From Colors to Spectra and Back Again

Concepts and Methods for Practical Spectral Rendering

van de Ruit, M.

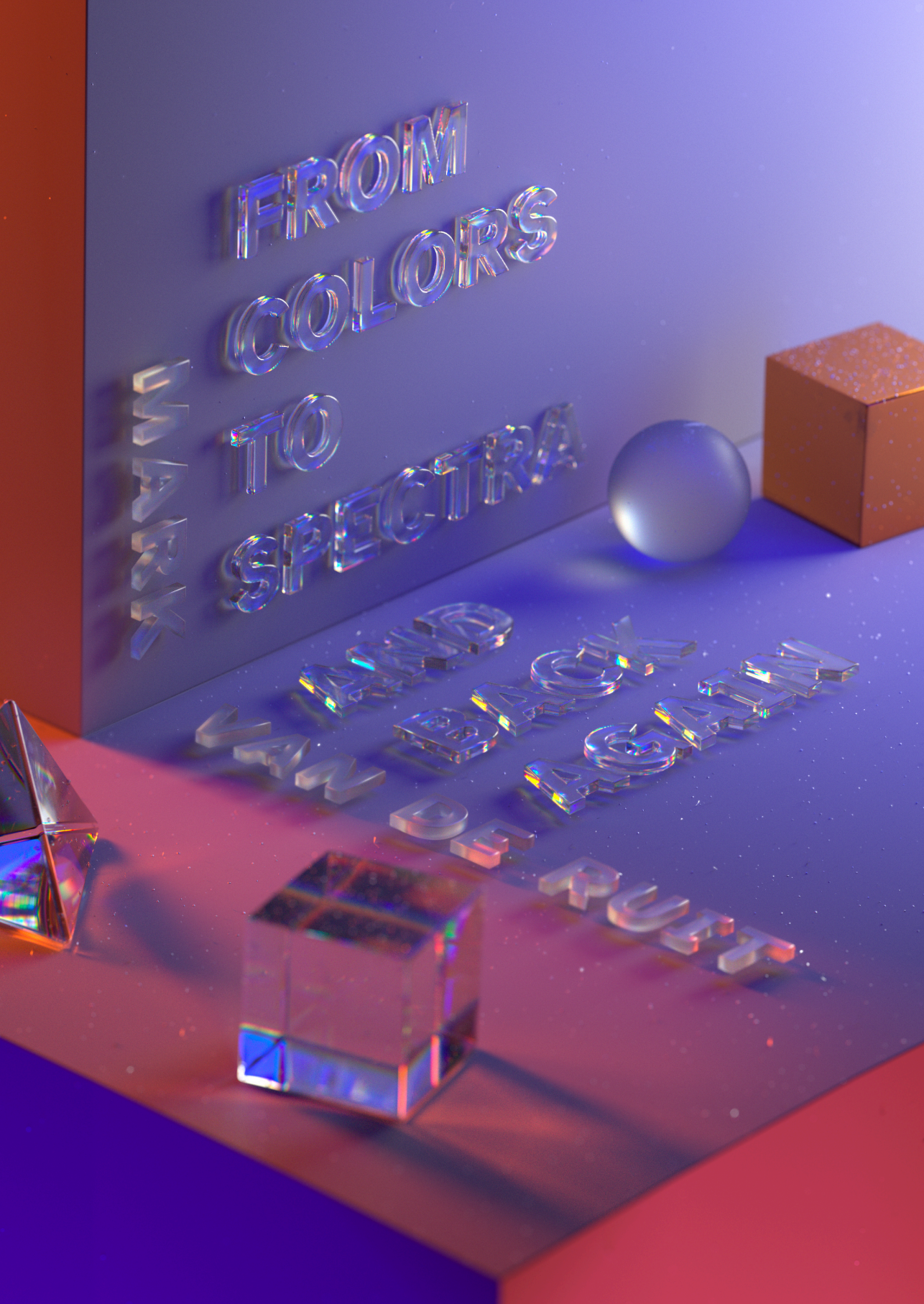
DOI
10.4233/uuid:a92f609c-295c-422f-9317-38fd47c08c66

Publication date
2025

Document Version
Final published version

Citation (APA)
van de Ruit, M. (2025). *From Colors to Spectra and Back Again: Concepts and Methods for Practical Spectral Rendering*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:a92f609c-295c-422f-9317-38fd47c08c66

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.


FROM
COLORS
TO
SPECTRA
WARX
AND
BACK
AGAIN
CAN BE PUT

From Colors to Spectra and Back Again

Concepts and Methods for Practical Spectral Rendering

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on Thursday 9 October 2025 at 15:00.

by

Mark VAN DE RUIT

Master of Science in Computer Science, Delft University of Technology, the Netherlands
born in Bergen op Zoom, the Netherlands.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. E. Eisemann,	Delft University of Technology, promotor
Dr. R. Marroquim,	Delft University of Technology, copromotor
Independent members:	
Dr. T. Boubekeur,	Adobe
Prof. dr.-ing. H.P.A. Lensch,	University of Tübingen
Prof. dr. S.C. Pont,	Delft University of Technology
Prof. dr. H.P. Urbach,	Delft University of Technology
Prof. dr. C.M. Jonker,	Delft University of Technology, reserve member

Keywords: Computer Graphics, Spectral Rendering, Metamerism, t-SNE

Printed by: Proefschriftenprinten.nl

With magic, you can turn a frog into a prince. With science, you can turn a frog into a PhD and you still have the frog you started with.

Sir Terry Pratchett

Contents

Summary

3D rendering is traditionally based on a tristimulus approximation, where all light, color, and spectral distributions are represented using three (RGB) values. For enhanced physical accuracy, spectral rendering algorithms can be employed. However, these methods are typically more computationally expensive and require scene and material data measured from the real world. With few exceptions, spectral rendering remains confined to academic research, with limited adoption in production pipelines due to the many challenges it poses. In this dissertation, we identify several of these challenges and propose practical solutions to each.

Chapter 2 addresses the problem of spectral wavelength sampling. Spectral Monte Carlo rendering tends to converge more slowly than tristimulus methods, due to the need to sample across the spectral domain. We propose a multi-pass approach: in an initial low-resolution render, we estimate camera-incident spectral radiance distributions. These estimates are then used to importance sample the spectral domain during the main render. This approach reduces sample rates in regions with high spectral variance, with little computational overhead.

Chapter 3 focuses on the challenge of color-to-spectrum uplifting. Since most available scene/material data is RGB-only, it is common to apply spectral uplifting to reconstruct plausible spectra. Traditional methods are typically deterministic, when in reality there is no one-to-one mapping. We propose an artist-controllable uplifting scheme that allows users to specify texture appearance under varying lighting and observer conditions. Our method formulates uplifting as a constrained optimization problem, using an interpolation scheme and data-driven reflectance generation to ensure physical plausibility. The resulting solution is lightweight, integrates easily into existing workflows, and provides intuitive control.

Chapter 4 revisits the uplifting problem beyond simple direct illumination. Our earlier method assumes appearance design under direct lighting only, making it difficult to control scene appearance under indirect illumination. We introduce a more refined uplifting scheme that incorporates spectral constraints applied to scene surfaces. These constraints guide spectral appearance even under light transport. Our method supports a more flexible authoring process, solves for uplifted spectra efficiently, and maintains a smaller memory footprint during rendering by using a compact spectral texture representation.

Appendix A focuses on the t-distributed stochastic neighbor embedding (t-SNE) algorithm. This non-linear form of dimensionality reduction is standard in exploratory data analysis, despite its high runtime and memory costs. To enable future data exploration methods in the context of spectral material design, we propose a novel t-SNE minimization. Our method constructs a pair of spatial hierarchies over embedding data, which are simultaneously traversed to approximate many computations. We demonstrate a significant performance increase over the state-of-the-art on a variety of datasets.

Samenvatting

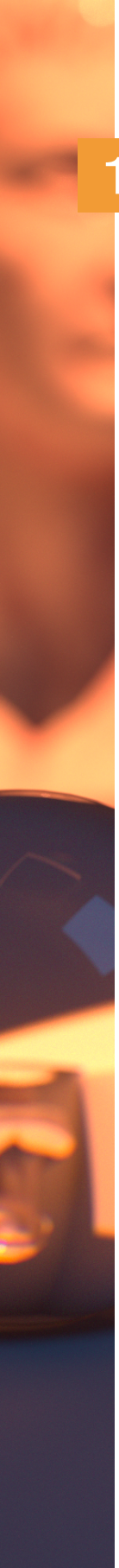
3D-rendering is veelal gebaseerd op een tristimulusbenadering, waarbij al het licht, kleuren en spectrale waardes worden gerepresenteerd met drie (RGB) getallen. Voor fysieke nauwkeurigheid kunnen spectral rendering-algoritmen worden ingezet. Deze zijn echter computationeel duurder en vereisen gemeten materiaaleigenschappen. Hierdoor blijft spectral rendering voornamelijk beperkt tot academisch onderzoek, met beperkte actuele toepassingen vanwege de vele problemen die het met zich meebrengt. In dit proefschrift identificeren we een aantal van deze problemen en stellen we praktische oplossingen voor.

Hoofdstuk 2 behandelt het probleem van spectral wavelength sampling. Spectral Monte Carlo-rendering convergeert meestal trager dan tristimulusmethoden, vanwege de noodzaak om het spectrale domein te sampelen. We stellen een multipass-benadering voor: in een initiële render bouwen we spectrale schattingen vanaf de camera. Deze schattingen worden vervolgens gebruikt om het spectrale domein gestuurd te samplen tijdens de echte render. Onze aanpak verlaagt de benodigde samples in gebieden met hoge variatie, met minimale rekenkosten.

Hoofdstuk 3 richt zich op het omzetten van kleuren naar spectra (spectral uplifting). Aangezien de meeste materiaaleigenschappen enkel in RGB-formaat beschikbaar zijn, is het gebruikelijk om uplifting toe te passen om plausibele spectra te reconstrueren. Traditionele methoden zijn doorgaans deterministisch, terwijl er in werkelijkheid geen eenduidige oplossing bestaat. Wij stellen een stuurbare upliftingmethode voor, waarmee gebruikers het uiterlijk van texturen kunnen specificeren onder verschillende lichtomstandigheden. Onze methode formuleert uplifting als een optimalizatieprobleem, waarbij een data-gedreven aanpak worden gebruikt om fysieke plausibiliteit te waarborgen. De uiteindelijke oplossing is snel, eenvoudig te integreren in bestaande methods, en biedt intuïtieve controle.

Hoofdstuk 4 herbekijkt dit upliftingprobleem buiten louter directe belichting. Onze eerdere methode gaat uit van directe verlichting, wat het moeilijk maakt om het uiterlijk van scènes te beheersen onder indirecte verlichting. We introduceren een verfijndere methode die spectrale instellingen op scène-oppervlakken toepast. Deze instellingen sturen het spectrale uiterlijk ook tijdens lichttransport. Onze methode ondersteunt een flexibeler ontwerpproces, lost efficiënt op, en behoudt een klein geheugengebruik tijdens rendering dankzij een compacte representatie.

Bijlage A richt zich op het t-distributed stochastic neighbor embedding (t-SNE) algoritme. Deze niet-lineaire dimensiereductie is standaard in data-analyse, ondanks de hoge reken- en geheugenkosten. Om toekomstige methoden voor data-exploratie in de context van spectraal materiaalontwerp mogelijk te maken, stellen we een nieuwe t-SNE-minimalisatie voor. Onze methode bouwt een paar hiërarchieën op over de data, die gelijktijdig worden doorlopen om veel berekeningen te benaderen. We tonen een aanzienlijke prestatieverbetering aan ten opzichte van huidige methodes op een verscheidenheid aan datasets.



1 Introduction

On May 17th, 1861, James Clerk Maxwell gave an evening lecture on color theory at the Royal Institution in London [1]. Like many of his contemporaries, he had developed an interest in physiology and color perception. In this area, he furthered the work of Thomas Young, who postulated that human vision was principally founded on three types of receptors [2, 3]. Before them, even Isaac Newton had been fascinated by color, writing in *Opticks* [4] in his own curious manner:

> *I speak here of Colours so far as they arise from Light. For they appear sometimes by other Causes, as when by the power of Phantasy we see Colours in a Dream, or a Mad-man sees things before him which are not there [...]*

To illustrate the developing theories, Maxwell - remembered for far greater contributions to science than this minor evening lecture - demonstrated what we now understand to be the first color photograph.

For this, he enlisted the help of the photographer Thomas Sutton, who laboriously created three black-and-white negatives of *"a bow made of ribbon, striped with various colours"*, exposed separately through red, green, and blue filters. During his lecture, Maxwell projected these negatives through the same filters, superimposing the images atop one another. While the process was cumbersome, the resulting projection reproduced the ribbon and its intricate patterns to some extent. As this should not have worked with the techniques of that time, the production's details led to some debate [5]. Nevertheless, there is little doubt that Maxwell's photograph aptly demonstrated the foundational principle of color primaries.

Human color vision is not, as even Newton sometimes postulated, sourced from some physical constitution of light. It is inherent to the physiology of the visual system, beginning with the stimulation of three photo-receptors, each having different but somewhat overlapping sensitivities to the spectral distribution of light [2]. Consequently, almost all visible colors can be reproduced by mixing an appropriate set of three lights - three primaries - in the right amounts. In Maxwell's demonstrative photograph, these were additive amounts of red, green, and blue-filtered light. In a much later letter to William Thomson [6], he would write:

> *Colour as perceived by us is a function of three independent variables [...] at least three are I think sufficient, but time will show if I thrive.*

Through Maxwell's contributions, and those of Young, von Helmholtz, and many others, the *tristimulus* theory of color indeed thrived, remaining established into the current era [7]. We now understand that light and color are disconnected concepts, linked by the *Young-Helmholtz trichromatic theory*. Modern color science is founded on this theory exactly because most humans are trichromats.

The reader may be familiar with these notions even if they are not a color scientist. Describing color in terms of primaries is foundational to many other fields: dye mixing, graphic design, printing, and the myriad of electronic displays that surround us. Computer-generated images are likewise often represented using three channels. This, inevitably, brings us to the topic of rendering.

Rendering is the process of computationally generating ”photographs” from a virtual scene description, and serves as the technological backing to many movies, video games, and simulations. In rendering, the notion of trichromacy has been somewhat extended beyond the scope of color representation. It is standard to encode not just rendered outputs, but scene inputs such as material and illuminant properties, in terms of color. Further, many computations pertaining to high-dimensional physical quantities are approximated using three values. This *tristimulus rendering* is known to be insufficient for physically accurate rendering [8]. Spectral phenomena such as fluorescence are challenging to implement [9, 10], while certain physiological color phenomena remain entirely unaccounted for.

