly refer the reader to Baydin et al. [2017] for an overview. This way of implementing derivatives requires one graph traversal for each differentiable parameter. Since we want to optimize millions of volume parameters, the associated computational cost quickly becomes prohibitive. Our initial experiments were based on such an estimator, taking tens of thousands of core-hours to optimize a 15 mm tall volume. Additionally, the resulting Jacobian matrix (in our case, derivatives of all texel values with respect to all volume parameters, a $I \times 2J$ matrix) leads to an unfavorable space-complexity. The use of *reverse-mode* AD, as implemented in Mitsuba2 [Nimier-David et al. 2019]), can ease this space-complexity by backpropagating derivative values through the renderer. However, reverse-mode AD requires storing computational graphs, which in case of a renderer can get very complex.

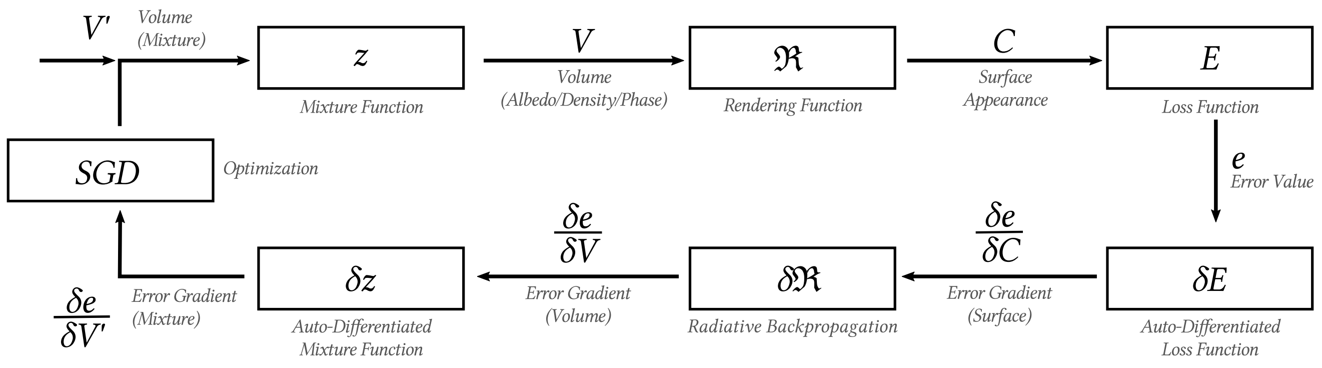


Fig. 3. Flow of data and their derivatives through the optimization loop. The voxelization of object geometry and the quantization of the optimization results are omitted for clarity. These are only executed once at the beginning and the end of the optimization process, respectively.

4.2 Radiative Backpropagation

We implement the volume derivatives using radiative backpropagation [Nimier-David et al. 2020]. It is a domain-specific instance of reverse-mode AD that eliminates the need for storing and traversing the renderers computational graph. Instead, the method propagates "derivative radiation" through the scene using light transport algorithms, an adjoint technique also seen in [Auzinger et al. 2018]. This radiation is emitted by the sensor and "consumed" by the scene parameters, effectively propagating all derivatives in one pass. This leads to major improvements in speed of computation and mitigates the storage requirements of traditional reverse-mode AD.

In our setting, the input for Radiative Backpropagation is derivative values $\partial e/\partial C$. As we are interested in derivatives with respect to the parameters of volumetric light transport, the output is $\partial e/\partial V$ (see Eq. 3, the second factor).

4.3 Derivatives in material space

Assuming a fixed phase function and its parameters, one way to implement the optimization is using the (μ_t, α) parameterization of voxels. However, a discretization step is then needed to turn the optimized volume into a printable specification. Because the printer gamut is smaller than what this voxel-parametrization can describe, it also implies the need for gamut mapping.

Other approaches incorporate the discretization step into the optimization loop (e.g., Elek et al. [2017]), guiding the optimizer through the limited gamut of printable medium parameters. Since our approach requires end-to-end derivatives, it would require calculating derivatives of the discretization step.

We propose an alternative solution that maintains differentiability and adjusts the volume properties in a space that is directly transformable into a material configuration. Assuming for the moment that the printer mixes the inks prior to placing them, we can express the optical properties of a voxel as a continuous affine combination of these materials. We call this the *material-space parameterization* of the medium:

$$V' = (\rho_{j,1}, \rho_{j,2}, ..., \rho_{j,N}) \quad \sum_{n=1}^{N} \rho_{j,n} = 1, \rho_{j,n} > 0\ , \tag{7}$$

Table 2. Material Parameters. Materials in **bold** represent the Stratasys "Vero Opaque Rigid" Family, as measured by Elek et al. [2017]. The remaining lines show hypothetical materials used for evaluation. The extinction-coefficient μ_t and the single scattering albedo α are given in the three Red, Green and Blue color channels. All materials are using a Heney-Greenstein phase function with $g = 0.4$

	$\mu_t [mm^{-1}]$			α		
Material	R	G	B	R	G	B
Cyan	9.0	4.5	7.5	0.05	0.7	0.98
Cyan2	6.0	3.0	5.0	0.05	0.95	0.98
Magenta	2.5	3.0	10.0	0.98	0.1	0.9
Magenta2	1.7	2.0	6.7	0.98	0.05	0.97
Yellow	2.25	3.75	19.0	0.997	0.995	0.15
Yellow2	3.0	5.0	19.0	0.92	0.92	0.15
Orange	3.0	4.0	15.0	0.98	0.32	0.114
Black	5.0	5.5	6.5	0.35	0.35	0.35
White	6.0	9.0	24.0	0.9991	0.9997	0.999
Transparent	10^{-4}	10^{-4}	10^{-4}	1.0	1.0	1.0

where N is the number of available printing materials and $\rho_{j,n}$ their weights in the mixture for voxel j. The materials are parameterized by their absorption and scattering coefficients, and their phase function, so that $M_n = (\mu_{a,n}, \mu_{s,n}, \phi_n)$. An example instance of printing material parameters, as they were measured by Elek et al. [2017], is shown in Table 2. This idea of parametrizing volumetric light transport using pre-set materials is analogous to the material mixture method presented by Gkioulekas et al. [2013].

Gradients of the loss function with respect to this material space parametrization can be obtained by applying the chain rule:

$$\frac{\partial e}{\partial V'} = \frac{\partial e}{\partial V}\frac{\partial V}{\partial V'}. \tag{8}$$

Such a volume parametrization allows the optimizer to interpolate within the whole range of the materials' properties to achieve an appearance match. As we will show in Sec. 4.7, it also allows for a simple discretization to ensure manufacturability. Finally, it allows us to easily adapt to different printing materials, without any precalculation and with negligible overhead in computation time.

4.4 Relation of material space and optical parameters

3D printing uses a subtractive color mixing system similar to 2D printing. In addition to the cyan, magenta, yellow, and black (CMYK) inks, a white substrate (W) and potentially a transparent material (T) are available. Mathematically, an affine combination of these base materials has absorption and scattering characteristics that can be computed as a linear combination. Please note that this is only possible using the absorption- and scattering-coefficients, since their behaviour is linear. In contrast, a weighted sum of α and μ_t does not work the same way.

Let z describe this mapping from the (continuous) material space $(\rho_1, \dots, \rho_N) \in \mathbb{M}$ to the space $(\mu_t, \alpha) \in \mathbb{S}$. Omitting the voxel index j for readability, we define it as:

$$z : \mathbb{M} \to \mathbb{S} = \left(\sum_{n=1}^{N} \rho_n(\mu_{s,n} + \mu_{a,n}), \quad \frac{\sum_{n=1}^{N} \rho_n \mu_{s,n}}{\sum_{n=1}^{N} \rho_n(\mu_{s,n} + \mu_{a,n})} \right). \tag{9}$$

We assume all materials share the same phase function (Henyey-Greenstein, $g = 0.4$). Phase functions mix in the following way [Gkioulekas et al. 2013]:

$$\phi(\cos\theta) = \frac{\sum_{n=1}^{N} \rho_n \mu_{s,n} \phi_n(\cos\theta)}{\sum_{n=1}^{N} \rho_n \mu_{s,n}}. \tag{10}$$

It can be easily seen that our assumption of equal phase functions for all materials results in the mixed phase function being the same as well.

Material space can be perceptively ambigous in several ways. First, and depending on the printing materials chosen, there can be many combinations in $\mathbb{M}$ leading to the same optical properties in $\mathbb{S}$. As an example, consider mixing a single voxel of gray color from CMYKW. Even if CMY were perfectly orthogonal, it can be mixed in several ways: either by an appropriate ratio of black and white ink, or by replacing black with an even mixture of cyan, magenta and yellow, combining their absorption. The material densities can differ, so possible solutions for a desired color can have different *scattering* behavior, while having the same single-scattering *albedo*. This creates a mixing ambiguity that is more complex than in 2D printing.

Second, creating a specific appearance is *locally* ambigous even within space $\mathbb{S}$. This was formalized with *similarity theory* [Wyman et al. 1989a,b; Zhao et al. 2014], which studies the equivalence classes of the volumetric parameters in radiative transfer, where different sets of parameters can lead to equivalent measurements of the radiance field. And finally, heterogeneous media can also have *non-local* ambiguities, following the observation that in such a medium, different spatial distributions of materials can lead to similar measurements [Gkioulekas et al. 2016].

Our parametrization helps the optimizer navigate these ambiguities. Following the example of mixing a gray color from above, the black ink will have a larger gradient value than each of the individual CMY inks, so the optimizer will favor the use of black ink for the respective voxel. This can be seen, for example, in Fig. 1, where the optimizer favors the use of a newly introduced orange ink for the creation of the orange patches and the red flowers in image Fig. 1(c). Further, assuming identical phase functions for all materials reduces the local ambiguities described by similarity theory. Empirical evidence [Gkioulekas et al. 2016, 2013; Zhao et al. 2016b] also suggests that this parametrization can be used in inverse rendering to recover ground-truth scattering properties in spite of the ambiguities.

Our proposed pipeline does not require computing the inverse of z, but only partial derivatives with respect to affine weights ρ. In practice, we implement the computation of gradients of the form $\partial V / \partial V'$ (Eq. 8, second factor) by auto-differentiating the material mixing function $z(V')$, Eq. 9. Together with $\partial e / \partial V = z'(V')$, one obtains an end-to-end derivative chain of the optimized function as visualized in Figure 3.