Notably, *color metamerism* - different spectral distributions resulting in the same color signal in some circumstances - cannot currently be expressed in tristimulus rendering. In his *Experiments on Colour* [1], Maxwell encounters this phenomenon:

> *That these experiments are really evidence relating to the constitution of the eye, and not mere comparisons of two things which are in themselves identical, may be shown by observing these resultant tints through coloured glasses, or by using gas-light instead of day-light. The tints which before appeared identical will now be manifestly different [...]*

And yet, this deliberate approximation has enabled rendering to mature at a lower computational cost. Expectedly, modern industry pipelines and workflows operate almost exclusively on the approximation of tristimulus rendering. More accurate *spectral rendering* models forego this assumption, but these are more computationally expensive and can necessitate scene/material properties that are measured from the real world. Spectral material and illuminant data is laborious to capture, and increases the memory consumption of material models. With few exceptions, spectral approaches therefore remain the domain of academic research and the rare predictive renderer.

However, production rendering is increasingly physically-based. There is a notable demand for virtual cloning; the accurate matching of real and digital objects and footage [11]. This merging of the physical and the virtual necessitates the virtual to become physically accurate, which lies beyond the scope of current tristimulus approaches. As such, the path forward appears to forego this approximation, if we can suitably overcome the many workflow incompatibilities and production challenges faced when switching from tristimulus to spectral rendering.

To this effect, we reach the core of this dissertation; to deconstruct several of these challenges, and to propose workable solutions to each. In the following, we detail the topics addressed in this body of work.

1.1 Spectral wavelength sampling

Spectral rendering methods generally extend existing Monte Carlo light transport algorithms. Such algorithms gather the illumination contributing to a point or area - such as a camera pixel - by randomly sampling paths of light scattered through a scene which gather at this area. In the limit, we consider all contributing paths. With finite samples, we can estimate a plausible image by applying this process to all pixels. Estimator error translates to noise in the image.

Spectral renderers sample, in addition to paths, the domain of visible wavelengths. In effect, we evaluate each path for a randomly selected wavelength, instead of the traditional three values. This increases the necessary sample rates for a low-error image, which prior work improves through wavelength clusters [12–14] sampled from relevant distributions [14, 15].

In Chapter 2, we propose a multi-pass approach. During a first low-resolution render, we build coarse estimates of camera-incident distributions of spectral radiance. Then, during the actual render, we use these distributions to importance sample the spectral domain, favoring selection of wavelengths that contribute more to the output image. Hereby, we reduce sample rates in areas of spectral variance, with little computational overhead.

1.2 Color-to-spectrum uplifting

Spectral rendering necessitates spectral scene data as inputs, representing physical quantities. As spectral material capture remains a laborious process - and most production pipelines favor color data either way - little tooling exists for artists to leverage such data. It is common to simply sidestep this issue and uplift existing color data to an appropriate spectral representation, particularly surface reflectance data. This is an ill-posed problem, as a color can stem from any number of metameric reflectances. Most solutions establish 1-to-1 mappings of colors to spectra, opting for the smoothest bounded function as a correct output [16–21].

In Chapter 3, we explore the limitations of this mapping, as it eliminates the under-constrained color behavior of the real world, where objects do unexpectedly mismatch due to metamerism. We propose a controllable uplifting scheme, using a fitted convex polytope that encloses a set of to-be-uplifted color inputs. We then solve for spectra associated with the polytope's vertices, and employ generalized barycentrics to transfer these spectra to the interior colors. As we control selection of metamers for the vertex spectra, we can modify the uplifting's exact behavior.

In Chapter 4, we refine this concept, foregoing a fitted polytope for a larger color system boundary, whose interior we tessellate. We associate controllable spectra with vertices inserted in this tessellation, and thereby obtain more fine-grained control over the behavior of our uplifting. Using a small basis, we show that our method even provides a compact representation for acquired spectral texture data.

1.3 Color-to-spectrum control

Accompanying controllable models for spectral uplifting, we explore constraint types that artists can specify. These enable us to solve for the target spectra, which in turn control these models. Our focus thus shifts between establishing 1-to-n spectral upliftings and specifying the resulting metameric behavior.

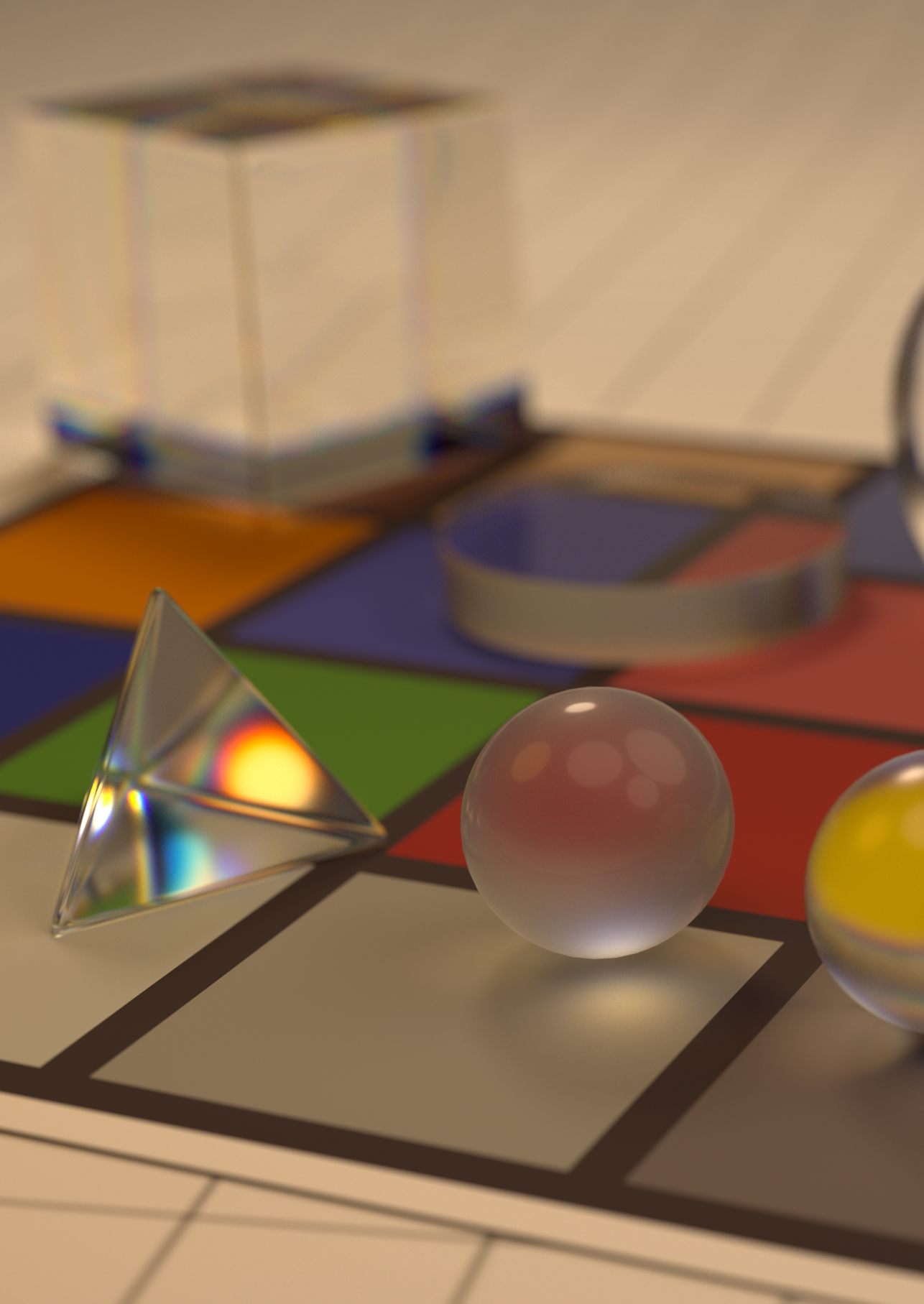
In Chapter 3, we define constraints in terms of illuminant- or observer-induced mismatching in a simplified linear color system. We enable artists to specify color behavior under such mismatches, and generate spectra matching this behavior. To validate inputs, we employ prior work on metamer mismatch boundaries [22].

In Chapter 4, we additionally consider illuminant-induced mismatching from the perspective of a surface point affected by indirect illumination in a scene. For such a point, we estimate incident irradiance to solve for a metamer mismatch boundary describing the uplifted non-linear color behavior. Finally, this allows us to arbitrarily place markers on scene surfaces and control the uplifted behavior with respect to the indirect illumination.

1.4 Bridging tristimulus and spectral workflows

A recurring theme throughout this dissertation is the set of differences and disconnects between tristimulus and spectral rendering workflows. In the spectral part, the complexity of scene inputs can hamper artistic control, yet also enables physical accuracy. With tooling such as controllable color-to-spectrum uplifting, we may be able to remove this disconnect. In effect, by carrying over the practical approaches available in tristimulus workflows, we can address its limitations.

In the following chapters, we detail our efforts. In each chapter, we describe a particular problem, explore the surrounding body of work, propose a practical solution, and evaluate said solution's efficacy compared to existing priors. Finally, in Chapter 5, we reflect on our efforts and provide a perspective on the future.



2 Improved Spectral Sampling using Multiple Passes

Abstract

Spectral Monte Carlo rendering simulates advanced light phenomena such as dispersion, but typically shows a slow convergence behavior. Properly sampling the spectral domain is challenging in scenes with complex spectral distributions.

In this chapter⋆, we propose a multi-pass approach. We generate coarse screen-space estimates of incident spectral radiance, and use these to then importance sample the spectral domain. Hereby, we lower variance and reduce noise with little overhead. Our method handles challenging scenarios with difficult spectral distributions, multiple emitters, and participating media. Finally, it can be integrated into existing spectral rendering pipelines for an additional acceleration.

⋆ This chapter is based on "*A Multi-Pass Method for Accelerated Spectral Sampling*", previously published in Computer Graphics Forum (Pacific Graphics 2021) [23].

2.1 Introduction

When producing photorealistic imagery, modern rendering systems typically employ advanced Monte Carlo light transport algorithms. Many of these systems are *trichromatic*, modeling all light, color, and different spectral distributions as a combination of three (RGB) values. This trichromatic approximation is known to be insufficient for accurate color reproduction [8]. Further, it makes it profoundly difficult to simulate physical phenomena such as chromatic light dispersion, diffraction, fluorescence, and polarization. Although extended spectral light transport algorithms have overcome these limitations, these are far more computationally expensive. They necessitate sampling of the spectral domain, significantly increasing the sample rates required to avoid noise.

In recent years, techniques decreased spectral noise by either tackling specific problems such as path-reuse during wavelength-dependent scattering [12–15], or through the application of novel rendering techniques [24]. Despite major improvements, modern spectral renderers may still converge poorly, which is partially attributed to the additional sampling of the spectral domain. Spectral sampling is generally done uniformly or with respect to sensor responses [12]. This increases variance when the observed radiance becomes highly non-uniform due to a multitude of complicated emission and reflectance spectra. Although *spectral power distributions* (SPDs) can be leveraged for importance sampling [14, 15], we show that this is not an optimal solution for many but the simplest scenarios.

Our contribution consists of an extended light transport algorithm where, before rendering, we invest time to build coarse screen-space estimates of incident spectral radiance distributions, and sample these distributions in a manner which avoids bias. This allows us to improve convergence behavior in complicated scenes with many different non-uniform spectra. We extend a unidirectional path tracer and show that our method improves performance where others may currently fall short. We additionally demonstrate a combination with *continuous multiple importance sampling* (CMIS) [14], leading to further improvements.

After covering notations and the state-of-the-art (Section 2.2), we expand on the components of our method (Section 2.3). We next discuss our implementation (Section 2.4) and evaluate its efficacy (Section 2.5) before concluding (Section 2.6).

2.2 Background

Spectral light transport Physically-based renderers are concerned with evaluating the *light transport equation* [25, 26], for which we use the path-integral formulation [27]. We measure the spectral radiance I entering a single pixel j as

Eq. 2.1
$$I_j = \int_\Lambda \int_\Omega f_j(\bar{x}, \lambda)\, d\mu(\bar{x})\, d\lambda,$$

where Λ denotes the *spectral domain* of wavelengths and Ω the *path space* of light transport paths $\bar{x} = x_0, \ldots, x_{n-1}$ of finite lengths n along which light viably travels from a light source to our sensor. The throughput for a single path and a given wavelength is then measured by $f_j(\bar{x}, \lambda)$. While this is challenging to solve directly, we can apply Monte Carlo integration to form the estimator

Eq 2.2
$$\widehat{I}_j = \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\bar{x}_i, \lambda_i)}{p(\bar{x}_i, \lambda_i)},$$

which converges as $N \to \infty$. Here $p(\bar{x}_i, \lambda_i)$ describes a *probability density function* (PDF) for the sampling of combined path-wavelength pairs, decomposed as

Eq 2.3
$$p(\bar{x}, \lambda) = p(\lambda) \cdot p(\bar{x} \mid \lambda).$$

For a uniform distribution, the convergence rate of this estimator resembles $\mathcal{O}(N^{-1/2})$. If a distribution is similar in shape to the integrand, variance may be reduced as samples are focused on places of interest — known as *importance sampling*. If a distribution differs significantly from the integrand, a slower convergence is likely.

The wavelength sampling distribution $p(\lambda)$ can be efficiently constructed as the product of a sensor response p_s and another distribution p_e, i.e. $p(\lambda) = p_s(\lambda) \cdot p_e(\lambda)$. The latter distribution is typically uniform or, preferably, proportional to emission in a scene. Evans and McCool [15] propose selecting a random emitter in the scene to obtain p_e. More recently, West et al. [14] applied a mixture of a scene's emission spectra for their technique.

Multiple wavelength sampling Evans and McCool [15] first noted that Equation 2.2 evaluates a single wavelength per light transport path, and proposed to propagate wavelength clusters until wavelength-dependency occurs, at which point all wavelengths but one are discarded to prevent exponential path growth. This was extended by Radziszewski et al. [12] to propagate multiple wavelengths along a single path in the case of non-specular dispersive scattering, and was formalized by Wilkie et al. [13] with *hero wavelength spectral sampling* (HWSS). The authors select a hero wavelength λ_h for which they compute a light transport path. A set of C wavelengths is then stratified across the spectrum using a rotation function

Eq 2.4
$$\lambda_s = (\lambda_h - \lambda_{min} + \frac{s}{C}\bar{\lambda}) \ mod \ \bar{\lambda} + \lambda_{min}.$$

where $\lambda_{min}, \lambda_{max}$ are the bounds of the spectral range, and $\bar{\lambda} = \lambda_{max} - \lambda_{min}$. These wavelengths are measured across the same path, and results are combined using *multiple importance sampling* [27] (MIS), leading to the following estimator:

Eq 2.5
$$\widehat{I}_j = \frac{1}{N} \sum_{i=1}^{N} \sum_{s=1}^{C} \frac{f_j(\bar{x}_i, \lambda_i^s)}{\sum_{k=1}^{C} p(\bar{x}_i, \lambda_i^k)}.$$

This concept has been extended to handle fluorescent effects [28]. In a different approach, Petitjean et al. [24] apply gradient-domain rendering [29, 30] to spectral rendering. They do not evaluate secondary wavelengths for the same path, but instead generate and later reconnect additional subpaths, allowing them to estimate spectral image gradients. These are used to reduce variance in a later step.

More recently, West et al. [14] show in their work on CMIS that stratifying wavelengths as in Equation 2.4 is inefficient if $p(\lambda)$ is not uniform. They propose rotating samples over the invertible *cumulative density function* (CDF) P_λ of this distribution, yielding the following rotation function:

Eq 2.6
$$\lambda_s = P_\lambda^{-1}((u + \frac{s}{C})\ mod\ 1) \cdot \bar{\lambda} + \lambda_{min}.$$

Here a uniformly distributed random variable u is instead stratified across a uniform distribution, and secondary wavelengths are then recovered through inversion transform sampling. As West et al. [14] use a mixture of emitter SPDs for $p_e(\lambda)$, their technique shows improvements especially for spiky illuminants.

2.3 Methodology

We reason that for spectral importance sampling, the distribution $p(\lambda)$ in Equation 2.3 should be optimally proportional to I_j, as then wavelengths with significant contribution are more densely sampled. Due to the complex nature of light transport, spectral renderers typically employ a predefined distribution for wavelength sampling. As mentioned, it is the product of sensor response p_s and a distribution p_e likely based on emission spectra. For example, a mixture of a scene's N_e emitter spectra $E_1, \ldots, E_{N_e}$ is constructed as

Eq 2.7
$$p_e(\lambda) = \frac{1}{w_e} \sum_{i=1}^{n} E_i(\lambda) \quad : \quad w_e = \sum_{i=1}^{n} \sum_{j=1}^{\lambda} E_i(j)$$

where w_e serves as a normalization constant. Such a mixture distribution is suitable for scenes with few or similar emitters, where it is likely proportional to observed radiances. Unfortunately, this strategy can be suboptimal even in simple scenarios (Figure 2.1). With non-uniform reflectance and participating media, observed radiances are usually not proportional to emitted radiances, and may vary significantly on a per-pixel basis. Further, in the presence of multiple emitters with thin, non-overlapping spectral bands, a mixture distribution is likewise suboptimal as no single emitter is sampled efficiently.

We propose to instead generate a viable p_e on a per-pixel basis, by prepending additional render passes to a conventional light transport algorithm. We split such a pass into two stages (Figure 2.2). In the first stage, we obtain a coarse but unbiased estimate $\widetilde{I}$ of the entire image. We do not map this image into a color space, but instead store the produced spectral-radiance values. In the second stage, we use

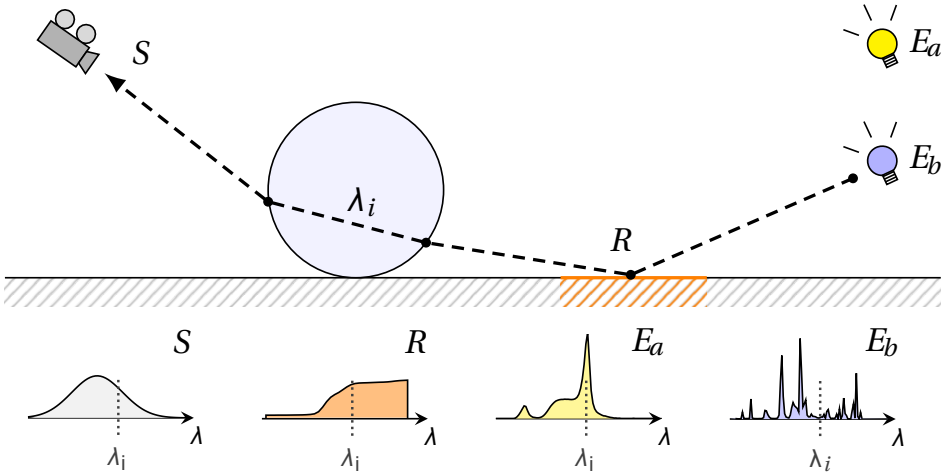


Fig 2.1 A light path passing through a dispersive medium is evaluated for wavelength λ_i, on which emitter E_a contributes greatly. Optimal wavelength sampling is proportional to the sensor response curve S, absorption at surface R, and the emitter E_b to which the path connects.

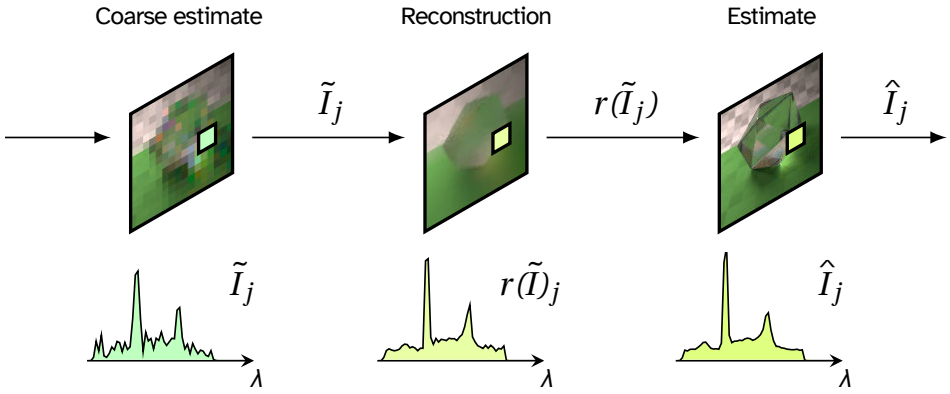


Fig 2.2 **Method overview.** For every pixel j, we obtain a coarse unbiased spectral-radiance estimate $\tilde{I}_j$, which we process in a reconstruction function r, and subsequently use as a wavelength sampling distribution, improving estimator efficiency.

a reconstruction function r on this image to obtain a biased but relatively noise-free estimate. In any subsequent pass, we can then employ the filtered radiances in $r(\tilde{I})$ as distributions for wavelength importance sampling. Instead of fitting a single spectral distribution to a scene, we essentially learn per-pixel distributions proportional to the incident radiance. This expands Equation 2.2 to the following:

Eq 2.8
$$\hat{I}_j = \frac{1}{N}\sum_{i=1}^{N} \frac{f_j(\bar{x}_i, \lambda_i)}{r(\tilde{I}_j)(\lambda_i) \cdot p_s(\lambda_i) \cdot p(\bar{x}_i \mid \lambda_i)},$$

where $r(\tilde{I}_j)(\lambda) \cdot p_s(\lambda)$ produces a viable probability density for sampling λ based on the estimate now available in pixel j. We detail our approach in the following.

Coarse estimate In a first pass, we obtain the coarse estimate $\widetilde{I}$. This step is straightforward: it has to remain cheap as it impacts total runtime. As the estimate should capture all aspects of incident spectral radiance, we cannot rely on biased alternatives and employ a path tracer with simple cost-saving measures, inducing:

1. A separate sample rate $\widetilde{N} < N$, reducing accuracy.
2. A separate image resolution, scaling the original by a factor $\alpha_s \leq 1$, increasing error at discontinuities.
3. Restriction to tracing paths of interest fully. Paths not encountering wavelength-dependent phenomena after several bounces are terminated early.

The reconstruction function will recover a spectral distribution that is adequate for our purposes. If necessary, said function filters noise, performs resampling, and accounts for culled paths with a fallback distribution (Equation 2.7). We evaluate the impact of image scaling and sample rates in Section 2.5. One measure we do not consider is a reduced spectral resolution, as this may cause thin wavelength bands to be insufficiently represented for effective importance sampling.

Reconstruction function The reconstruction function r processes the coarse image estimate $\widetilde{I}$. For simplicity, we employ a joint/cross bilateral filter [31, 32] for fast edge-preserving filtering. It leverages a secondary *guide image* I' to mark discontinuities; we rely on secondary scene information such as direct depth, normals, and albedo. Conventional renderers expose these attributes and a single ray per pixel suffices. Discontinuities are marked by the difference in pixel values in the guide image; therefore similar pixels separated by some discontinuity can still be considered during filtering. The filter is defined as

$$r_{filt}(\widetilde{I}_j) = \frac{1}{w_j} \sum_{k \in \Omega} G_{\sigma_s}(\|k-j\|)\, G_{\sigma_r}(\|I'_k - I'_j\|)\, \widetilde{I}_k$$
$$\text{Eq 2.9} \qquad : \quad w_j = \sum_{k \in \Omega} G_{\sigma_s}(\|k-j\|)\, G_{\sigma_r}(\|I'_k - I'_j\|),$$

where Ω is a local image neighborhood of pixels around j, and G_{σ_r} and G_{σ_s} are range and spatial Gaussian filters with standard deviations σ_r and σ_s, respectively. The weight w_j is a normalization factor that ensures all weights sum to 1, even in a discrete filter. We combine with joint bilateral upsampling [33], to perform a combined edge-aware resampling and filtering to a target resolution.

While we considered more advanced filters such as NL-means [34], we saw little improvement at a considerable computational overhead. Further, while we considered more recent work on denoising, these typically focus on trichromatic rendering, while we explicitly retain high-resolution spectral distributions after filtering.

The filtered distributions should sufficiently cover the contributing wavelengths, as these might otherwise introduce bias when importance sampling. To avoid this issue, even for low sample rates, we employ defensive mixture sampling, adding

an offset to our sampling distribution. Instead of a constant offset, we define a spectral distribution p_ϵ based on emitter spectra present in the scene:

Eq 2.10
$$\forall_\lambda \; p_\epsilon(\lambda) = \begin{cases} w_\epsilon & \text{if } \exists\, E : E(\lambda) \neq 0, \\ 0 & \text{else.} \end{cases}$$

Here, w_ϵ is a normalization weight ensuring a normalized distribution. The application of mixture sampling yields the defensive function $r_{def}(\widetilde{I}_j) = \epsilon \cdot p_\epsilon + (1-\epsilon)\cdot \widetilde{I}_j$, where the choice of $\epsilon \in [0,1]$ trades off potential benefits and detriments of the distributions in $\widetilde{I}$. Note that, if a collection of emitters contributed on all wavelengths uniformly, p_ϵ would become a constant offset.

Multiple pre-passes The described procedure lends itself to multiple passes. One pass serves as input to sample the next estimate, generating spectral distributions of continually improving quality. For K passes, we recursively define:

Eq 2.11
$$\widetilde{I}_j^k = \frac{1}{\widetilde{N}^k} \sum_{i=1}^{\widetilde{N}^k} \frac{f_j(\bar{x}_i, \lambda_i)}{p_j^k(\lambda_i) \cdot p_s(\lambda) \cdot p(\bar{x}_i \mid \lambda_i)}, \quad k \in [1, \dots, K],$$

where $p_j^k(\lambda)$ in turn samples the $k-1$th pass as

Eq 2.12
$$p_j^k(\lambda) = \begin{cases} r(\widetilde{I}_j^{k-1})(\lambda) & k > 1, \\ p_e(\lambda) & k = 1. \end{cases}$$

The first pass ($k = 1$) reduces to Equation 2.2, falling back to a default distribution such as an emitter mixture (Equation 2.7). The image scale α_s^k and sample rate $\widetilde{N}^k$ of the coarse estimate now vary in subsequent passes. We analyze several configurations (passes, scaling, and sample rate) in Section 2.5.

Sample reuse Note that samples from earlier passes are not combined with samples in later passes. As each pass forms an independent estimator with a unique distribution, a naïve averaging of different passes may lead to a noisier image. Robuster combinations of repeated passes have been explored in the context of path guiding [35], but may necessitate variance estimation and introduction of bias. In the interest of predictive spectral rendering, we do not include this combination.

Fallback mechanism A limitation of our method is the handling of near-uniform spectral distributions. Given uniformly distributed incident radiance, the best strategy remains to sample wavelengths uniformly or based on an emitter. Our estimate is unlikely to match such distributions perfectly. Consequently, we would expect reduced efficiency. To counteract this, we use the previously described multi-pass method to detect such cases in an earlier, cheaper pass, skipping the computation of all affected pixels in later passes.

We set a threshold on the *mean squared error* (MSE) between our normalized distribution and a default distribution E such as an emitter mixture (Equation 2.7), to determine when to fall back. We establish a threshold h and define the function

Eq 2.13
$$t_h(\tilde{I}) = \frac{1}{\Lambda}\sum_{i=1}^{\Lambda}(\tilde{I}[i] - E[i])^2 \leq h,$$

where we assume a discrete spectral representation with Λ bins, using $\tilde{I}[i]$ and $E[i]$ to access the spectral entries. Inserting this function into Equation 2.12 yields:

Eq 2.14
$$p_j^k(\lambda) = \begin{cases} r(\tilde{I}^{k-1})_j(\lambda) & k > 1 \ \wedge \ t_h(\tilde{I}_j^{k-1}), \\ p(\lambda) & k = 1 \ \vee \ \neg \ t_h(\tilde{I}_j^{k-1}). \end{cases}$$

Later passes only contribute to pixels with significantly different spectral distributions, eliminating most of the overhead of our method where it is unnecessary.