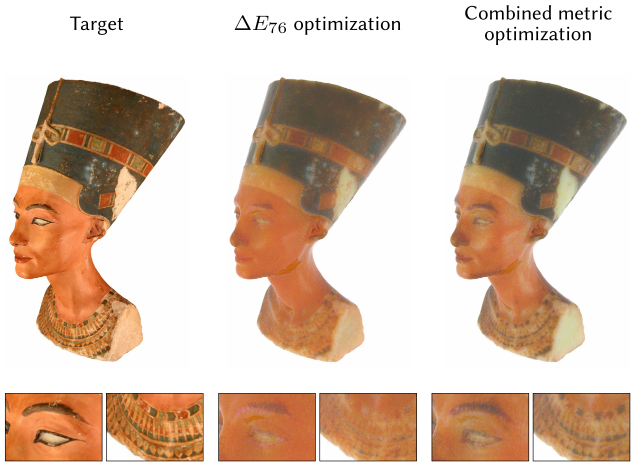


Fig. 4. Comparison of the results optimized with different loss functions on this 4 cm tall 3D-scanned model. We can see that only relying on a ΔE_{76}-based loss function (Eq. 11) leads to a good overall color fidelity, but poor contrast in details, such as in the eye region or the necklace. That can be significantly improved with an introduction of 3D voxel-based loss function that take the structure into account as well. Similar behavior can be seen in our slab experiments in Figs. 1 and 19.

4.5 Error metrics

Our framework is driven by minimizing the loss function $E(C, T)$. It governs the extent with which each visual stimulus affects the final results. Based on the exploration of existing image difference metrics in Sec. 2.3, we want to find a loss function for our use case with the following qualities:

- *differentiable*, in order to compute $\partial E / \partial C$,
- without discontinuities affecting the optimizations convergence,
- *perceptual*, to take the human visual system into account, considering *color fidelity*, *contrast*, and *structure*, and
- *geometry-aware*, to preserve local features.

In Fig. 4, we illustrate how significant the difference can be between the convergence with a simple error metric, compared to our more sophisticated metric combination. The remainder of this subsection discusses how we built this single well-behaved metric by first selecting suitable 2D metrics and then generalizing them to voxelized 3D surfaces.

Color fidelity. In order to steer the optimization towards maximum color fidelity, we use the mean squared error in the CIELAB color space, inspired by ΔE_{76} (CIE dE 76). Note that MSE-style errors like ΔE_{76} are computed per-pixel (per-texel) and are invariant to the local neighborhood, hence they can be directly used in 3D:

$$E_1(C,T) = \frac{1}{I}\sum_{i=1}^{I}\left(\left(c_{i,L^*}-t_{i,L^*}\right)^2+\left(c_{i,a^*}-t_{i,a^*}\right)^2+\left(c_{i,b^*}-t_{i,b^*}\right)^2\right), \quad (11)$$

where indices L^*, a^*, and b^* denote L*a*b* components of the respective texel.

Contrast and structure. E_1 is a global color metric that is not sensitive to visual stimuli such as local contrast and structure. To improve on that, we employ a normalized structural similarity index measure (SSIM), computed over local neighborhoods with a Gaussian sliding window and convolutions [Orihuela and Ebrahimi 2019].

While SSIM was originally designed for 2D signals, we generalize it to 3D as follows: Let P_{c_i} and P_{t_i} denote local neighborhoods of the i-th predicted and target texels. Let G_σ denote a 3D Gaussian kernel with a standard deviation σ. The mean intensities $\bar{c_i}$ and $\bar{t_i}$, standard deviations σ_{c_i} and σ_{t_i}, and covariance $\sigma_{c_it_i}$ are then given by:

$$\bar{c_i} = G_\sigma * P_{c_i}, \qquad \sigma^2_{c_i} = G_\sigma * P^2_{c_i} - \bar{c_i}^2, \quad (12)$$

$$\bar{t_i} = G_\sigma * P_{t_i}, \qquad \sigma^2_{t_i} = G_\sigma * P^2_{t_i} - \bar{t_i}^2, \quad (13)$$

$$\sigma_{c_it_i} = G_\sigma * \left(P_{c_i}\cdot P_{t_i}\right) - \bar{c_i}\bar{c_t}, \quad (14)$$

where $*$ denotes the discrete convolution operator. For the i-th texel neighborhood, local SSIM is then computed as:

$$SSIM(c_i,t_i) = \frac{2\bar{c_i}\bar{t_i}+K_1}{\bar{c_i}^2+\bar{t_i}^2+K_1}\cdot\frac{2\sigma_{c_it_i}+K_2}{\sigma^2_{c_i}+\sigma^2_{t_i}+K_2}, \quad (15)$$

where $K_1 = 0.01^2$ and $K_2 = 0.03^2$ are constants ensuring numerical stability. The final loss function $E_2(C,T)$ is then computed as a mean over all local neighborhoods, considering the three L*a*b* components separately:

$$E_2(C,T) = 1 - \frac{1}{I}\sum_{i=1}^{I}\sum_{L^*a^*b^*} SSIM(c_i,t_i). \quad (16)$$

For computing E_2, we experimentally observed better behaviors with the L*a*b* components being normalized to $[0,1]$.

Edge similarity. As Chen et al. [2006] observed, SSIM prefers blurred data over noisy data. This poses a problem, since we use a Monte Carlo estimator with inherent variance. They propose to modify SSIM with the Sobel operator used for edge detection, denoting their metric as ESSIM.

Here, we use a simpler error E_3 based on MSE between the texels with a Sobel operator applied. S_x, S_y, and S_z denote the 3D Sobel kernels along the x, y, and z axes, respectively. The convolution between the kernels and C or T approximate the derivatives of the texels' signal along the three axes. Computing an MSE over the results in each of the three axes separately, and doing the same with the L*a*b* components, gives our third metric:

$$E_3(C,T) = 100\cdot\frac{1}{3}\sum_{x,y,z}\sum_{L^*a^*b^*}\frac{1}{I}\frac{1}{4}\frac{1}{32}\sum_{i=1}^{I}\left(S*C - S*T\right)^2. \quad (17)$$

The normalization constants are given by the number of axes (3), the minimum and maximum values in the convolution (-32 and 32), and the maximum value after the square ($(-1-1)^2 = 4$). The additional factor of 100 is used to ensure the typical error values are in the similar range as those of E_1 and E_2, which leads to a more intuitive weighting in the final metric. Note that similarly to E_2, we are considering the normalized L*a*b* color space.

Final metric. We obtain the final loss function $E(C,T)$ as a weighted sum of the proposed metrics. To ensure that changing the individual weights does not change the overall magnitude of the error value, we define the sum such that the weights are always normalized:

$$E(C,T) = \frac{\gamma_1}{\gamma}E_1(C,T) + \frac{\gamma_2}{\gamma}E_2(C,T) + \frac{\gamma_3}{\gamma}E_3(C,T), \quad (18)$$

where γ_1, γ_2, and γ_3 are the weights of E_1, E_2, and E_3, respectively, and $\gamma = \gamma_1 + \gamma_2 + \gamma_3$. In Fig. 19, we illustrate how changing the individual weights influences the result. Experimentally, we observed that the highest overall color accurancy and saturation are achieved by setting $\gamma_2 = \gamma_3 = 0$. However, that leads to a loss of contrast and details, which may be significant in certain situations, such as the Nefertiti model in Fig. 4. Hence, we recommend using $\gamma_1 = 1$, $\gamma_2 \in [3,9]$, and $\gamma_3 \leq 3$, depending on the desired appearance. Higher values may lead to an unpleasant loss of saturation, in case of γ_2, or substantial color casts, in case of γ_3.

4.6 Optimization algorithm

We implemented the optimization using the TensorFlow framework [Abadi et al. 2015]. The renderer is implemented as a function into the framework, using the custom gradient decorator. AdaGrad [Duchi et al. 2011] is used as the optimization algorithm, but also experimented with a simple gradient descent with momentum and did not notice any significant differences. We constrain the optimizer to positive values using the sigmoid function. A successive per-voxel normalization of the ink mixture vector constrains the result to the feasible set.

This is in contrast to projected gradient descent (e.g., [Duchi et al. 2008]), where Euclidean projection onto the feasible set is used after applying the gradient in each step. It is also possible to use exponentiated gradient descent (Shem-Tov et al. [2020] show a domain-specific application). Both algorithms perform favorably if only few components of the inputs are relevant for the loss, which is the case for bigger volumes. There, voxels far beneath the surface have negligible influence on surface appearance. We see this as an interesting direction for future work.

4.7 Probabilistic ink quantization

The material space $\mathbb{M}$ parametrization introduced above assumes a printhead that physically mixes inks within the printed volume.

In reality, however, the printer can only deposit one material per addressable voxel and relies on spatial mixing together with lateral light transport to blend colors. Thus, after the optimization loop, the resulting volumes V' with affine weights require discretization to obtain a valid printer input.

Existing halftoning or contoning approaches [Babaei et al. 2017; Brunton et al. 2015; Sumin et al. 2019] could be applied directly, but without their color mapping as we operate directly in material space. We propose a simple solution based on random sampling, exploiting the fact that printers can reach significantly higher resolutions than the 300 DPI we run our optimization in. While upsampling to the printer resolution, we stochastically assign a single material to each voxel based on its affine weights: For each (micro-)voxel in the higher-resolution quantized grid, an ink is drawn at random, using the corresponding (macro-)voxel's material mixture as the probability distribution. At a native resolution of $600 \times 300 \times 940$ DPI, this gives a ratio of about $1:6$ between the number of voxels in the optimized and discretized grids of the printer. Figure 5 visualizes this procedure in a 2D example where a lower-resolution grid is upsampled to yield a discrete printer assignment.

A downside of this simple solution is that random sampling is known to introduce white noise. To some extent, this is mitigated by the fact that the scattering medium acts as a low pass filter that removes high frequency components, the cutoff frequency being a function of the opaqueness of the materials. However, some of this noise is still visible on the optimized objects, seen in Figs. 10 and 11 as graininess on the object's surface. In Fig. 6 we show a side-by-side comparison of a continuous material space and corresponding probabilistically quantized versions: we see that the remaining noise has a small impact on the results. In the future, implementing more sophisticated halftoning techniques, like the ones mentioned above, can help eliminate the noise introduced by random sampling.

4.8 Simulations vs. real printouts

Our printer model assumes perfectly disjunct, cubical voxels that are each filled with exactly one material. In contrast, the actual printing process is subject to a variety of imperfections. Physical mixing of adjacent voxels can occour around their boundaries.

This is especially noticable on the printout's surface, where support material and colored inks interact. To counter this, prints can be done with thin wrap of transparent material, that is subsequently removed by abrasion. This will also remove remnants of support

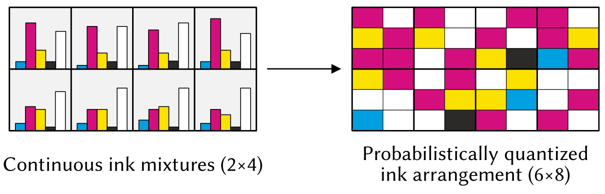

Continuous ink mixtures (2×4)
Probabilistically quantized ink arrangement (6×8)

Fig. 5. A diagram of a discrete grid with continuous ink mixtures (left) and its corresponding probabilistically quantized ink arrangement (right). In this example, the quantized grid has two times higher resolution in the horizontal axis, and three times higher in the vertical axis, which roughly corresponds to the XZ axes of our $600 \times 300 \times 940$ DPI quantized grid.

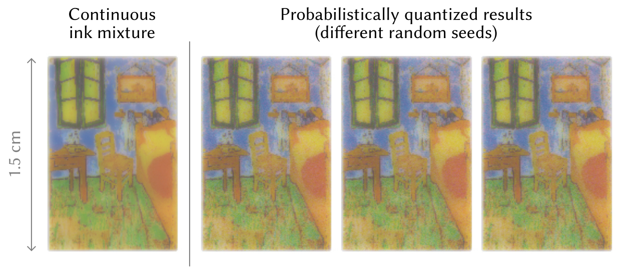

Continuous ink mixture
Probabilistically quantized results (different random seeds)
1.5 cm

Fig. 6. Comparing renderings of a solution with continuous material mixtures (left, 300 DPI) to probabilistically quantized results (right, $600 \times 300 \times 940$ DPI) for a 1 mm planar target. Despite different random seeds, about 70% of the quantized voxels are *identical*; the variance does not cause any substantial visual difference.

material on the surface. A coat of transparent paint establishes the smooth dielectric surface we assume in the optimization.

Inside the prinout, inks from adjacent voxels can flow into each other before UV light is applied to harden them. The resulting mixture of pigments creates a new material, which can be modelled under the assumptions given in Sect. 4.4. Since the physical mixing occours at a higher spatial frequency than we model the volume at, the simulations effectively under-sample the physical printout. Fig. 6 helps quantify the impact of this effect: On the left is a rendering of a continous mixture mapped into absorption/scattering space using Eq. 9 at 300 DPI, simulating perfect physical mixing inside each voxel. The images on the right side of the figure show renderings with voxels of discrete materials, at a higher spatial resolution. No aliasing artifacts are visible.

Assuming an identical phase function for all material also introduces differences between simulation and real printouts, because the phase functions of the actual materials are not identical. Different but optically equivalent mixtures in simulation will have small differences in their appearance when printed. Lastly, using RGB (as opposed to spectral rendering) to handle color throughout the pipeline limits the accuracy of the simulation of the participating medium, introducing color reproduction artifacts. Some effects in participating media can be simulated correctly only by a spectral workflow. One such effect is spectral sharpening, where the repeated application of an absorption spectrum results in a narrowing of its peaks.

Inverse rendered solutions depend on the assumed lighting conditions. This means that under different lighting than those used during optimization, the resulting medium may actually have incorrect appearance. If the display conditions of the printout are known beforehand, this lighting can be "baked" into the optimized result by replicating it in the renderer. Otherwise, assuming hemispherical illumination provides a good default-case.

5 EVALUATION

In the following, we show results obtained using the presented pipeline and compare them to other solutions.

Setup. As both Elek et al. [2017] and Sumin et al. [2019] demonstrated the faithfulness of predictive renderings in comparison to physical printouts, we rely on virtual simulations to evaluate our method. We see this as a viable way, as all our figures depict quantized solutions that could be directly fabricated. For the optical properties of inks, use the measurements by Elek et al. [2017] for the *Vero Rigid Opaque* materials used in the Stratasys J7/J8 3D printer family. Alongside, we designed hypothetical materials (Table 2) to showcase our pipeline's flexibility. For discretization we use a distance-field based voxelization process with uniform 300 DPI resolution as presented in [Sumin et al. 2019]. Our renderer assumes a uniform surface illumination with unity radiance. The sensor is in *texel-space*, i.e., each fragment of the sensor corresponds to exactly one voxel on the surface. Radiative backpropagation is used with the $L_i = 1$ approximation. We initialize the optimization process with a 97% white mixture for all voxels, the remainder is split equally into the other inks to make sure their initial gradients are non-zero. For the final quantization, the resolution is increased to a typical printer resolution of $600 \times 300 \times 940$ DPI. We model the surface of all objects as a smooth dielectric with an refractive index of 1.5.

5.1 Opaque Volume Appearance

We apply our framework to recover the opaque, textured appearance of a scattering 3D print and demonstrate our results in comparison to previous work along this line of research.

Contrast and spatial resolution. Starting with a basic property, we design a test pattern to evaluate edge contrast and lateral scattering compensation. The pattern features black and white stripes of varying widths, with the smallest being one voxel (about 85 µm). As shown in Fig. 7, a naïve placement of black and white ink results in a well-preserved lightness match on larger patches at the cost of color-bleeding around edges. Our solution stays color-neutral with slightly increased lightness deviation in flat regions. Despite variance from multiple sources, our optimization reproduces fine details accurately and balances local contrast, spatial details, and color-bleeding. The cutouts in Fig. 8 enlarge a single step-function from Fig. 7. Upon closer inspection, one can notice that our optimizer decides to mix a substantial amount of colored ink into the black regions, mainly

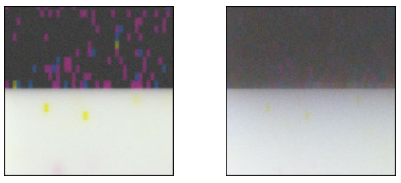


Fig. 8. Rendering of a discretized optimization result for a step function; (a) artificially increased density to show individual voxels, (b) actual appearance.

magenta, and yellow ink into the white region. This counters the slight tints of the white and black materials and ensures the result stays color neutral, unlike the naïvely extruded solution. Such shortcuts are possible with color-table-based approaches [Babaei et al. 2017; Brunton et al. 2015; Elek et al. 2017], but require manual discovery, an explicit specification, and an extension of the mapping. Our method discovers and selectively applies these alternative solutions without intervention, while explicit choice would have to be incorporated in the metric.

Color and structure fidelity. Testing color reproduction, we run the optimization on textured slabs which show a combination of smooth color gradients, hard edges, and fine details. Fig. 9 shows the performance of the proposed method in comparison with the state of the art [Sumin et al. 2019].

In the top image, we observe overall higher saturation for our method. Fine details around the table, chair and painting are slightly more pronounced on the left side. The structure of the floor is however more refined for the previous method.

Looking at the bottom image, they perform better at reproducing the grainyness in the sky. When comparing the challenging case of black-on-white details on the cottage's wall, which have the extend of single voxels, the methods seem to be almost on par. We note, that the dark-orange spots along the path are difficult to reproduce, as also visible in the cutouts in Fig. 1.

Metric weighting. In Sec. 4.5, we proposed a new combined metric for 3D applications that balances color accuracy, local structure, and

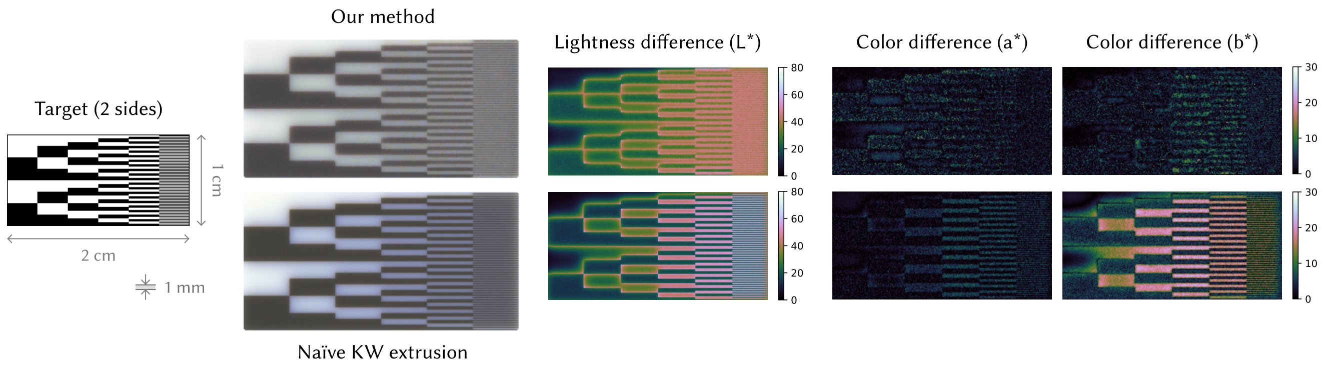


Fig. 7. Testing maximum resolution and color fidelity of a 1 mm thick planar target with a "zebra" test pattern (left). The pattern contains black and white stripes of increasing frequency (stripe heights: 30, 15, 10, 5, 3, and 1 voxels; 1 voxel is about 85um at 300 DPI). Our method (top) reproduces even the tiniest details without any moiré fringes or significant color shifts. A naïve solution (bottom) consisting of black (K) and white (W) inks extruded through the whole slab results in slightly better contrast, but noticeable color shift towards blue and over-darkening of the right part of the slab. To verify, difference images in the CIELAB color space are attached (right), computed as Euclidean distances in L*, a*, and b* separately.