Integration with multiple wavelength sampling Reusing a light transport path for multiple wavelengths as in HWSS [13] remains an efficient technique for spectral noise reduction. West et al. [14] demonstrate the advantages of independent sample placement across spectral distributions, as opposed to a stratified one. We follow their approach, sampling a number of wavelengths warped according to our method's derived spectral distributions as per Equation 2.6. We construct an MIS estimator to combine C independently placed wavelengths as

Eq 2.15
$$\widehat{I}_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{s=1}^{C}\frac{f_j(\bar{x}_i, \lambda_i^s)\cdot w(\bar{x}_i, \lambda_i^s)}{r(\tilde{I}_j)(\lambda_i^s)\cdot p_s(\lambda)\cdot p(\bar{x}_i \mid \lambda_i^s)},$$

where $w(\bar{x}, \lambda)$ denotes the MIS weight accommodating for repeated sampling of $\bar{x}_i$ using the different wavelengths. By applying, for example, the *balance heuristic* [27]

Eq 2.16
$$w(\bar{x}_i, \lambda_i^s) = \frac{p(\bar{x}_i \mid \lambda_i^s)}{\sum_{k=1}^{C} p(\bar{x}_i \mid \lambda_i^k)},$$

we obtain the following estimator:

Eq 2.17
$$\widehat{I}_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{s=1}^{C}\frac{f_j(\bar{x}_i, \lambda_i^s)}{r(\tilde{I}_j)(\lambda_i^s)\cdot p_s(\lambda)\cdot \sum_{k=1}^{C} p(\bar{x}_i \mid \lambda_i^k)}.$$

This is identical to the CMIS estimator described by West et al. [14] but, instead of a fixed distribution, leverages our per-pixel distribution.

2.4 Implementation

We implement our method in Mitsuba [36], a research-oriented *C++*-based renderer. To simulate wavelength dependency, we extend specular and non-specular dielectric BSDFs with Cauchy's equation, enabling light dispersion in dielectric materials. Mitsuba uses a discrete *binned* spectral representation, which we configure to use 64 equally-sized bins over a $360-830\ \mathrm{nm}$ range (approximately $7\ \mathrm{nm}$ per bin in the visible light spectrum), mirroring CIE observer functions [37]. This discretization suffices for accurately representing emitters with thin wavelength bands, which are common for fluorescent lights and other gas-discharge lamps.

Our method has two parts: preprocessing, which consists of earlier render passes with reconstruction passes in between, and rendering, which produces a final estimate. Given this distinction, we can prepend the preprocessing to clones of Mitsuba's unbiased path tracing and volumetric path tracing integrators. These support *next-event-estimation*, and we extend them to leverage HWSS [13] and CMIS [14] for wavelength-dependent paths. For non-wavelength-dependent paths, all wavelengths are propagated. The only further modification to these integrators is the replacement of their respective wavelength sampling distributions with our distributions of choice. As a sensor response curve, we adapt the curve described in [12] for all evaluated methods.

2.5 Results

We evaluate our method on scenes with varied combinations of spectral distributions; the full set of reflectance spectra from a *Macbeth Color Checker* [38] and emission spectra covering common types of emitters, such as LEDs, incandescent bulbs, and fluorescent lights, from the *Lamp Spectral Power Distribution Database* (LSPDD) [39] under CC-Y-NC-ND 2.5 CA license, listed in Table 2.1 and displayed in Figure 2.3. We use the participating media parameters readily available in Mitsuba [40]. For each render, we produce RGB and full spectral outputs, the latter of which we use for error computations. References are produced with $N = 256k$ samples for smaller scenes and $N = 512k$ for larger scenes, with an unbiased unidirectional path tracer. We provide comparable error metrics as MSE. For a fair comparison, the measured runtime of our method always includes preprocessing overhead.

2.5.1 Parameter Evaluation

We first define a baseline configuration and then vary specific parameters. To avoid over-fitting, we use a separate geometrically simple scene with spectra not applied in the rest of the paper. Derived parameters are kept constant for all further results. We manually fix parameters whose influence is minimal; the range component of the bilateral filter $\sigma_r = 0.015$ by visually determining that edge preservation

Name	Short	LSPDD idx.	Type	Color temp.
GE Candle	INC	2484	Inc.	2450K
Philips Candlelight	LED1	2471	LED1	2700K
Ledtech PAR20	LED2	2470	LED2	5828K
Globe Twister	CFL1	2488	CFL	4749K
ELume PAR30 LN Flood	CFL2	2627	CFL	4066K
Energystar Twister	CFL3	2479	CFL	2700K

Tab 2.1 **Emission spectra**. Name, LSPDD index [39], emitter type and color temperature. *Short names* are referenced in the text.

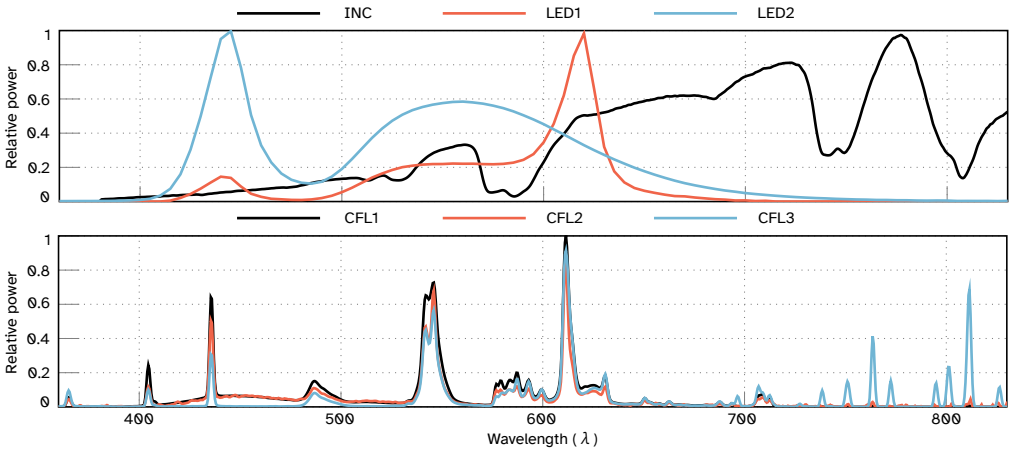


Fig 2.3 **Emission spectra**. We show used emission spectra (Table 2.1) obtained from the LSPDD [39]. Note the differences between incandescent, LED and CFL emitters.

is maintained, the amount of safe defensive mixture sampling with $\epsilon = 0.05$ (lower values occasionally produce undersampled wavelengths, but have little influence on effectiveness), and the fallback threshold $h = 0.0002$ (hereby, it only triggers on distributions near-identical to Equation 2.7).

The sample rate $\widetilde{N}$ does affect convergence (Figure 2.4), and we render our test scene for increasing $\widetilde{N}$ while generating a single pre-pass at full image scale ($K = 1, a_s = 1, \sigma_s = 1$). The influence on convergence diminishes for larger values, indicating that a low sample rate of $\widetilde{N} = 128$ suffices for our test scene. Our method's overhead is evident: a doubling of $\widetilde{N}$ incurs an expected doubling of preprocessing time, implying the importance of a careful choice. We consider three configurations of multiple passes (Figure 2.5). The topmost configuration (image scaled) quarters the number of pixels in each earlier pass, while the middle configuration (sample scaled) quarters the sample rate instead. The bottom configuration (both scaled) halves both parameters in each earlier pass. Each configuration requires near-equal preprocessing times, barring minor differences in scheduling and filtering. For each configuration, we show the influence of different sample rates ($\widetilde{N}$) and spatial filtering (σ_s) over two passes ($K = 2$). Using ($\widetilde{N} = 128, \sigma_s = 1$) we then

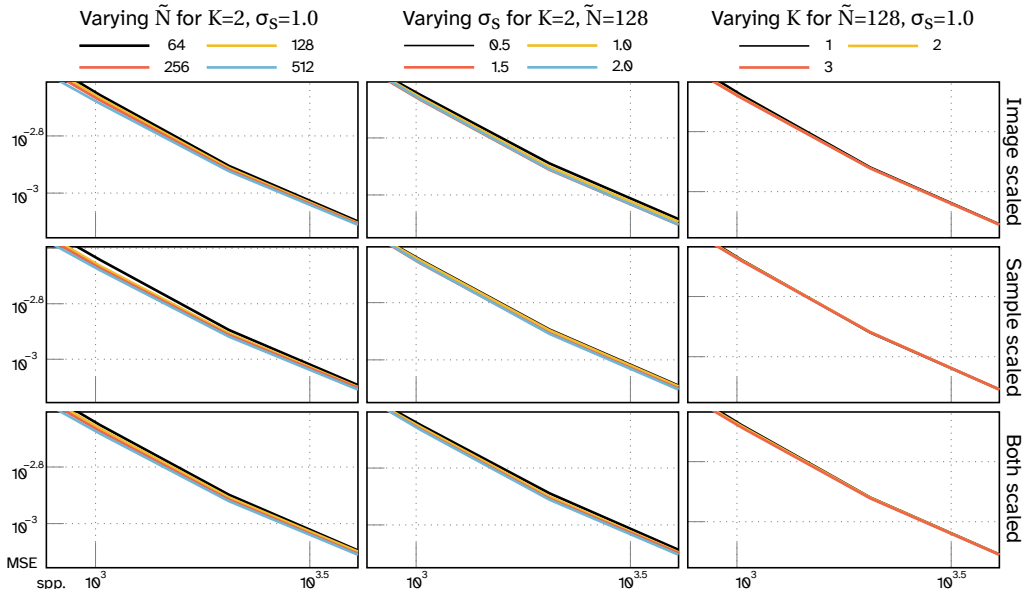


Fig 2.4 **Parameter evaluation**. We show the influence of a single pass with varying sample rate, on estimator convergence and runtime.

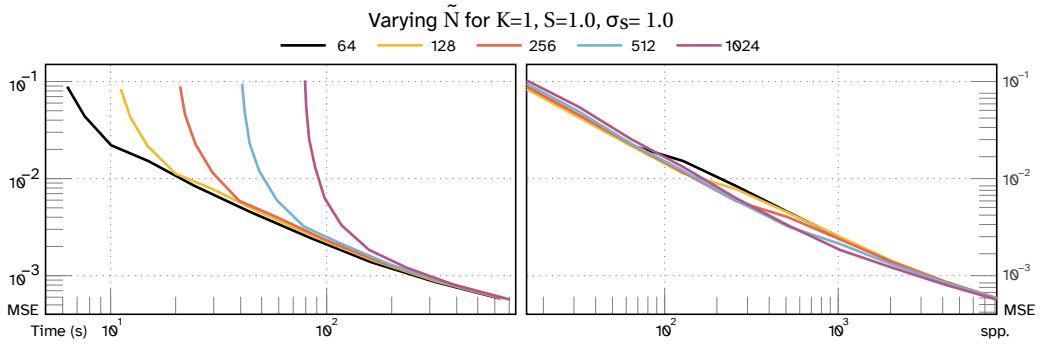


Fig 2.5 **Parameter evaluation**. We show the influence of different parameters in conjunction with two or more passes in three configurations

vary the number of passes. Evidently, an increased sample rate provides minor benefits at significant cost, so we retain $\widetilde{N} = 128$. We further select $\sigma_s = 1.75$ as a suitable spatial filter. Finally, while improvements from two or more passes are almost negligible, we select $K = 2$ as this allows us to leverage the fallback mechanism, reducing preprocessing overhead where a simpler sampling strategy suffices. Ultimately, differences between the three configurations are minimal, as such we select the sample scaled configuration, which remains the simplest approach.

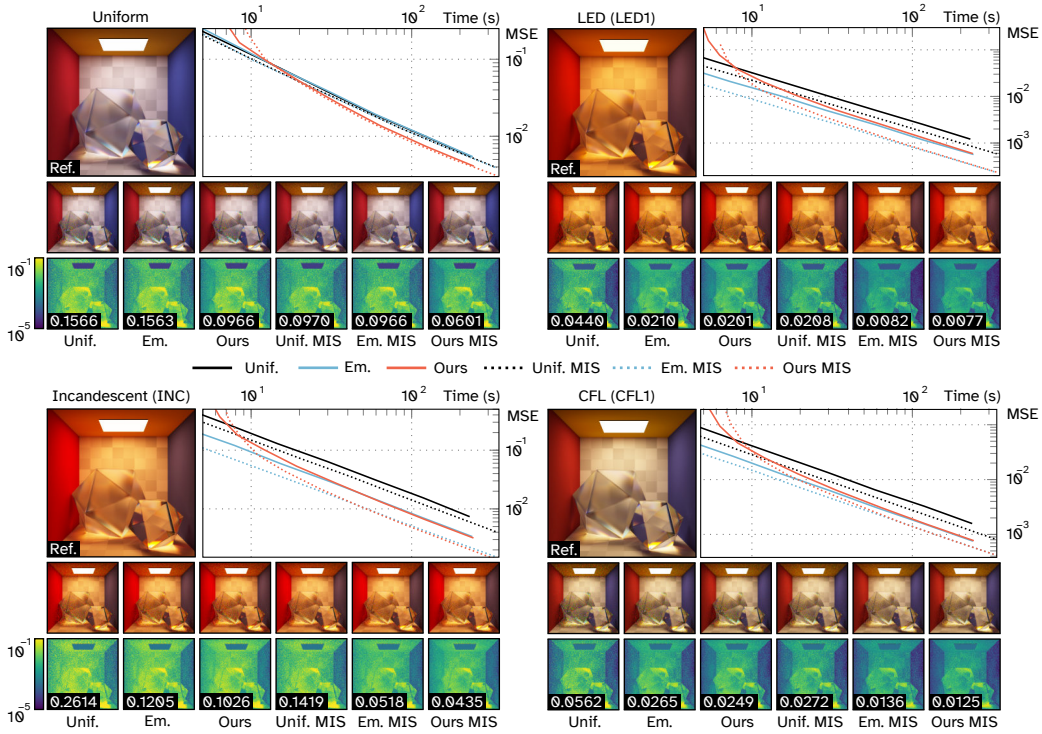


Fig 2.6 **Single emitter results**. We compare uniform sampling (*Unif.*), emitter sampling (*Em.*) and our method (*Ours.*) across four scenes with a single emitter whose distribution is varied (Table 2.1). In these scenes, emitter sampling is near-optimal. Our method matches emitter sampling in effectiveness, and outperforms it when emitter sampling becomes suboptimal.

2.5.2 Method Evaluation

We evaluate three wavelength sampling distributions for p_e: a uniform distribution (Unif.), the emitter mixture described in Equation 2.7 (Em.), and our per-pixel distribution (Ours), each multiplied by a sensor response p_s. We provide MSE over time for each method, and additionally show difference images for $N = 256$ spp. We evaluate both single and multiple wavelength sampling. Note that, for the latter, sampling a uniform distribution does not equate HWSS [13]. Said method stratifies wavelengths across the full spectrum (Equation 2.4), which can be problematic for a wide spectral range with a limited sensor response. We instead stratify across the distribution (Equation 2.6) as demonstrated in CMIS [14]. For the emitter distribution, multiple wavelength sampling employs evaluates CMIS [14].

Single emitter scenes We first compare each method in four scenes (Figure 2.6) containing a single emitter whose spectral distribution is made to vary. Emitter sampling is near-optimal in these scenes, and we expect our method to match it in performance. As demonstrated, this is the case. Our method delivers lower error in all cases, but is offset by an increase in runtime attributed to its preprocessing

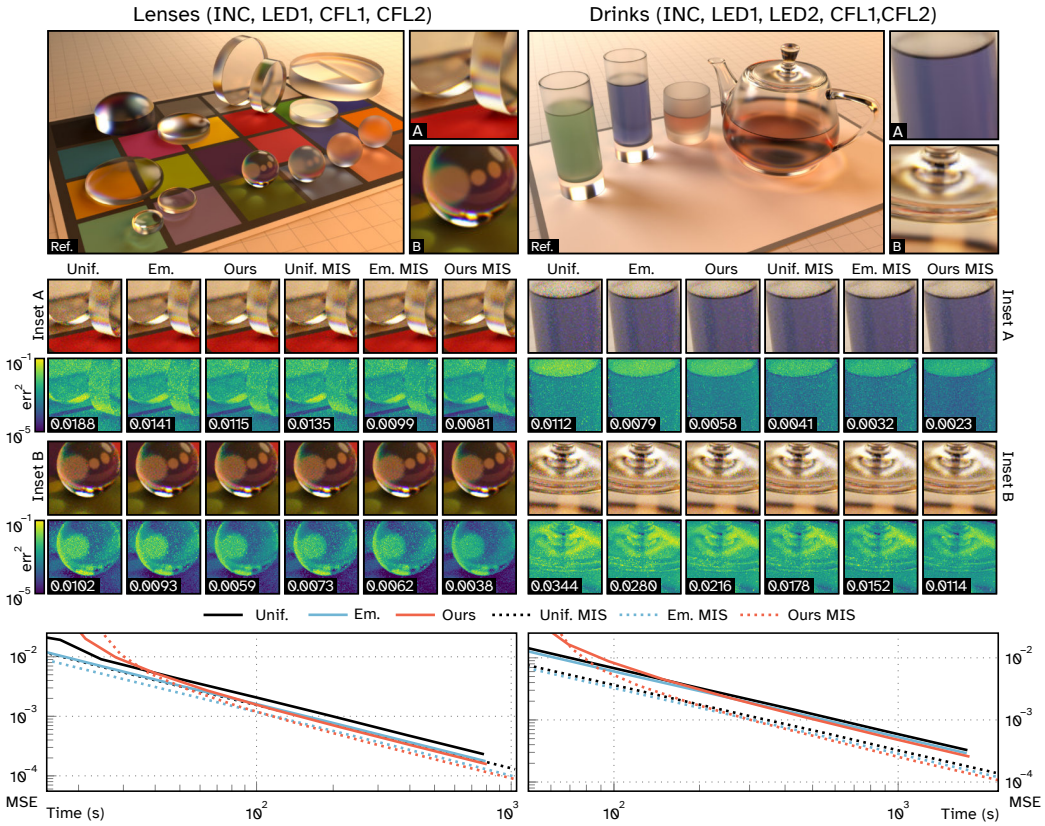


Fig 2.7 **Multiple emitter results**. We compare uniform sampling (*Unif.*), emitter sampling (*Em.*) and our method (*Ours*) in two scenes using a varied number of spectral distributions (Table 2.1). Plotted graphs are acquired over the whole image, and listed errors are for highlighted image insets. Our method is generally effective, but shows the strongest improvement in areas where there is significant absorption or chromatic dispersion.

overhead. As the estimator converges, this overhead diminishes relative to the total runtime. Interestingly, a notable improvement is visible in the *Uniform* scene, where our method affects areas with heavy absorption. Emitter sampling reduces to uniform sampling, losing its efficiency. Even with multiple wavelength sampling - which typically increases runtime costs - this difference remains significant.

Multiple emitter scenes We next compare each method in two scenes (Figure 2.7: *Lenses, Drinks, Boxes*) of varying complexity, each containing multiple different emitters. In the *Lenses* scene, we highlight two insets (*A*, *B*, $N = 256$ spp.) where highly dispersive phenomena are visible, demonstrating that our method provides a suitable sampling distribution for fine details. Error decrease varies strongly across the image, as in many places methods either do not necessitate wavelength sampling, or our method's fallback mechanism triggers. For $N = 1024$

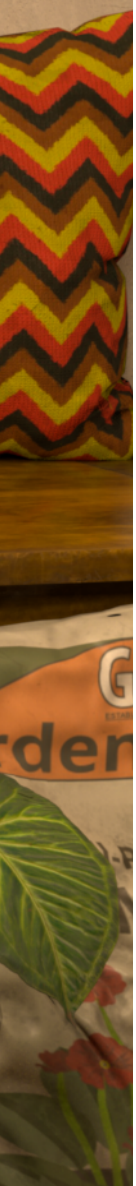
spp, we see an overall decrease in error compared to emitter sampling by 19.2%. The error decrease is small in some areas (inset *A*: 9.4%) but comparatively strong in others (inset *B*: 33.9%). We then demonstrate handling of participating media in the *Drinks scene*. Areas undergoing heavy absorption show strong improvements (inset *A*: 21.1%), though complex highlights caused by different emitters are also handled (inset *B*, 20.1%). Overall, for $N = 1024$ spp., this error decrease incurs a 17.6% time overhead. This overhead diminishes to 3.7% for $N = 4096$ spp., while the error decrease is only slightly reduced (inset *A*: 18.3%, inset *B*: 15.1%).

Across tests, emitter sampling remains comparatively performant. While for simpler scenarios our method is on par, the difference becomes pronounced when there is heavy absorption. Uniform sampling is easily outperformed by either method in all test cases, demonstrating the benefits of a proper sampling distribution.

2.6 Conclusion

We have developed a multi-pass method for accelerated spectral rendering, which is a simple method counteracting wavelength sampling problems in spectral light transport. We demonstrated that investing compute time to derive an approximate spectral-radiance distribution per pixel can improve convergence and reduce variance when using this distribution for importance sampling. Our method handles complex non-uniform spectral distributions, which are common in real-world emission and reflectance spectra. Given the benefits for difficult scenarios, we hope that it will contribute to making the use of spectral rendering more common.

In the future, we hope that integration with other spectral effects, such as fluorescence, becomes possible. This may be possible by storing unshifted wavelengths in the pre-pass.



3 Direct Spectral Uplifting via Controllable Color Constraints

Abstract

Spectral rendering is a crucial solution for photorealistic rendering. However, most available texture assets are RGB-only, and access to spectral content is limited. Uplifting methods that recover full spectral representations from RGB inputs have therefore received much attention. Yet, most methods are deterministic, while in reality, there is no one-to-one mapping. As a consequence, the appearance of uplifted textures is fully determined under all illuminants. Hereby, metamers, which are materials with differing spectral responses that appear identical under a specific illumination, are excluded.

In this chapter⋆, we propose a method which makes this uplifting process controllable. Hereby, a user can define texture appearance under various lighting conditions, leading to a greatly increased flexibility for content design. Our method determines the space of possible metameric manipulations and enables interactive adjustments, while maintaining a set of user-specified appearance constraints.

To achieve this goal, we formulate the problem as a constrained optimization, building upon an interpolation scheme and data-based reflectance generation, which maintain plausibility. Besides its value for artistic control, our solution is lightweight and can be executed on the fly, which keeps its memory consumption low and makes it easy to integrate into existing frameworks.

⋆ This chapter is based on "*Metameric: Spectral Uplifting via Controllable Color Constraints*", previously published in SIGGRAPH Conferece Proceedings (SIGGRAPH 2023) [41].

3.1 Introduction

Physically-based rendering systems are able to generate increasingly photorealistic imagery. While early approaches encoded physical properties as RGB colors, spectral rendering has become an important component for realism. A trichromatic approximation cannot accurately predict color phenomena under complex lights [8]. Spectral rendering, albeit at increased computational costs, addresses this shortcoming by performing light-transport for various wavelengths. It enables accurate color reproduction and supports phenomena such as fluorescence.

For spectral rendering, scene assets need spectral material properties, e.g., reflectance. This is a challenge, as spectral material capture is laborious and the related memory consumption can be significant. Further, content-authoring pipelines tend to target RGB and no solution allows an artist to easily interact with spectral definitions. Instead, a large body of work sidesteps these issues by uplifting colors to full spectra. This is an inherently ill-posed problem, as a color can stem from an infinite number of metameric spectra. Typically, solutions opt for the smooth and bounded shapes seen in reflectances, establishing a 1-to-1 mapping between RGB and resulting spectra. Such a conversion is restrictive, as it disallows metameric behavior. Hence, color matching becomes an issue; an uplifted material produces well-specified colors only under a single illuminant. In consequence, an artist may have to tweak materials to produce expected results under different illuminants.

We propose to make spectral uplifting controllable, so a user can define simultaneous material appearances under different illuminants through metameric behavior. For an input texture, we derive a simple polytope, forming a convex hull around the RGB texture data in an $\mathbb{R}^3$ space. For each vertex of this hull, a smooth and bounded reflectance is generated, respecting user-provided color constraints that define how the reflectance stored in each vertex should appear under different illuminants. We then use generalized barycentric coordinates to transfer the spectra from the vertices of the convex hull to the enclosed RGB texture data. By employing prior work on metamer mismatch volume estimation [22], we can ensure that user-provided color constraints are restricted to possible solutions, ensuring a minimal roundtrip error. As the uplifting builds upon an interpolation scheme, our solution provides a simple, compressed format for spectral textures, leading to a practical representation for spectral-rendering contexts.

Specifically, we make the following contributions:

- An efficient uplifting technique supporting color-constancy;
- A solution for constrained artistic control;
- A compact representation for spectral textures.

In the following, we cover the relevant background (Section 3.2), before presenting our method (Section 3.3) and results (Section 3.5). Finally, we discuss our findings (Section 3.6) and conclude (Section 3.7).

3.2 Background

Color theory Color as a phenomenon is determined by the signals of a set of observing sensors, which respond differently to the spectral distribution of incident light [37]. Such sensors are described by observer functions. Given an illuminant's spectral power distribution (SPD) $e(\lambda)$ (equal energy *E* or *D65* daylight illuminant are often-chosen whitepoints), we can express the response to a spectral surface reflectance $r(\lambda)$ as

Eq 3.1
$$\Phi(r) = \int_{\Lambda} \phi(\lambda)\ e(\lambda)\ r(\lambda)\ d\lambda,$$

where $\phi(\lambda)$ is a drop-in for the set of observer functions, and Λ the range of light (often the visible spectrum). This is formalized by the *International Commision on Illumination* (CIE) in the CIE XYZ color space with a trio of standard observer functions $\bar{x}, \bar{y}$ and $\bar{z}$. A variety of color spaces is defined as linear transformations of CIE XYZ, such as sRGB, which is prevalent in computer graphics.

The mapping Φ in Equation 3.1 describes a linear transformation from an essentially infinite-dimensional spectral space $\mathbb{X}$ to a (usually) three-dimensional color space. We call this combination of observer functions and illuminants a *color system*. The set of possible responses $\{r \in \mathbb{X} \mid \Phi(r) \neq 0\}$ in any given color system forms a convex region, referred to as its *object color solid* (OCS).

Metamerism We summarize Finlayson and Morovic [42] and Logvinenko et al. [43], who discuss metamerism and related concepts. Given a known color system and signal, uplifting means inverting Equation 3.1 by finding a reflectance s.t.

Eq 3.2
$$\Phi^{-1}(\Phi(r)) = r.$$

This is an ill-posed problem due to the underdetermined nature of the linear system in Equation 3.1. Thus, there is a convex set of reflectances, being *metameric* with respect to this color system, i.e.,

Eq 3.3
$$\Phi^{-1}(\Phi(r)) = \{r' \in \mathbb{X} \mid \Phi(r) = \Phi(r')\}.$$

All spectra in the above *metamer set* are solutions to Equation 3.2. A secondary color system Ψ with differing observer or illuminant, applied to the reflectances in this metamer set, might have its signal responses differ. This is called respectively *observer-induced* and *illuminant-induced* mismatching. Formally, if the metamer set $\Phi^{-1}(\Phi(r))$ is mapped to Ψ, this results in a non-singleton OCS called a *metamer mismatch region*. This region represents the full range of colors that may be observed after a color-system change.

Much work focuses on estimating OCS boundaries. Given its novelty, we highlight the work of Logvinenko et al. [43]. It uses a linear mapping $\Gamma : \mathbb{X} \to \mathbb{R}^6$ s.t. $\Gamma(r) =$

(z, z'), where $z = \Phi(r)$ and $z' = \Psi(r)$ form the corresponding color signals over the set of Equation 3.3. The authors show that, for a given z, the set of signals in a metamer mismatch volume is a cross-section of Γ:

Eq 3.4
$$\mathcal{M}(z, \Phi, \Psi) = \{z' \in \mathbb{R}^3 \mid (z, z') \in \Gamma\}.$$

While points inside the mismatch volume are formed by different metamers, boundary points reduce to a single *optimal* spectrum. A notable property of optimal spectra is that they are elementary step-functions, consisting only of transitions between zeroes and ones. Knowledge on optimal spectra has expanded over the years [44–46]. In our work, we apply the method of Mackiewicz et al. [22] to find optimal spectra on mismatch volume boundaries, leading to a conservative approximate convex hull around $\mathcal{M}$. This approach establishes maximal theoretical boundaries to metameric mismatching. Empirically-established boundaries seem substantially smaller [47], when ignoring structural colors.

Spectral distributions We briefly expand on the properties of natural spectra. Illuminants typically describe a quantity of energy per wavelength, while reflectances describe a surface's effectiveness in reflecting said energy. Illuminants are positively unbounded and express a variety of shapes dependent on the involved physical processes. In contrast, reflectances are bounded to $(0, 1)$, as they are energy-conserving. Fluorescent effects break this constraint, but should not be encoded as one-parameter functions. Further, reflectances are typically band-limited in the visible spectrum. This is considered a property of natural pigments, but not of structural colors [48]. In this and much of the related work, we restrict ourselves to handling smooth reflectances.

Spectral uplifting While we briefly cover prior methods, we refer the reader to Weidlich et al. [49] for a broader categorization.