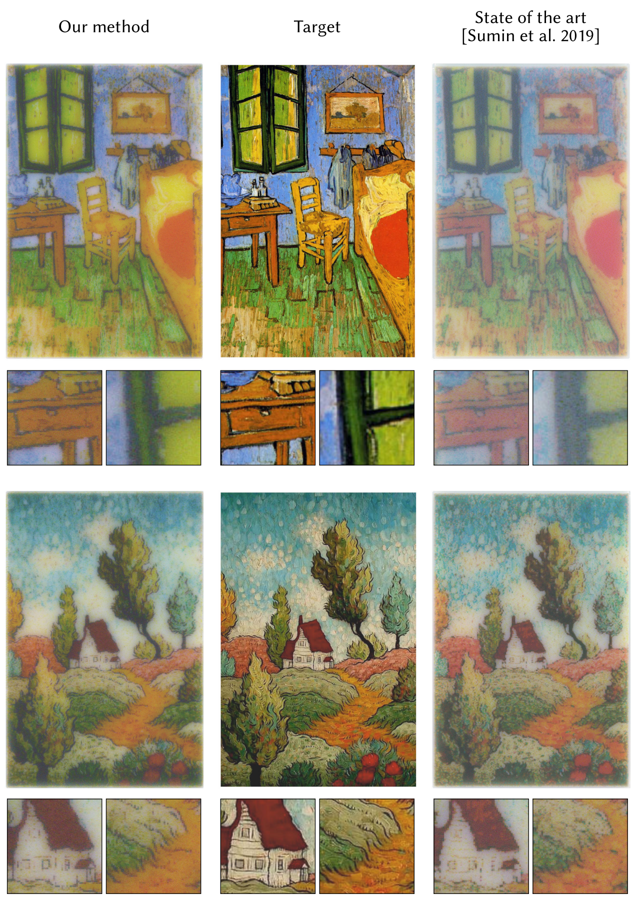


Fig. 9. Comparison of our results (left) to the state of the art (right) [Sumin et al. 2019] for a 2.5D planar slab, which has a different texture on each side. The slab is 30x20 mm large and 2.5 mm thick. **Top:** On the first side of the slab, our method achieves a significantly better color accuracy, saturation, and contrast than the state of the art. **Bottom:** The differences on this side are more subtle: our result overall appears "smoother" and managed to reproduce the dark edges better.

edge sharpness. Using different weighting for the components E_1, E_2, and E_3, one can steer the optimization trajectory. The parameter-space exploration shown in Fig. 19 adjusts the relative weights for the latter two, while keeping $E1$ (color accuracy) constant.

We observe that increasing values of E_2 in the first row leads to higher contrast around edges, as can be seen in the outlines of the chair and bed on the left side. This also leads to desaturation of colors for example on the blue wall.

Increasing the weight for E_3 leads to even stronger contrast, but at the cost of severe hue shifts towards the higher values (bottom row). The combination of all three metrics towards the bottom right inherits the properties of individual components.

Co-optimizing different visual stimuli seems to lead to them influencing one another. This is likely because maximum contrast and best color fidelity are conflicting goals: A black to white step function has maximum contrast, but no color.

Table 3. Summary of optimization results. We use three sets of hardware: ⊡ denotes a NVIDIA RTX 3080 (10 GB) and $*$ means using six materials instead of five. ♮ runs on a NVIDIA TITAN RTX (24 GB). [Sumin et al. 2019] is run on a machine with 20 cores (non-HT) †.

Model	Size	Volume voxels	Surface voxels	Total runtime	Samples	Setup
Orange	35 mm	3.70 M	359 k	215 min	128 spp	⊡∗
Slab	15 mm	0.69 M	118 k	50 min	128 spp	⊡
	30 mm	2.73 M	405 k	424 min	128 spp	⊡
				430 min	128 spp	⊡∗
				342 min	512 spp	†
Cat	60 mm	8.62 M	637 k	1350 min	128 spp	♮
				1429 min	512 spp	†
Nefertiti	40 mm	10.50 M	576 k	1042 min	32 spp	♮
				3071 min	512 spp	†

General 3D geometry. Combining our previous results into a practical setting, we run our framework on several objects with general 3D geometry. Fig. 1 (right) shows them side-by-side to illustrate their scale. The cat model shown in Fig. 11 was selected for its complex curvature, thin features such as the ears and tail, and a highly detailed face texture. A 3D scan of Nefertiti was chosen for its very thin, but high-contrast details such as the eyebrows, eyeliner, and necklace area. Optimization results and a comparison to the state of the art is shown in Fig. 10.

Computational cost. All models shown were run on consumer-grade hardware equipped with either the NVIDIA TITAN RTX (24 GB memory) or 3080 RTX (10 GB memory) GPU. Table 3 details runtimes and setups used for individual models and shows that the total runtimes of our pipeline and the method of Sumin et al. [2019] seem roughly comparable, though a GPU-based implementation of their method would likely shift this intuition in their favor. That already takes into account that we adjusted our learning rate to achieve good convergence within about 150-350 iterations, whereas the heuristic pipeline of Sumin et al. [2019] usually reaches the lowest error in 10 steps.

In Fig. 18, we illustrate that our runtimes scale roughly linearly with respect to model sizes. In Fig. 17, we show how our method can be adjusted in a "time vs. quality" manner, where decreasing the sample counts results in faster runtimes, but lower quality of details.

Initialization. It can be suspected that the optimized objective function contains many local minima, most of them having very similar values. With gradient descent being a local optimization method, convergence to the absolute global minimum cannot be guaranteed, but we can show that even different initializations still converge to an almost identical error value. Results of this initialization study can be found in Figs. 12, 13 and 14 for the 30x20mm slab model. Initializing with results obtained from Sumin et al. [2019]

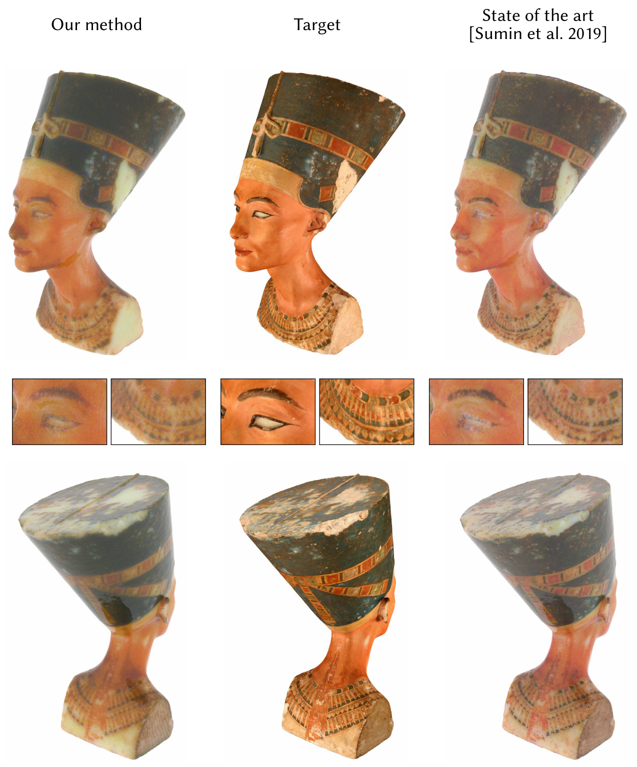


Fig. 10. A 4 cm tall reproduction of a 3D-scanned model of Nefertiti. Difficult areas are the eye region, where our method shows better detail, and the necklace, where Sumin et al. [2019] excels in contrast

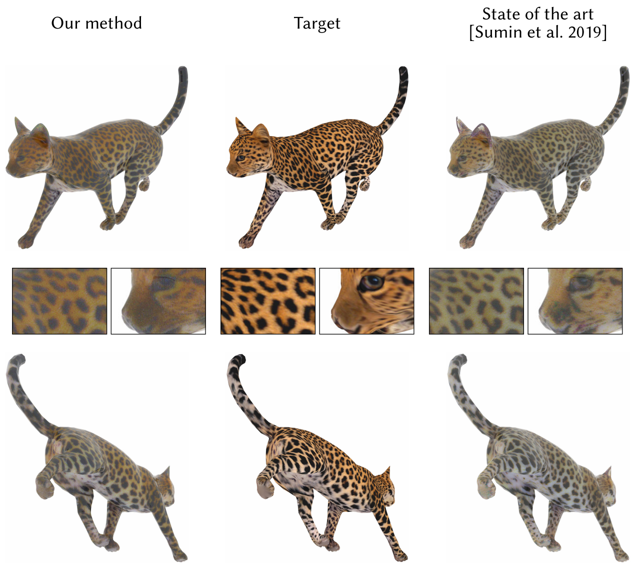


Fig. 11. Cat model: Its orange-brown texture seems to be especially tough to match. Notice that our result is slightly darker, towards brown, while Sumin et al. [2019] are slightly lighter.

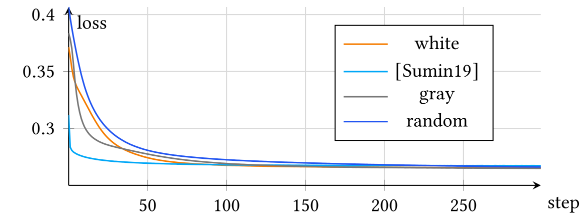


Fig. 12. Convergence plots of different initializations using the 30x20mm slab model.

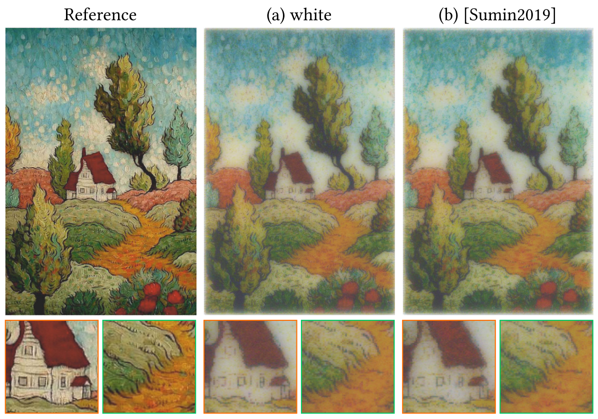


Fig. 13. Comparing different initializations of the 30mm slab model. (a) our default initialization with a 97% white mixture, (b) initialized with the converged result from [Sumin et al. 2019]

greatly accelerates the convergence of the optimization. Without this informed start, the fastest convergence is obtained when initializing with white, while a random mixture needs more iterations to converge. Inspecting the results reveals small differences. Initializing the mixture with random values seems to result in mixtures with less black, suggesting that the optimizer chooses to favor darkening by using a combination of CMY. The resulting images show higher saturation, but lower contrast. Initializing the pipeline with a solution from Sumin et al. [2019] shows no significant differences to the white initialization. We investigated this further by studying differences in L,a,b seperately, and found the biggest delta in the sky lightness in the scene of Fig. 13.