An early method for spectral uplifting, which nowadays has mostly theoretical implications, was proposed by MacAdam [16]. The more recent and long-time standard method of Smits [17] describes uplifting as an optimization problem, generating combinations of seven precomputed primary spectra. While this approach is fast, the resulting reflectances can break boundedness constraints. More recently, Meng et al. [18] precompute sets of spectra on a grid spanning the xy chromaticity plane, recovering intermediates through interpolation. Their method produces smooth reflectances, but introduces round-trip errors for highly saturated colors. Otsu et al. [19] improve on this approach by clustering measured spectra into a KD-tree over the xy plane. Inside each cluster, the authors interpolate weights applied to PCA-derived basis functions. While the resulting reflectances are inherently smooth, discontinuities arise at cluster boundaries, as different bases are used. Finally, Mallet and Yuksel [50] describe convex combinations of three bases on sRGB gamut vertices, enabling reconstruction of input data with little roundtrip

error. However, due to the saturated shapes of their basis, uplifted spectra become arguably blocky. Further, their approach is restricted to uplifting of in-gamut sRGB data. Our method extends this concept, while avoiding such problems.

In their work, Jakob and Hanika [20] use a low-dimensional parameterization of a sigmoidal function space. They precompute function coefficients inside a three-dimensional color lookup table. Function reconstruction from interpolated coefficients is inherently fast, and produces smooth reflectances with low round-trip error. Several extensions cover out-of-gamut spectral effects such as fluorescents [51, 52]. A similar, Fourier-space based approach is demonstrated by Peters et al. [21], addressing certain round- issues of the sigmoidal. Tódóva et al. [53, 54] expand on this method, to our knowledge being the first to introduce constrained spectral uplifting. They enable seeding of the method's coefficient generation, such that certain acquired spectra are reproduced accurately. Coefficient generation is costly, however, and their complex encoding of seeded constraints introduces overhead during rendering. Our approaches differ fundamentally. We generate reflectances from user-provided color constraints, implying we recover color-matched spectra, not reproductions. While enforcing color-constancy, this enables authoring of metamers for which spectral acquisition is difficult. Combined with our toolkit's interactivity, this allows artistic expression in uplifting.

3.3 Methodology

Here, we present our constrained spectral uplifting, illustrated in Figure 3.1. The core of our solution comprises three elements. First, we generate a low-complexity convex hull in $\mathbb{R}^3$ around the texel colors of an RGB texture, and recover generalized barycentric coordinates to represent these texel colors as convex combinations of the hull's vertices. Second, we generate a small number of metamers with minimal round-trip error for the vertices of this convex hull, fulfilling artist-provided constraints that indicate the appearance of the vertex reflectances in different color systems. Third, during rendering, we spectrally uplift the input texture as a convex combination of the vertex metamers.

We first cover our method's foundation (subsection 3.3.1), followed by convex-hull construction (subsection 3.3.2). We then describe solving for vertex reflectances and texture uplifting (subsection 3.3.3). Afterwards, we analyze user-guided uplifting and explain how to steer it via a set of RGB textures with known color systems (subsection 3.3.4). We reserve implementation details for Section 3.5.

3.3.1 Foundation

As a color system describes a linear transformation (Equation 3.1), for any given color signal within that color system, there exists a set of metameric reflectances. Let n color signals $\Phi(r_1), \dots, \Phi(r_n)$ with corresponding reflectances $r_1, \dots, r_n$. We can then define a convex combination of these color signals using scalar weights

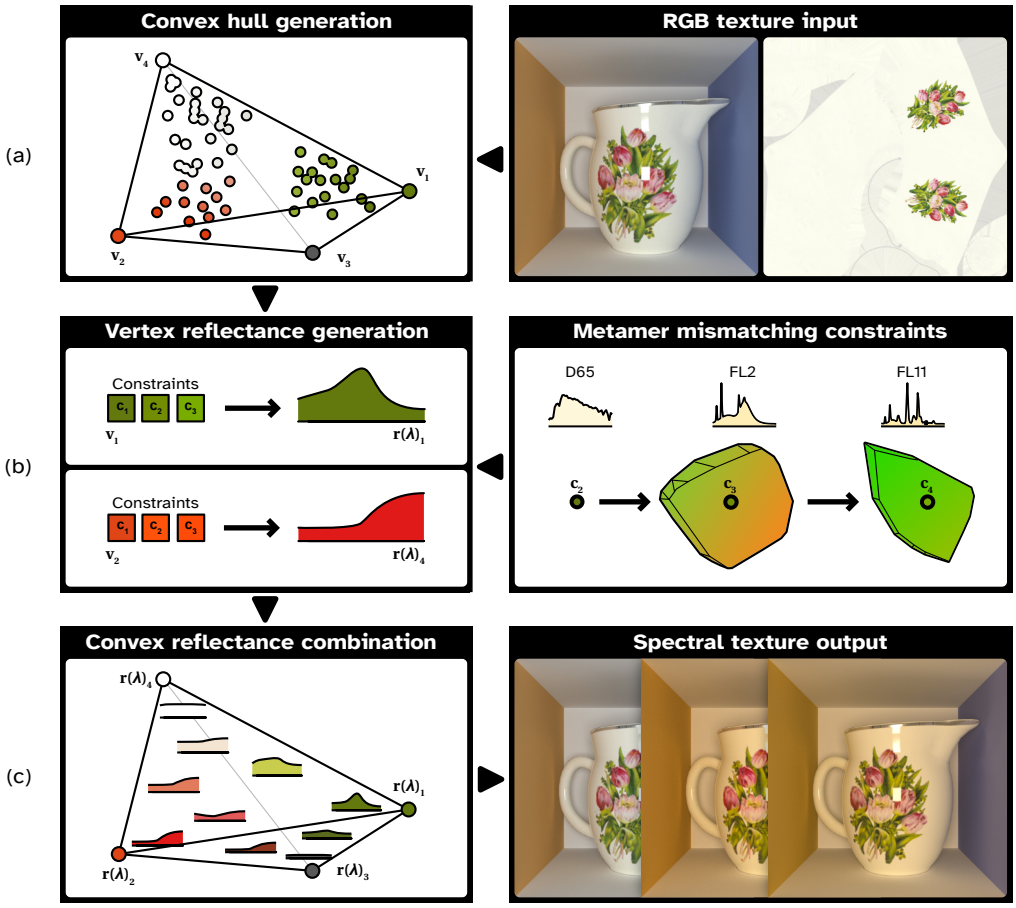


Fig 3.1 Method overview. As a preprocess, we (a) generate a low-complexity convex hull around a texture's RGB inputs. For each hull vertex, we then (b) generate a spectral reflectance satisfying artist-configurable color constraints. Finally, during rendering, we (c) recover texture reflectances as convex combinations of vertex reflectances.

$a_1, \ldots, a_n$, satisfying $\forall_i \; a_i \geq 0$ and $\sum a_i = 1$. Due to linearity:

Eq 3.5
$$\sum \Phi(r_i) a_i = \Phi\left(\sum r_i a_i\right).$$

In consequence, we see a direct relationship between linearly combined reflectances and linearly combined color signals. For example, the mean of two metamers is itself a metamer. This principle holds for arbitrary reflectances (Figure 3.2), and is used in most prior work.

Meng et al [18] apply this observation to interpolate spectra mapped to the xy chromaticity plane. Yet, a colorspace is trichromatic ($\mathbb{R}^3$). Any interpolation between three points leads to a triangle, while points outside the triangle's plane cannot be interpolated. Therefore, mapping multiple color signals to a plane is

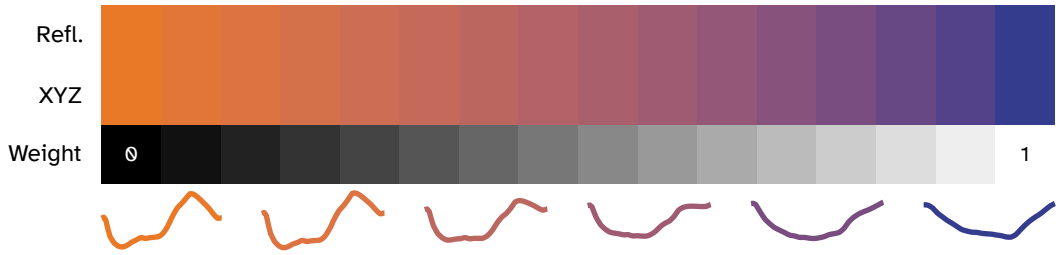


Fig 3.2 Following Equation 3.5, we show colors obtained from linearly mixing two reflectances, and corresponding results obtained from mixing their respective colors instead.

inherently problematic. Instead, we generate a convex hull with n vertices around the input colors in an $\mathbb{R}^3$ space such as CIE XYZ. This hull is minimally a 3-simplex (tetrahedron), though added vertices provide benefits such as a finer-grained control over uplifting. As all input colors of the texture to be uplifted lie within the hull, we can describe these colors as convex combinations of the hull's vertices, with scalar weights $a_1, \dots a_n$ ($\forall_i a_i \geq 0$, $\sum a_i = 1$). As each vertex position is a color signal, we can find a suitable reflectance for each vertex from the signal's metamer set. Following Equation 3.5, we then recover valid reflectances for each input color, by computing the convex combination for these vertex reflectances.

3.3.2 Convex hull

In theory, the advantage of interpolating reflectances from the convex hull is that if vertex reflectances are bounded to [0, 1], so will a resulting convex combination. Yet, the convex hull should not jut out of the color space, as, otherwise, reflectances cannot be reliably found for these vertices; they would not map to any color signal in the color system. This situation is common when only relying on a tetrahedron. For example, consider an RGB texture covering most of the gamut, and its $\mathbb{R}^3$ representation (Figure 3.3). An enclosing tetrahedron's vertices would lie outside the color-system boundaries. We avoid this issue using a general polygonal hull.

For a general polygonal hull, we can determine suitable weights to enable convex combinations. We rely on Mean Value Coordinates (MVCs) [55], using the method of Ju et al. [56] to adapt MVCs for triangle meshes. MVCs are in $[0, 1]$, sum to one, and are continuous within a hull's interior. The latter property ensures that recovered reflectances do not impress sudden discontinuities between upliftings, which is a problem Otsu et al. [19] encounter.

Hull generation Tan et al. [57, 58] show in their image segmentation work, that four to ten vertex structures encompass most input images, implying that a coarse hull typically suffices. We similarly generate an enclosing mesh around input colors using the *Quickhull* algorithm [59] and subsequently perform a progressive mesh simplification [60] using repeated edge contractions. To generate a simplified

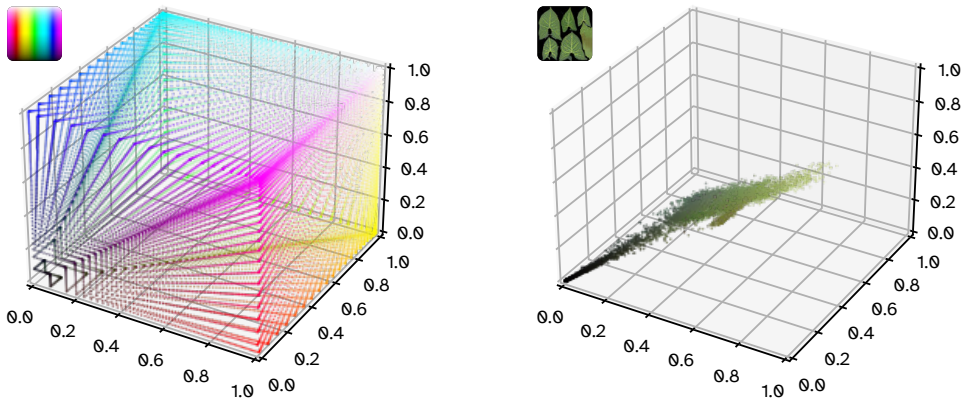


Fig 3.3 A pair of $\mathbb{R}^3$ mappings of texture data, in linearized sRGB.

convex hull, we should place a vertex resulting from a contraction strictly outside the existing hull. Sander et al. [61] formulate such an optimization by ensuring that each new vertex strictly adds volume to a mesh. They then perform the contractions in an order that greedily minimizes this added volume. This approach might move contracted vertices outside the color system's gamut, in which case Tan et al. [57, 58] encounter reconstruction errors. To ensure valid vertex placement, we establish boundary constraints as follows.

First, we determine the region of possible color signals that a color system can produce, which is a convex hull; the OCS (Section 3.2). For example, a green illuminant will not allow any reflectance to produce a red color. We find this OCS hull using the recent sampling-based approach of Mackiewicz et al. [22]. We then restrict vertex contractions to the hull's interior as otherwise, at a later stage, we cannot find reflectances for vertices outside the color system. Whenever a potential contraction falls outside the hull, the vertex is projected to the hull's surface. If this reprojection results in a contraction with negative volume, it is discarded and another is selected. Contractions are then repeated until the intended number of vertices is reached, or no further contractions can be performed without breaking the above constraints. We show a convex hull generated by repeated progressive mesh simplification in Figure 3.4.

3.3.3 Reflectance generation

Given the convex hull, we next generate reflectances for its vertices.

As in prior work, instead of working with spectra directly, we use weighted sums of basis functions obtained through PCA, which is an established representation for low-banded spectra based on measured data [19, 62–65]. We build our basis using a dataset of $\sim 41M$ reflectances [47], which gathers earlier works [66–69] representing a variety of natural and synthetic materials such as sediment, wood, plant-life, skin, food, paints, plastics, and textiles. Reflectances are measured over

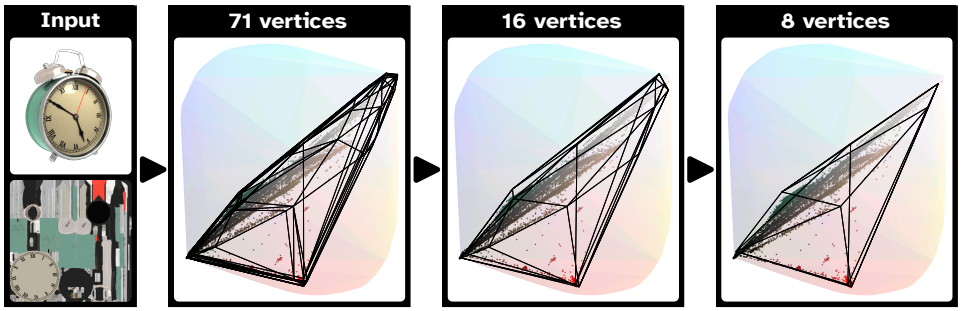


Fig 3.4 Progressive mesh simplification [59] of a convex hull, following the work of Tan et al. [57, 58], fitted around texture data (left). This results in simpler structures, but through volume-preserving constraints remains convex. Contracted vertices are bounded to a *object color solid*, the convex region of possible responses in a color system.

the $400-700\text{nm}$ spectral range at 10nm intervals.