5.2 Applications

A key feature of the proposed pipeline is the ease of tailoring it to different use cases. By changing the behavior of its individual components, the objective of the optimization can be adapted. In the following, we show how two previously complex tasks that required specialized solutions naturally follow from our parametrization.

Extended Ink-set. Introducing a new (specialized) printing material is useful for increased color accuracy and spectral reproduction [Shi et al. 2018], controlling reflectance [Piovarči et al. 2020], and co-optimizing mechanical and optical parameters [Morovič et al. 2019]. In our pipeline, the optical properties of new materials are

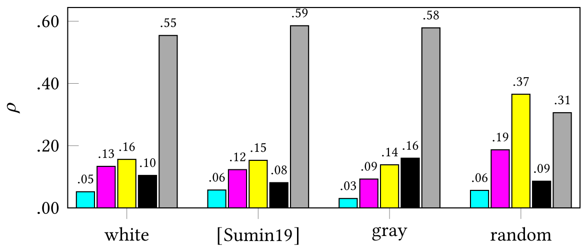


Fig. 14. Average ink usage of the top 8 layers of the lower half of the 30*mm* slab model (Fig. 13), different initializations

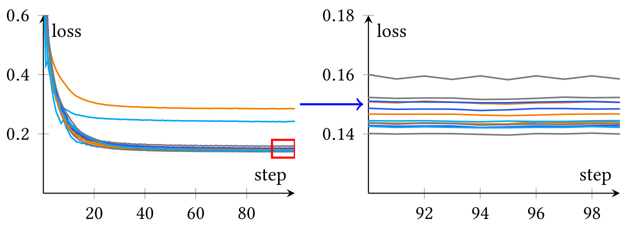


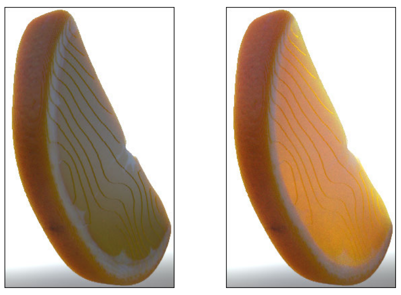
Fig. 15. Material-selection use-case for a printer that can use 6 different materials. Running all possible $\binom{6}{4}$ combinations of the first 6 inks from Tab. 2, plus Black and White, reveals the combination Cyan, Magenta, Magenta2, Yellow having the lowest loss value and thus being the inkset best suited for reproducing the target on the printer.

simply added to the available set with material mixing (Eq. 9) and quantization (Sec. 4.7) adjusted accordingly. An example of this can be seen in Fig. 1, where an additional sixth, orange material is being used to improve the color on the pathway. The associated time penalty is negligible as shown in Table 3. This is due to the fact that rendering and computing the light-transport derivatives always takes place in RGB. The material mixture is handled before the rendering, and after radiative backpropagation, so the associated memory bandwidth impact is very small. Previous methods required expensive rebuilding or measuring of color tables whereas our method adapts to new material combinations on the fly.

Furthermore, because of this flexibility, it becomes feasible to tackle the long-standing *ink selection* problem from 2D printing. This is analogous to hyperparameter-optimization in machine learning where an outer loop is optimizing the meta-parameters of the inner training loop.

The convergence graph in Fig. 15 reveals a certain optimal ink-set for the 15mm slab model shown in Fig. 1. For this proof-of-concept we show a brute-force search but note that more sophisticated strategies (e.g., Shi et al. [2018] and Ansari et al. [2020]) could be transferred from the aforementioned fields in the future.

Translucency. A special case of an extended material set is the addition of transparent ink. Brunton et al. [2018] perform informed, probabilistic replacement of white ink with transparent material to reproduce perceptual transparency cues. In a similar way, we can constrain the optimizer to a use a fixed amount of transparent material in order to enforce translucency in the optimization result.



(a) Opaque (b) Translucent

Fig. 16. Translucency reproduction of an orange slice. (a) shows the optimization results using the opaque material set and a dE76 loss function. In (b) the optimizer is constrained to 80% transparent ink for the pulp.

One such example can be seen in Fig. 16, where the optimizer is constrained to 80% transparent material for the pulp of the orange, and was allowed to choose freely in the peel. This results in a more realistic appearance reproduction compared to the opaque version in Fig. 16a.

Translucent appearance is a perceptively highly complex phenomenon. The human visual system depends on certain visual cues to disambiguate the appearance of different materials, such as color saturation, edge sharpness and the appearance under side- or backlighting [Fleming and Bülthoff 2005; Xiao et al. 2014, 2020].

Future work can build on this and integrate spatially-varying constraints, or perceptive metrics that include translucency, to replicate artist-friendly control akin to Brunton et al. [2018] or directionally-dependent goals as discussed by Zheng et al. [2020].

6 CONCLUSION

Resin-based full-color 3D-printing is capable of creating objects with very complex light transport properties. Voxels deep inside of the volume can have significant influence on the surface appearance of printouts, depending on how the surrounding volume is shaped and configured. Modeling these interdependencies on a global scale can give a degree of control over the results that was previously not attainable. Combining Monte Carlo estimates of light transport gradients from a physically based renderer with a volume parameterization based on the actual printer materials gives such a model. Its application in the context of gradient-descent optimization with a suitable loss function leads to results that come close to what is possible given the physical limits of the printing process.

The flexibility of this approach is illustrated by the fact that problems that previously required specialized solutions turn into straightforward extensions of the same pipeline. The extended inkset, inkset optimization and translucency use-cases have shown that. Our metrics parameters and setting the sampling rate for the MC estimates give intuitive controls over the optimization process on a "color vs. contrast" and "time vs. quality" basis.

The ambiguities that stem from the available materials can actively be exploited by the optimizer to attain a good match, instead of complicating the creation of pre-computed mapping tables. It is intriguing to see that in this highly non-convex solution space, there are fundamentally different solutions that still seem to equally satisfy the complicated quality assessment of human perception, as shown in the comparison of the herein presented method with the previous state of the art.

Further work still remains. The way the human visual system perceives different features proves to be very difficult to distill into a conclusive metric, as the field of image quality- and image-difference-metrics shows. Implementing new discoveries in this field into the pipeline will be easy, as long as they are differentiable.

Furthermore, with the optimization being gradient-based, it only explores a limited subset of the whole solution space. One can suspect that better solutions exist, such as building transparent channels inside the volume that would optically connect areas that are geometrically far from each other. But such complex configurations cannot be found by the optimizer without a good initialization.

ACKNOWLEDGMENTS

We thank Sketchfab users Evgeni Yanev (Orange Slice), C. Yamahata and Thot_htp (Nefertiti) for their 3D models, and the reviewers for their valuable feedback. This work has received funding from GA UK project 1164620 of Charles University, and by the Czech Science Foundation under grant number GAČR-19-07626S. This work was further supported by the Charles University grant SVV-260588.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.

Navid Ansari, Omid Alizadeh-Mousavi, Hans-Peter Seidel, and Vahid Babaei. 2020. Mixed integer ink selection for spectral reproduction. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 39, 6 (Nov. 2020), 255:1–255:16. https://doi.org/10.1145/3414685.3417761

Thomas Auzinger, Wolfgang Heidrich, and Bernd Bickel. 2018. Computational design of nanostructural color for additive manufacturing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37, 4 (July 2018), 159:1–159:16. https://doi.org/10.1145/3197517.3201376

Vahid Babaei, Kiril Vidimče, Michael Foshey, Alexandre Kaspar, Piotr Didyk, and Wojciech Matusik. 2017. Color Contoning for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36, 4 (July 2017), 124:1–124:15. https://doi.org/10.1145/3072959.3073605

Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2017. Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research* 18, 1 (2017), 5595–5637.

Alan Brunton, Can Ates Arikan, Tejas Madan Tanksale, and Philipp Urban. 2018. 3D Printing Spatially Varying Color and Translucency. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37, 4 (July 2018), 157:1–157:13. https://doi.org/10.1145/3197517.3201349

Alan Brunton, Can Ates Arikan, and Philipp Urban. 2015. Pushing the Limits of 3D Color Printing: Error Diffusion with Translucent Materials. *ACM Transactions on Graphics* 35, 1 (Dec. 2015), 4:1–4:13. https://doi.org/10.1145/2832905

Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Towards Learning-based Inverse Subsurface Scattering. In *2020 IEEE International Conference on Computational Photography, ICCP 2020, Saint Louis, MO, USA, April 24-26, 2020.* IEEE, New York, NY, USA, 1–12. https://doi.org/10.1109/ICCP48838.2020.9105209

Guan-Hao Chen, Chun-Ling Yang, Lai-Man Po, and Sheng-Li Xie. 2006. Edge-Based Structural Similarity for Image Quality Assessment. In *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, Vol. 2. IEEE, New York, NY, USA, 933–936.

Yue Dong, Jiaping Wang, Fabio Pellacini, Xin Tong, and Baining Guo. 2010. Fabricating spatially-varying subsurface scattering. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 29, 4 (July 2010), 62:1–62:10. https://doi.org/10.1145/1778765.1778799

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient Projections onto the L1-Ball for Learning in High Dimensions. In *25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 272–279. https://doi.org/10.1145/1390156.1390191

Oskar Elek, Denis Sumin, Ran Zhang, Tim Weyrich, Karol Myszkowski, Bernd Bickel, Alexander Wilkie, and Jaroslav Křivánek. 2017. Scattering-aware Texture Reproduction for 3D Printing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 36, 6 (Nov. 2017), 241:1–241:15. https://doi.org/10.1145/3130800.3130890

Roland W Fleming and Heinrich H Bülthoff. 2005. Low-level image cues in the perception of translucent materials. *ACM Transactions on Applied Perception (TAP)* 2, 3 (2005), 346–382.