PCA reduces the dimensionality of a space. This leads to smooth reflectances, given a low number of basis functions; prior work has established that three functions are minimally required for most reflectances [62]. In short, we represent input reflectances as discretized, $k-$dimensional vectors r, resampling inputs if necessary. We then acquire m basis functions ($m \leq k$) using PCA, encoding the first m principal component eigenvectors in a $k \times m$ matrix B. Contrary to prior work, we retain $m > 3$ functions, ensuring that a single color system does not fully determine the resulting reflectance, leaving room for metameric mismatching.

We next formulate a discretization of Equation 3.1: $\Phi(r) = \Phi B w$, where w is an unknown $m-$dimensional weight vector and $\Phi(r)$ is a known color signal. The corresponding color system Φ is encoded as a $3 \times k$ matrix of observer functions, multiplied with a measured illuminant. Solving for w allows us to recover reflectances as $r = Bw$, which we store explicitly as a $k-$dimensional vector in the corresponding hull vertex. We solve via a linear-programming optimization [70]. Hereby, we can enforce the solution's boundedness using additional constraints, i.e., $0 \leq Bw \leq 1$.

With a single color-system constraint (e.g, colors under *D65*), we can always find a reflectance (Equation 3.3). However, the system is underconstrained, thus, we can attempt to satisfy c secondary constraints $\Psi_1(r), \ldots, \Psi_c(r)$ under different color systems $\Psi_1, \ldots, \Psi_c$ simultaneously. In other words, for each vertex, we can specify its color signals in different color systems (e.g., colors under *D65* and *FL11*). The metamer set, with respect to all constraints, is then:

Eq 3.6
$$\Phi^{-1}(\Phi(r)) = \{r' \in \mathcal{X} \mid \Phi(r) = \Phi(r') \wedge \forall_i \Psi_i(r) = \Psi_i(r')\}.$$

Involving all constraints, a solution w implies the existence of a reflectance $r = Bw$, which lies in the intersection of all color signals' respective metamer sets. The opposite also holds; no shared metamer exists if the sets' intersection is empty.

Determining high-dimensional metamer set boundaries is expensive, unless re-

strictions are imposed [42]. However, we can establish boundaries on mismatching between color signals in different color systems. To determine the region of valid weights w, we hence restrict the space of solutions by establishing input constraints in $\mathscr{R}^3$, one constraint at a a time. In each step, we produce a metamer mismatch volume (Equation 3.4), s.t. prior constraints are satisfied. We employ the method of Mackiewicz et al. [22] to find this volume, clipping the constraint space as before. The order of constraints does not matter, as the resulting set of solutions lies within the insersection of all metamer sets.

Optimization then leads to a valid weight w among all possible solutions in the remaining set. We optimize for the smallest norm. If a metamer mismatch volume collapses to a point, a single weight forms the optimal solution. Upon finding w, we convert back to a spectrum ($r = Bw$) and store the result in the corresponding hull vertex. We store this spectrum and not the weight, as the number of vertices is low, and we avoid this multiplication during rendering. This further allows using an acquired reflectance, should this be available for the vertex.

Texture reconstruction Once vertex reflectances are determined, uplifting of the input texture's color signals is straightforward by invoking the corresponding convex weights and Equation 3.5. Indeed, uplifting a pixel p boils down to a matrix multiplication between MVC weights and vertex reflectances - for a single wavelength, only an inner product is needed. Specifically, given the MVC vector of pixel p; $A(p) := (\alpha_1, \cdots, \alpha_n)$, and vertex reflectances $R_1, \cdots R_n$, the uplifted spectrum is equal to $A(p)(R_1, \cdots, R_n)$.

3.3.4 User interaction

To enable constrained control, our toolkit provides an interface. To begin uplifting, a user provides an input texture and primary color system (e.g. CIE XYZ and *D65*). As our method can uplift without secondary constraints, we immediately show reflectances and spectral renders of the input texture for different color systems. The interface then focuses on the convex hull in $\mathbb{R}^3$. Users can modify the hull to suit their needs, or add constraints to vertices.

As the user selects a vertex, a weight map illustrates how texels are affected by changes to this vertex. Next, the user can add a color system (e.g., *FL11*), and the interface shows a mismatch volume for the vertex in this system. The volume's interior describes all color signals that can be produced under *FL11*, while maintaining appearance in *D65*. The user can freely modify the vertex color in this second system, and a reflectance is generated fulfilling both constraints (*D65* and *FL11*). As the volume is color-coded, a user can intuit how a secondary constraint affects appearance of affected texels under *FL11*. Once satisfied, the user can add further constraints on other vertices, or via additional color systems.

The interface is interactive and provides direct visual feedback. An executable is available, and we show an example session in a supplementary video.

Image seeding Direct vertex manipulation gives full control and allows precise definitions. An additional option is to define constraints indirectly by providing measured images, which can be acquired or authored. The user then provides a color system per input image. The first image and color system (e.g., *D65*) is used as a reference. As input constraints can fall outside of the metamer mismatching boundaries of convex hull vertices, we cannot guarantee that a solution exist s.t. the input is reproduced without roundtrip error. In consequence, we minimize this error while preserving the primary input image. We proceed as follows.

We sample a random subset of texels from the primary image, and obtain secondary color signals at these texels from the secondary images. For each sample, we test whether a solution exists by attempting to recover a suitable reflectance, as authored textures may violate constraint boundaries, and otherwise sample again. We now have a set of texels with valid color constraints, and connect these to the convex hull using the previously computed MVCs. We next solve for vertex reflectances whose convex combinations satisfy the constraints at each sampled texel. This can again be specified as a linear programming optimization, in which basis function weights are found from which we can derive vertex reflectances.

If no solution exists, we reduce or relax the set of texel constraints. Hereby, we increase the space of possible solutions, but incur roundtrip error in secondary color systems. As the convex hull is unaffected, the primary will always be reproduced.

3.4 Implementation

Spectral-rendering systems typically vectorize evaluation of a single light path across several wavelengths. In line with this, we describe our approach. Texture sampling can be described as a function accepting a combined position-wavelength pair, i.e. $f(p, \{\lambda_1, \dots, \lambda_k\})$, with $k = 4$ being common. To facilitate vectorized texture access, we store the m constrained vertex spectra of our convex hull in a $1 \times m$ texture R. Here n is the number of wavelength bins used and each pixel stores the bin's w components.

We further store the convex weights $a_1, \dots, a_n$ for each pixel in a two-dimensional texture A, which is the size of our input rgb texture.

Texture sampling is then expressed as a vectorized inner product:

$$f(p, \{\lambda_1, \dots, \lambda_k\}) = \begin{bmatrix} r(\lambda_1) \\ \vdots \\ r(\lambda_k) \end{bmatrix} = \begin{bmatrix} A(p) \cdot R(\lambda_1) \\ \vdots \\ A(p) \cdot R(\lambda_k) \end{bmatrix},$$

where the convex weights A are shared across all products, while the per-wavelength reflectances R are evaluated independently. Internally, the system uses XYZ color values for a device-independent representation.

3.5 Results

First, we evaluate accuracy, discuss implementation details, and establish used parameters. We then evaluate our method's per-vertex reflectance recovery (subsection 3.5.1), followed by full-texture upliftings (subsection 3.5.2). Finally, we detail rendering performance (subsection 3.5.3).

Implementation We implement most components as a preprocess in our uplifting toolkit. We operate on CIE XYZ values internally, enabling device-independent operations. We rely on *OpenMesh* [71] for mesh simplification, and the *COIN-OR CLP* solver [72] for linear-programming. We offload most work to *OpenGL*; MVC computation and rendering of uplifted textures are examples, though we note that MVCs are sensitive to numerical precision due to imprecise trigonometric functions. Finally, our user interface leverages *Dear ImGui* [73].

We define a straightforward exportable texture format consisting of two data blocks; vertex reflectances and MVCs. Disk storage is compressed using *zlib* [74]. We further implement a texture plugin for *Mitsuba 3* [75]. Uplifting is implemented as four inner products, vectorized over four wavelengths. MVCs are shared across vector units, while vertex data is sampled per unit. We show example renders in *Mitsuba*, using authored spectral materials (Figure 3.8).

Setup We set a $400-700\mathrm{nm}$ spectral range for all methods, and describe discrete spectra using $k = 64$ bins to handle high-frequency illuminants, resampling PCA inputs where necessary. Our choice of spectral range is motivated by our basis; our toolkit supports wider ranges such as the customary $360-380\mathrm{nm}$. Across tests, we use $n = 8$ convex hull vertices; note that our method uses higher numbers if a hull cannot be simplified further.

While prior work employs $m = 3$ basis functions, Mackiewicz et al. [22] show that metamer mismatching boundaries are reduced by any linear model. The choice of m thus pivots on two factors; low values generate low-banded reflectances, while high values enable metameric mismatching. We generate mismatch volumes and interior metamer sets for $m = 6, 8, \ldots, 16$ selecting $m = 12$ for further tests (Figure 3.5). As shown, mismatch boundaries are conservative approximations dependent on m. However, we do not consider this a problem, as Zhang et al. [47] show that empirically measured boundaries are smaller than the theoretical maximum.

3.5.1 Single-reflectance recovery

We sample reflectances from a *BabelColor Average* dataset [59], and measure color signals for standard illuminants *D65*, *A*, *E*, *FL2*, *FL11*, and *LED-RGB1*. We uplift for *D65* using the methods of Smits [17], Meng et al. [18], Otsu et al. [19], and Jakob and Hanika [20]. We then apply our method without and with $c = 1, 2, 3$ secondary constraints, for *FL2*, *FL11* and *LED-RGB1*.

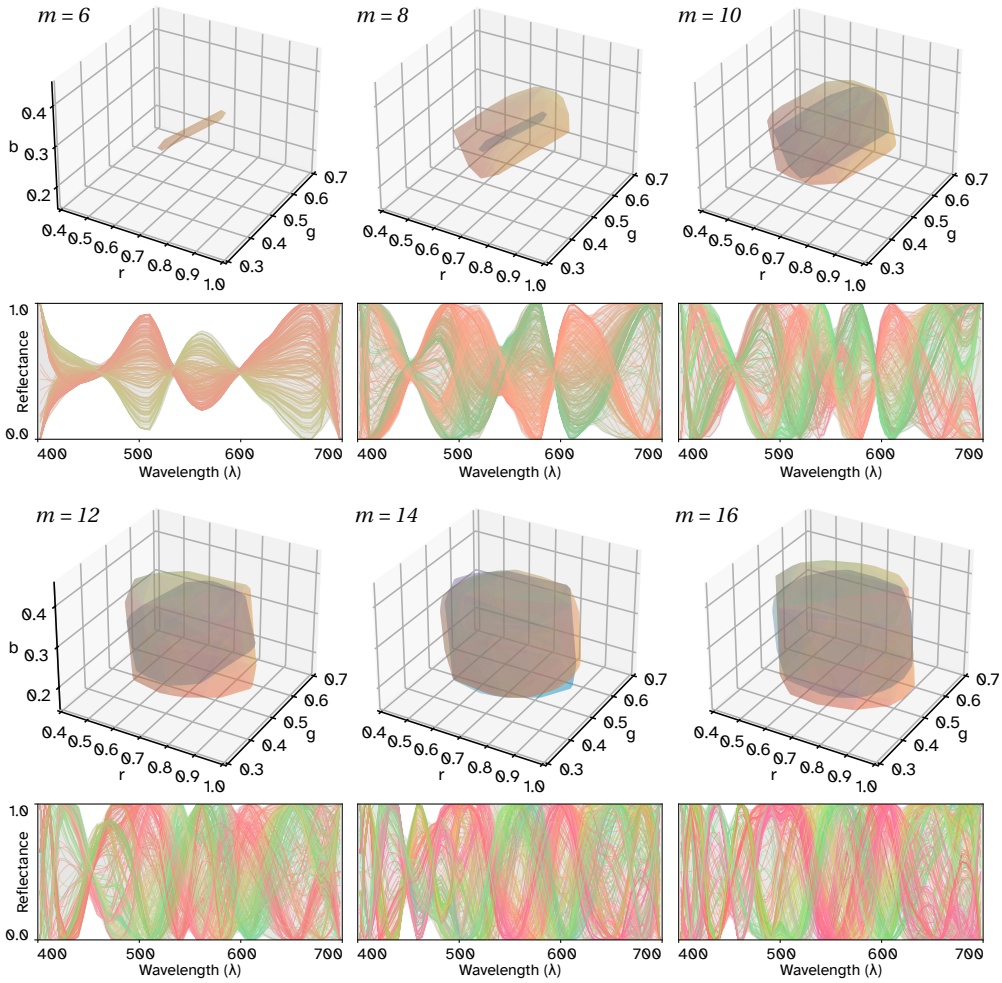


Fig 3.5 We generate metamer mismatch volumes (top) for illuminant-induced mismatching ($D65 \rightarrow FL11$) of a neutral gray reflectance, using m basis functions. We show sampled metamer reflectances (bottom) for each volume for $m \geq 6$. In more restrictive cases ($m < 6$), volumes collapsed to singletons.

Figure 3.6 shows roundtrip colors and CIE LAB ΔE_{00} color difference for each method ($\Delta E_{00} \leq 1$ implies no perceptible difference; $\Delta E_{00} \leq 2$ implies minimal mismatching). With the exception of Smits [17], all methods correctly recover for *D65*. Further, all methods except Otsu et al. [19] display metameric mismatching for *FL11*; our method without additional color constraints is no exception here ($\Delta E_{00} = 7.85$). With constraints, results improves significantly; we measure $\Delta E_{00} \leq 0.72$ ($c = 2$) and $\Delta E_{00} \leq 0.14$ ($c = 3$), far below the perceptible limits for all illuminants.

	Sigmoid	n = 4	n = 8	n = 16	Ground
Runtime (s)	31.35	32.83	34.18	35.10	36.40
Memory (MB)	16.0	16.0	32.0	64.0	792.0

Tab 3.1 Runtime/texture memory during rendering; our method with n vertices, the sigmoidal [20], and a hyperspectral ground with 186 bins. Run on Ryzen 9 5950X & NVIDIA RTX 3070.

3.5.2 Multiple-reflectance recovery

We demonstrate manual texture authoring in Figure 3.8. We establish a baseline for *D65* using the sigmoidal [20]. Without further constraints, our method produces smooth metamers to the baseline. We then show two mismatched upliftings under *FL11*, illustrating the variety our method enables (note the ΔE_{00} measures). We further show that a user can generate color-constant upliftings for *FL11*, while mismatching for *D65*.