J. R. Frisvad, S. A. Jensen, J. S. Madsen, A. Correia, L. Yang, S. K. S. Gregersen, Y. Meuret, and P.-E. Hansen. 2020. Survey of Models for Acquiring the Optical Properties of Translucent Materials. *Computer Graphics Forum* 39, 2 (2020), 729–755. https://doi.org/10.1111/cgf.14023 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14023

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An Evaluation of Computational Imaging Techniques for Heterogeneous Inverse Scattering. In *European Conference on Computer Vision.* Springer, Berlin, Germany, 685–701. https://doi.org/10.1007/978-3-319-46487-9

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 162:1–162:13. https://doi.org/10.1145/2508363.2508377

Miloš Hašan, Martin Fuchs, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. 2010. Physical Reproduction of Materials with Specified Subsurface Scattering. *ACM Trans. Graph.* 29, 4, Article 61 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778798

Roman Hochuli, Samuel Powell, Simon Arridge, and Ben Cox. 2016. Quantitative photoacoustic tomography using forward and adjoint Monte Carlo models of radiance. *Journal of biomedical optics* 21, 12 (2016), 126004.

Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. 2015. Matching Real Fabrics with Micro-Appearance Models. *ACM Trans. Graph.* 35, 1 (2015), 1–1.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)* 37, 6 (July 2018), 125:1–125:12. https://doi.org/10.1145/3306346.3322954

M Ronnier Luo, Guihua Cui, and Bryan Rigg. 2001. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 26, 5 (2001), 340–350.

A. Luongo, V. Falster, M. B. Doest, M. M. Ribo, E. R. Eiriksson, D. B. Pedersen, and J. R. Frisvad. 2020. Microstructure Control in 3D Printing with Digital Light Processing. *Computer Graphics Forum* 39, 1 (2020), 347–359. https://doi.org/10.1111/cgf.13807 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13807

M. Magnor, G. Kindlmann, N. Duric, and C. Hansen. 2004. Constrained inverse volume rendering for planetary nebulae. In *IEEE Visualization 2004.* IEEE, New York, NY, USA, 83–90. https://doi.org/10.1109/VISUAL.2004.18

K McLaren. 1976. XIII - The development of the CIE 1976 (L* a* b*) uniform colour space and colour-difference formula. *Journal of the Society of Dyers and Colourists* 92, 9 (1976), 338–341.

Peter Morovič, Ján Morovič, Ingeborg Tastl, Melanie Gottwals, and Gary Dispoto. 2019. Co-optimization of color and mechanical properties by volumetric voxel control. *Struct Multidisc Optim* 60, 3 (Sept. 2019), 895–908. https://doi.org/10.1007/s00158-019-02240-8

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020), 146:1–146:15. https://doi.org/10.1145/3386569.3392406

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38, 6 (Dec. 2019), 203:1–203:17. https://doi.org/10.1145/3355089.3356498

Isabel Molina Orihuela and Mehran Ebrahimi. 2019. An Efficient Algorithm for Computing the Derivative of Mean Structural Similarity Index Measure. In *Image Analysis and Recognition*, Fakhri Karray, Aurélio Campilho, and Alfred Yu (Eds.). Springer International Publishing, Cham, 55–66.

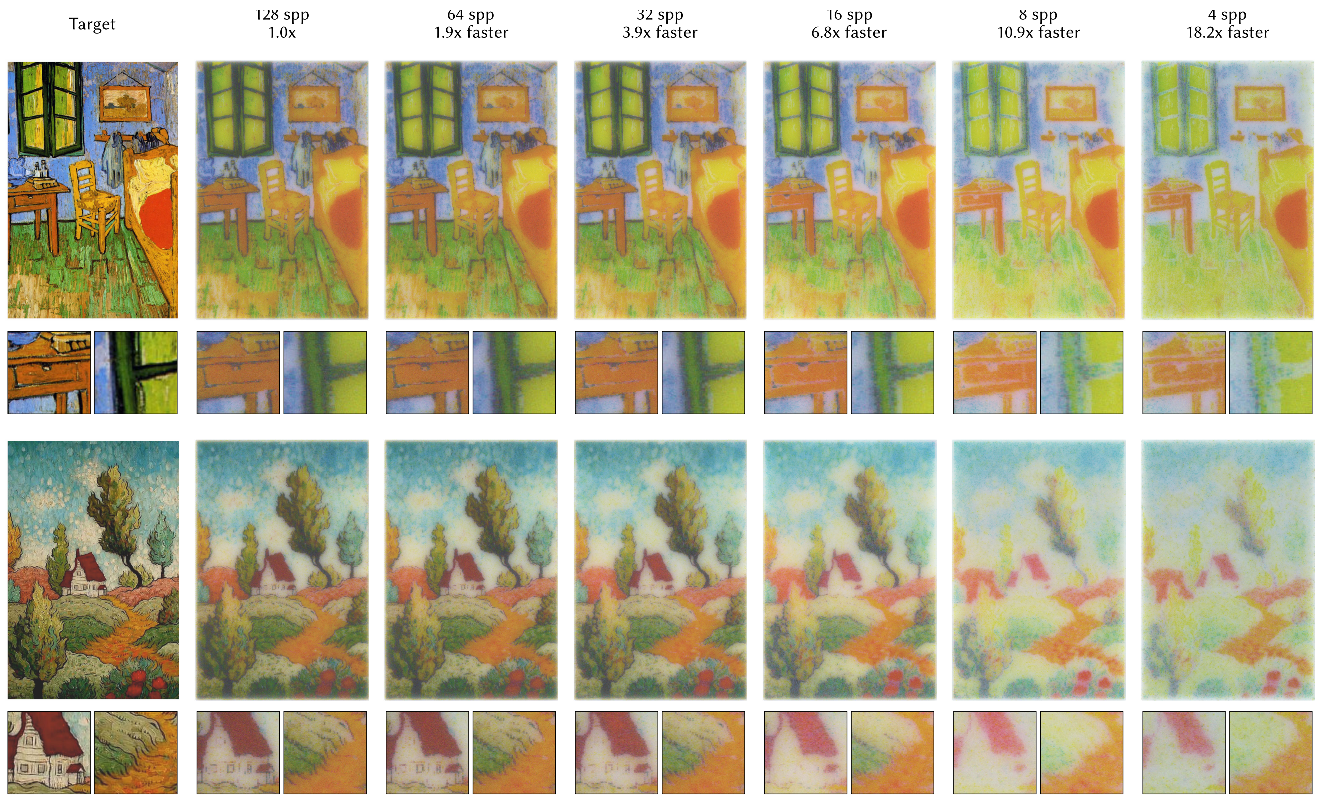


Fig. 17. Illustrating how the convergence can be adjusted in a "time vs. quality" manner. A 30x20x2.5mm double-sided slab, identical to the one in Fig. 9, was optimized with the same learning rate, but different sample counts per texel (spp). The ΔE_{76}-based metric was used for simplicity, so $\gamma_2 = \gamma_3 = 0$. Notice that lowering spp results in proportionally faster iterations, but lose details due to the noise in the Monte Carlo estimates of the derivatives. Despite the 32 spp optimization being almost four times faster, the differences will be almost negligible when looking at the object from further away.

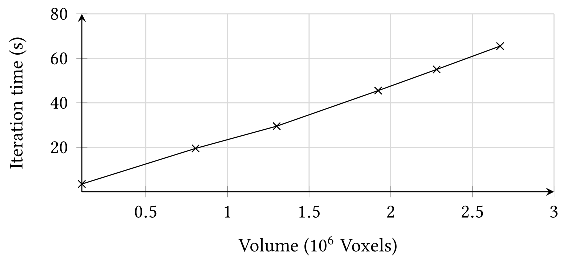


Fig. 18. Time per iteration vs. volume for a spherical model (128 spp, 3080RTX)

Marios Papas, Christian Regg, Wojciech Jarosz, Bernd Bickel, Philip Jackson, Wojciech Matusik, Steve Marschner, and Markus Gross. 2013. Fabricating translucent materials using continuous pigment mixtures. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Michal Piovarči, Michael Foshey, Vahid Babaei, Szymon Rusinkiewicz, Wojciech Matusik, and Piotr Didyk. 2020. Towards spatially varying gloss reproduction for 3D printing. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–13.

Jens Preiss, Felipe Fernandes, and Philipp Urban. 2014. Color-image quality assessment: From prediction to optimization. *IEEE Transactions on Image Processing* 23, 3 (2014), 1366–1378.

Olivier Rouiller, Bernd Bickel, Jan Kautz, Wojciech Matusik, and Marc Alexa. 2013. 3D-printing spatially varying BRDFs. *IEEE computer graphics and applications* 33, 6 (2013), 48–57.

Gaurav Sharma, Wencheng Wu, and Edul N Dalal. 2005. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 30, 1 (2005), 21–30.

Kfir Shem-Tov, Sai Praveen Bangaru, Anat Levin, and Ioannis Gkioulekas. 2020. Towards Reflectometry from Interreflections. In *2020 IEEE International Conference on Computational Photography (ICCP)*. IEEE, New York, NY, USA, 1–12. https://doi.org/10.1109/ICCP48838.2020.9105251

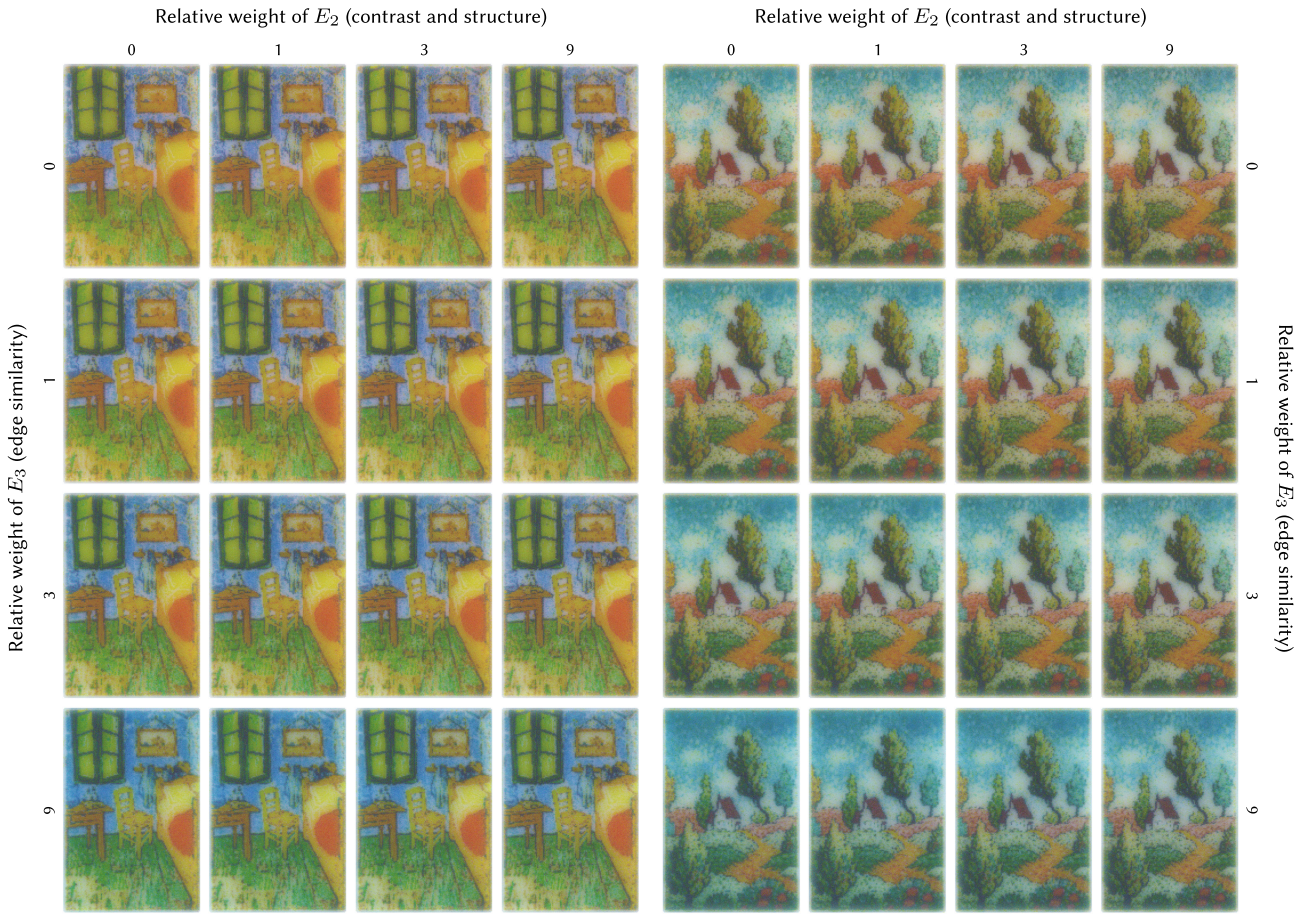
Liang Shi, Vahid Babaei, Changil Kim, Michael Foshey, Yuanming Hu, Pitchaya Sitthi-Amorn, Szymon Rusinkiewicz, and Wojciech Matusik. 2018. Deep multispectral painting reproduction via multi-layer, custom-ink printing. *ACM Trans. Graph.* 37, 6 (Dec. 2018), 1–15. https://doi.org/10.1145/3272127.3275057

Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, Karol Myszkowski, and Tim Weyrich. 2019. Geometry-Aware Scattering Compensation for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 38, 4 (July 2019), 111:1–111:14. https://doi.org/10.1145/3306346.3322992

Philipp Urban, Tejas Madan Tanksale, Alan Brunton, Bui Minh Vu, and Shigeki Nakauchi. 2019. Redefining A in RGBA: Towards a Standard for Graphical 3D Printing. *ACM Trans. Graph.* 38, 3 (June 2019), 21:1–21:14. https://doi.org/10.1145/3319910

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals,*



Relative weight of E_2 (contrast and structure)
0
1
3
9
Relative weight of E_2 (contrast and structure)
0
1
3
9
Relative weight of E_3 (edge similarity)
0
1
3
9
Relative weight of E_3 (edge similarity)
0
1
3
9

Fig. 19

Systems & Computers, 2003, Vol. 2. IEEE, New York, NY, USA, 1398–1402.

Douglas R Wyman, Michael S Patterson, and Brian C Wilson. 1989a. Similarity relations for anisotropic scattering in Monte Carlo simulations of deeply penetrating neutral particles. *J. Comput. Phys.* 81, 1 (1989), 137–150.

Douglas R Wyman, Michael S Patterson, and Brian C Wilson. 1989b. Similarity relations for the interaction parameters in radiation transport. *Applied optics* 28, 24 (1989), 5243–5249.

Bei Xiao, Bruce Walter, Ioannis Gkioulekas, Todd Zickler, Edward Adelson, and Kavita Bala. 2014. Looking against the light: How perception of translucency depends on lighting direction. *Journal of vision* 14, 3 (2014), 17–17.

Bei Xiao, Shuang Zhao, Ioannis Gkioulekas, Wenyan Bi, and Kavita Bala. 2020. Effect of geometric sharpness on translucent material perception. *Journal of vision* 20, 7 (2020), 10–10.

Wufeng Xue, Lei Zhang, Xuanqin Mou, and Alan C Bovik. 2013. Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. *IEEE Transactions on Image Processing* 23, 2 (2013), 684–695.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A differential theory of radiative transfer. *ACM Transactions on Graphics (TOG)* 38, 6 (Nov. 2019), 227:1–227:16. https://doi.org/10.1145/3355089.3356522

Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. 2011. FSIM: A feature similarity index for image quality assessment. *IEEE transactions on Image Processing* 20, 8 (2011), 2378–2386.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. Comp. Vision & Pat. Rec. (CVPR)*. IEEE, New York, NY, USA, 586–595.

Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. 2016a. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging* 3, 1 (2016), 47–57.

Shuang Zhao, Ravi Ramamoorthi, and Kavita Bala. 2014. High-order similarity relations in radiative transfer. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. 2016b. Downsampling scattering parameters for rendering anisotropic media. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–11.

Quan Zheng, Vahid Babaei, Gordon Wetzstein, Hans-Peter Seidel, Matthias Zwicker, and Gurprit Singh. 2020. Neural Light Field 3D Printing. *ACM Trans. Graph.* 39, 6, Article 207 (Nov. 2020), 12 pages. https://doi.org/10.1145/3414685.3417879

3.3 Voxelization and auxiallary data structure

To process a textured mesh into a data structure the optimization pipeline can work on, we first need to convert the geometriy into an isotropic voxel grid, using OpenVDB Museth [2013]. Polyjet printers can have different resolutions across their axis, from which we choose the lowest. Only in the final step, after the optimization, this is up-sampled and halftoned to the native printer resolution. After the volume is voxelized, the volume elements are classified into interior, exterior and surface voxels, respectively. The surface voxels get a target color and a normal vector assigned.

The interior voxels are then processed to create a map to their respective nearest surface point. For each internal voxel, the unit hemisphere is sampled and rays are intersected with the object surface to discover initial connections. These connections are clustered using a weight based on their, and the corresponding surface normal direction. The clusters are then iteratively refined by re-sampling using a von Mises-Fisher distribution and merging distributions using the same weight as for the initial clustering. This results in a surface correspondence mapping that aligns with the surface normal directions.

The sensor employed to sample the surface appearance of an object uses the surface voxels as an origin to shoot rays into the volume along the inverse normal direction.

Conclusion

Differentiable rendering is an exciting and challenging topic in computer graphics. It allows for the use of gradient-based optimization to solve inverse rendering problems that can recover scene parameters from observations. This allows to connect a material model to physical reality. We have explored the recovery of material parameters in two directions:

Appearance fabrication, where a virtual appearance specification is translated into the instructions of a 3D printer. The material-model there is a heterogeneous scattering medium consisting of discrete voxels made of exactly one of several available inks

Appearance matching, where a real-world observation gets translated into a virtual appearance model. The material-model is based on a procedural solid texture that is anatomically informed

The two use-cases do not only differ in their starting- and endpoints, but in several other ways. The optimization tasks differ in how sensitive they are to a good initialization, and both require a suitable material parameterization and loss function to converge gracefully.

The inverse volume rendering task for our appearance fabrication contribution worked very well with a naive initialization. This is likely due to the many minima the solution space contains, most of them with a similar loss value. We parameterized the appearance model using material mixtures, which was necessary to circumvent solving the very challenging integer programming problem the discrete material-to-voxel assignment would otherwise pose. A composite loss function was used, allowing for control over the conflicting goals of best possible contrast and color accuracy.

In contrast, the initialization of our wood appearance matching approach has rigid error bounds that need to be honored for the optimization to converge to good results, or rings will be skipped. The parameterization based on phase-information from complex gabor filters allows for the recovery of a smooth deformation field without foldover, in concert with a simple L2 loss.

Future Challenges

So far, we have only considered diffuse surface reflectance in appearance fabrication, and shown a qualitative proof-of-concept for how to incorporate spatially varying translucency. Embedding translucency quantitatively, so it is measurable with a metric that can then drive the optimizer, is a very difficult problem. Translucency and object shape are related and create perceptual ambiguity. Different illumination conditions add to this, and the human visual system uses several of such cues to disambiguate. Recent success with using neural networks as perceptive loss functions seem a promising cue to warrant exploring a similar approach to measure translucency.

Volumetric light transport simulated in RGB colorspace leads to inaccuracies in prediction, which then also affects derivatives computed by a differentiable renderer. Taking appearance fabrication to a spectral workflow will both alleviate these errors, and pave a way to an even more challenging problem: Some of the available printing materials show fluorescence. Fluorescence adds to the prediction error, and adds another dimension to the ambiguities of volumetric appearance beyond the ones discussed here. This can pose both a challenge and a great opportunity.

We have shown evidence that inverse volumetric light transport works well with naive initializations. There are, however, configurations that in some cases will lead to way better optimized results, which the optimizer simply cannot explore by gradient descent from these starting points. One can imagine, for example, 3D prints that use an internal structure of ”light channels” made from transparent or white voxels, to optically connect otherwise largely separated parts of the object.

Wood exhibits volumetric light transport, especially in veneer-like applications. Light transport there is to a large degree explainable by the multiple changes in index of refraction when light traverses across the elongated wood cells. While modeling of all cells and rendering this using brute force volumetric path tracing is possible, it is also computationally very expensive. Approximating the cellular geometry using a fully anisotropic phase function like SGGX seems like a logical step to replace modeling of the volumetric microgeometry explicitly. The computational cost of full volumetric light transport simulation remains, however. A approximative BSSRDF, akin to the very successful hair reflectance models, can be a building block towards making volumetric wood appearance practical in production. Based on general cell shape, this BSSRDF could be useful to describe light transport in other porous materials, like foam or sponges, and enable appearance matching applications for these as well.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

Navid Ansari, Omid Alizadeh-Mousavi, Hans-Peter Seidel, and Vahid Babaei. Mixed integer ink selection for spectral reproduction. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

Navid Ansari, Hans-Peter Seidel, and Vahid Babaei. Mixed integer neural inverse design. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022.

User "Asdfiel". Snowdonia, wales, 2021. URL `https://forums.flightsimulator.com/t/special-nvidia-prize-weekly-screenshot-challenge-dream-vacation/377224/935`.

Vahid Babaei, Kiril Vidimče, Michael Foshey, Alexandre Kaspar, Piotr Didyk, and Wojciech Matusik. Color Contoning for 3D Printing. *ACM Transactions on Graphics*, 36(4):124:1–124:15, July 2017. ISSN 0730-0301.

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

Sai Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. Systematically differentiating parametric discontinuities. *ACM Trans. Graph.*, 40(107):107:1–107:17, 2021.

Michael C. Bartholomew-Biggs, Steve Brown, Bruce Christianson, and Laurence C. W. Dixon. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, 124:171–190, 2000.

Harold Oliver Beals and Terry Chaffin Davis. Figure in wood: an illustrated review. *Bull Agric Exp Stn Auburn Univ*, 1977.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods.* Academic press, 2014.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *ACM SIGGRAPH*, volume 2012, pages 1–7. vol. 2012, 2012.

Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)*, 1(1):7–24, 1982.

Gustavo Coutinho. Porsche 911 carrera t, 2021. URL https://corona-renderer.com/gallery/23071.

Henri E Cuny, Cyrille BK Rathgeber, David Frank, Patrick Fonti, and Meriem Fournier. Kinetics of tracheid development explain conifer tree-ring structure. *New Phytologist*, 203(4):1231–1241, 2014.

Frank Dellaert and Lin Yen-Chen. Neural volume rendering: Nerf and beyond, 2021.

Eugene d'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM transactions on graphics (TOG)*, 30(4):1–14, 2011.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)*, 37(4):1–15, 2018.

Oskar Elek, Denis Sumin, Ran Zhang, Tim Weyrich, Karol Myszkowski, Bernd Bickel, Alexander Wilkie, and Jaroslav Krivanek. Scattering-aware texture reproduction for 3d printing. *ACM Transactions on Graphics*, 36(6), 2017.

Eli M Gafni and Dimitri P Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22(6): 936–964, 1984.

Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. Gabor noise by example. *ACM Transactions on Graphics (ToG)*, 31(4):1–9, 2012.

Mathieu Galtier, Stéphane Blanco, Jérémi Dauchet, Mouna El Hafi, Vincent Eymet, Richard Fournier, Maxime Roger, Christophe Spiesser, and Guillaume Terrée. Radiative transfer and spectroscopic databases: A line-sampling monte carlo approach. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 172:83–97, 2016. ISSN 0022-4073. Eurotherm Conference No. 105: Computational Thermal Radiation in Participating Media V.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (TOG)*, 32(6):1–13, 2013.

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

Darya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. Brdf representation and acquisition. In *Computer Graphics Forum*, volume 35, pages 625–650. Wiley Online Library, 2016.

Yu Guo, Miloš Hašan, Lingqi Yan, and Shuang Zhao. A bayesian inference framework for procedural material parameter estimation. In *Computer Graphics Forum*, volume 39, pages 255–266. Wiley Online Library, 2020.

Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 165–174, 1993.

Richard C Haskell, Lars O Svaasand, Tsong-Tseh Tsay, Ti-Chen Feng, Matthew S McAdams, and Bruce J Tromberg. Boundary conditions for the diffusion equation in radiative transfer. *JOSA A*, 11(10):2727–2741, 1994.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

Andreas Hänsel. *Holz und Holzwerkstoffe - Prüfung - Struktur - Eigenschaften.* Logos Verlag Berlin GmbH, Berlin, 2014. ISBN 978-3-832-53697-8.

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

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022a.

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, 2022b. https://mitsuba-renderer.org.

Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019.

James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

Alwin Kienle, Cosimo D'Andrea, Florian Foschum, Paola Taroni, and Antonio Pifferi. Light propagation in dry and wet softwood. *Optics Express*, 16(13): 9895–9906, 2008.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ryunosuke Kitamura, Tetsuya Inagaki, and Satoru Tsuchikawa. Determination of true optical absorption and scattering coefficient of wooden cell wall substance by time-of-flight near infrared spectroscopy. *Optics Express*, 24(4):3999–4009, 2016.

Paul Kubelka. New contributions to the optics of intensely light-scattering materials. part i. *Josa*, 38(5):448–457, 1948.

Ares Lagae, Peter Vangorp, Toon Lenaerts, and Philip Dutré. Procedural isotropic stochastic textures by example. *Computers & Graphics*, 34(4): 312–321, 2010.

Maria Larsson, TakashiI Ijiri, Hironori YoshidaA, Johannes Aj Huber, Magnus Fredriksson, Olof Broman, and Takeo Igarashi. Procedural texturing of solid wood with knots. *ACM Transactions on Graphics (TOG)*, 2022.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

Albert Julius Liu, Zhao Dong, Miloš Hašan, and Steve Marschner. Simulating the structure and texture of solid wood. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016.

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

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

Gustav Mie. Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Annalen der physik*, 330(3):377–445, 1908.

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. *Communications of the ACM*, 65(1):99–106, 2021.

Laurence B. Miller. *Monte Carlo analysis of reactivity coefficients in fast reactors general theory and applications.* PhD thesis, Argonne National Lab. (ANL), Argonne, IL (United States), 3 1967.

Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)*, 32(3):1–22, 2013.

Uwe Naumann. Optimal jacobian accumulation is np-complete. *Mathematical Programming*, 112(2):427–441, 2008.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017 Workshop Autodiff Submission*, 2017.

Scott A Prahl, Martin JC van Gemert, and Ashley J Welch. Determining the optical properties of turbid media by using the adding–doubling method. *Applied optics*, 32(4):559–568, 1993.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Craig Schroeder. Practical course on computing derivatives in code. In *ACM SIGGRAPH 2022 Courses*, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393621.

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: differentiable material graphs for procedural material capture. *ACM Transactions on Graphics*, 2020.

Hiroyuki Sugimoto, Sakiko Kawabuchi, Masatoshi Sugimori, and Joseph Gril. Reflection and transmission of visible light by sugi wood: effects of cellular structure and densification. *Journal of Wood Science*, 64(6):738–744, 2018.

Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, J KŖivánek, Karol Myszkowski, and Tim Weyrich. Geometry-aware scattering compensation for 3d printing. *ACM Transactions on Graphics*, 38(4), 2019.

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

John K Vennard. *Elementary fluid mechanics.* Read Books Ltd, Alcester, England, March 2011. ISBN 9781406700107.

Delio Vicini, Sébastien 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.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4):125:1–125:18, July 2022.

ScreenAge 3D Visualization. The amazing story behind the 3d renderings in ikea catalogues, 2017. URL `https://www.screenage.com.au/ikea-catalogue-3d-renderings/`.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Robert Edwin Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

E Woodcock, T Murphy, P Hemmings, and S Longworth. Techniques used in the gem code for monte carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, volume 557, 1965.

Douglas R Wyman, Michael S Patterson, and Brian C Wilson. Similarity relations for the interaction parameters in radiation transport. *Applied optics*, 28(24): 5243–5249, 1989.

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. Monte carlo estimators for differential light transport. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021.

Shuang Zhao, Ravi Ramamoorthi, and Kavita Bala. High-order similarity relations in radiative transfer. *ACM Trans. Graph.*, 33(4), jul 2014. ISSN 0730-0301.

List of Figures

List of Tables

List of Abbreviations and Symbols

$\partial_\pi x, \frac{\partial x}{\partial \pi}$	Partial derivative of x with respect to π
$\mathbb{S}, \mathbb{H}$	Unit sphere, unit hemisphere
$\mathbb{R}$	set of rational numbers
$\hat{X}$	random variable
$p(x)$	probability density function for x
$\hat{n}$	Surface normal vector
$\bar{x}$	Light path
Ω	Path space, set of all possible paths through a scene
$\mu(\bar{x})$	measure in path space
$L(\omega, x)$	Radiance at x from ω
σ_t	volumetric density
σ_a	volumetric absorption coefficient
σ_s	volumetric scattering coefficient
α	volumetric single scattering albedo
C	(apparent) color
BSDF	Bidirectional scattering distribution function
BRDF	Bidirectional reflectance distribution function
BSSRDF	Bidirectional scattering surface reflectance distribution function

List of publications

- Denis Sumin, Tobias Rittig, Vahid Babaei, **Thomas Nindel**, Alexander Wilkie, Piotr Dydik, Bernd Bickel, Jaroslav Křivánek, Karol Myszkowski, Tim Weyrich. Geometry-Aware Scattering Compensation for 3D Printing. In *ACM Transactions on Graphics, July 2019*
- **Thomas K. Nindel**, Tomáš Iser, Tobias Rittig, Alexander Wilkie, Jaroslav Křivánek: A Gradient-Based Framework for 3D Print Appearance Optimization. In *ACM Transactions on Graphics, August 2022*
- **Thomas K. Nindel**, Mohcen Hafidi, Tomáš Iser, and Alexander Wilkie: Automatic inference of a anatomically meaningful solid wood texture from a single photograph. Submitted to *Eurographics 2023, Journal Track, Reviews pending*
- Tomáš Iser, Tobias Rittig, Emilie Nogué, **Thomas Nindel**, Alexander Wilkie: Affordable Spectral Measurements of Translucent Materials. In *ACM Transactions on Graphics, December 2022*

A. Attachments

A browser-based implementation of the volumetric path tracer sketched in Figure 1.11, that also allows interacting with its source code, can be found at `https://www.shadertoy.com/view/tlBXWW`