We next test recovery of hyperspectral data, sampling textures from the *HyTexila* dataset [76] in *Leaf*, *Textile*, *Stone* and *Wood* categories. Each 1024^2 texture stores 186 spectral channels $(400 - 1000\text{nm})$. We acquire color textures under the illuminants used in subsection 3.5.1, and perform renders for *D65* using the sigmoidal [20]. We seed our method with color images under *FL2*, *FL11*, and *LED-RGB1*. While we expect to avoid metameric failure, roundtrip error is likely for secondary color systems, as constraints are relaxed to preserve the convex hull.

Figure 3.7 shows roundtrip error and recovered reflectances. Both methods correctly handle *D65*, but show mismatching in some situations. Our method's constraint fitting varies in quality between images; *Leaf*/*Wood* textures are recovered without perceptible error, while *Textile*/*Stone* uplifts show mismatching under *FL11*, as our method fails to correctly fit a number of outlier texels (e.g. *Textile*; $\Delta E_{00} \leq 4.56, \mu = 0.95$). The sigmoidal, in comparison, reconstructs the smooth reflectances of *Wood*/*Stone* textures, but produces visible metameric failure for *Leaf*/*Textile*.

3.5.3 Memory and runtime

We compare memory and runtime of the methods used in subsection 3.5.2. We render a 1024^2px image at $256\ spp$. (averaging ten renders), s.t. a screen-filling render of the *Wood* texture is generated. All methods use single-precision, though we speculate that half-precision suffices for parts of each method. For our method, we measure for $n = \{4, 8, 16\}$ vertices. All choices gave a valid convex hull.

Table 3.1 lists results for each method. Results are as expected; the sigmoidal shares texture reads across four wavelengths, and is as fast as trichromatic rendering. Our method requires five reads per four wavelengths (one for MVCs, four for wavelength reflectances), whose size depends on vertex count. Note the slight overhead for increased counts. All methods compare favorably to the hyperspectral, as excessive memory consumption makes this intractable for large scenes.

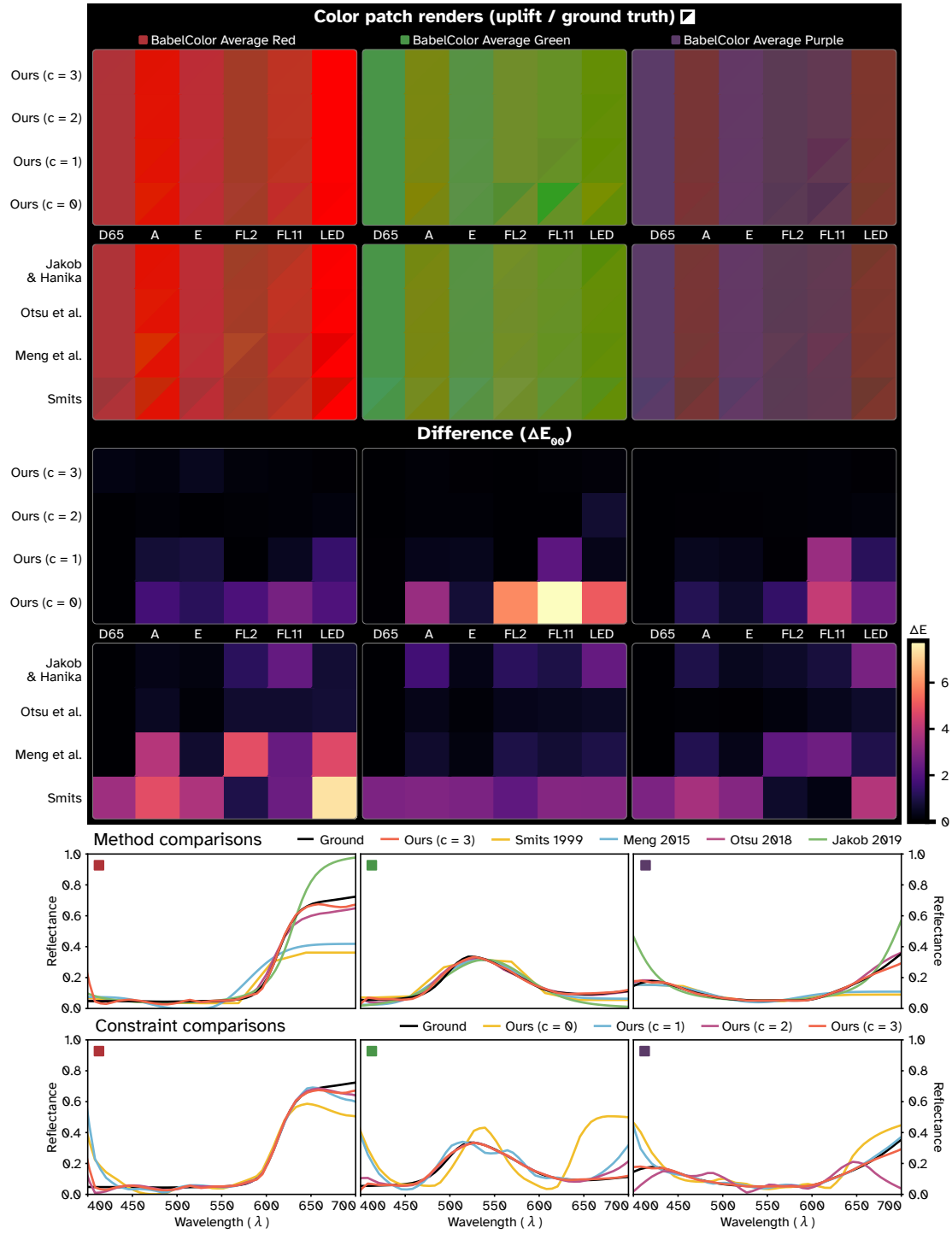


Fig 3.6 Single-reflectance recovery for samples from a *BabelColor Average* dataset [59]. We display roundtrip and ground truth for uplifting methods (top). We constrain our method for *FL2*, *FL11*, and *LED-RGB1*, in addition to *D65*. We further show CIE LAB ΔE_{00} of each roundtrip (center). We finally compare uplifted reflectances and different constraint sets (bottom).

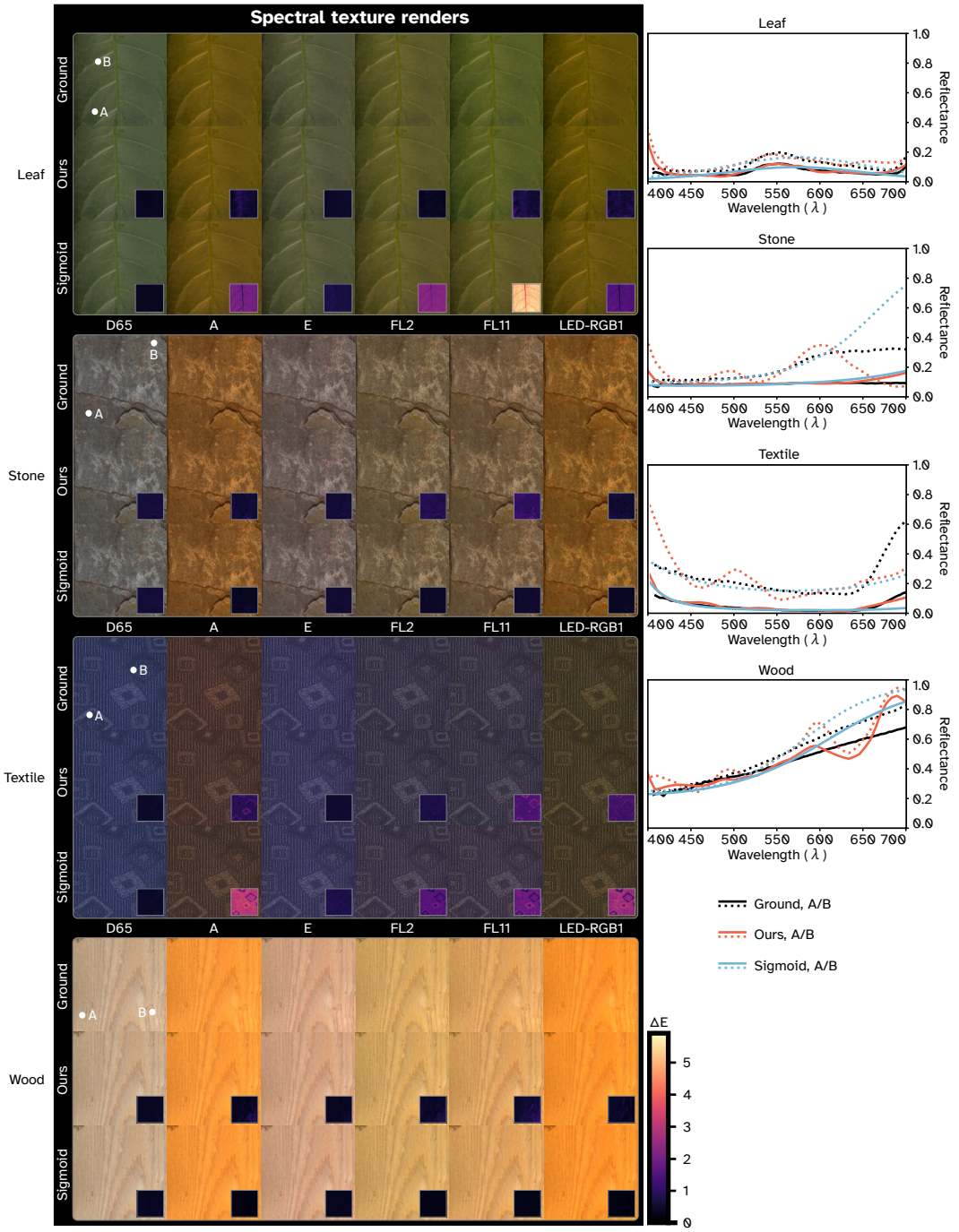


Fig 3.7 Multiple reflectance recovery for ground truth spectral images from the HyTexila dataset [76]. We constrain our method for *FL2*, *FL11*, and *LED-RGB1*, in addition to *D65*. We show roundtrip results and ΔE_{00} for our method and the sigmoidal of Jakob and Hanika [20] (left), and compare pairs of uplifted reflectances for each image (right).

3.6 Discussion

In the following, we reflect on prior results and several aspects of our method.

Reflectance recovery Our method's ability to recover the right metamers depends on secondary constraints (subsection 3.5.1). While recovery for *D65* is without error, results for $c \geq 2$ and the shapes of recovered spectra are similar to those of Otsu et al. [19]. As their approach requires no secondary constraints, this bears discussion. In short, their method restricts the PCA basis to a representable set through clustering and use of $m = 3$ functions. This leads to recovery of low-banded reflectances reproducing the *BabelColor Average* dataset [59] particularly well, but remains a 1-to-1 mapping. If we explicity select metamers in Figure 3.6, their method mismatches, arbitrarily favoring our method. This is not a failure of either method, instead demonstrating the necessity of constrained uplifting. To produce similar results to Otsu et al. [19] without secondary constraints, we could apply their clustering approach, using a different basis set across convex hull vertices.

During hyperspectral texture recovery (subsection 3.5.2), our method recovers the *D65* input, but mismatches in a number of texels for e.g. *FL11*. Vertex constraints are fitted on a sampled subset of the image. We relax this fitting s.t. the convex hull is preserved. As the hull specifies a particular convex combination for each sample, this implies constraints can at times not perfectly fit all samples. The reconstruction is hence imperfect, though error is mostly imperceptible. The sigmoidal [20] provides good reconstruction of *Wood* and *Stone* textures, as the function's shape matches their reflectances. However, this method shows full mismatching for *Leaf*, where recovered reflectances do not resemble the ground truth.

Basis function restrictions We employ $m = 12$ basis functions in tests, which exceeds the minimum [62], but enables metameric control. This flexibility implies that a low number of constraints admits many solutions, making results nondeterministic. To avoid this behavior, one could vary m depending on the context. We leave this as future work, but if we can establish in which volume (Figure 3.5) a constraint lies (preferably without solving for volumes), this enables reflectance recovery using the minimum required basis, while allowing artists to leverage more bases when necessary.

We further note that, as our basis uses common material data, reproduction is not guaranteed for all materials. If necessary, our toolkit supports loading bases that target specific material classes.

Convex hull restrictions Our method has a notable limitation; image segmentation through a convex hull restricts the set of available upliftings. Consider an input image with two identical texels. No constraint can separate these texels into separate metamers, as they share weights and thus upliftings. If the input were

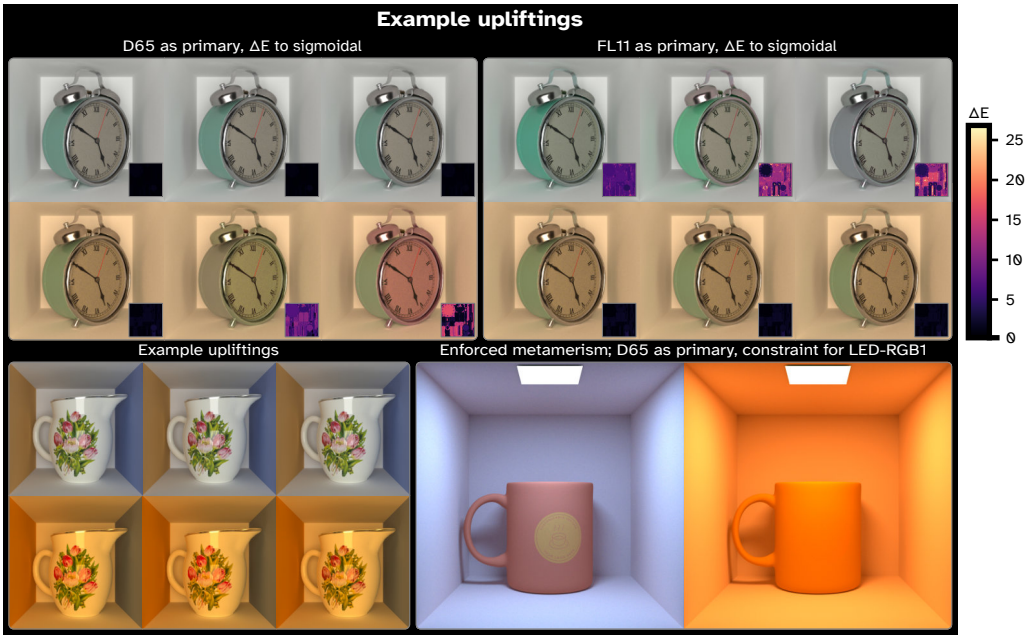


Fig 3.8 Example upliftings. Using one-to-one uplifting produces one reflectance. We uplift colors under *D65* which we then constrain to mismatch for *FL11* (top-left), and the exact inverse (top-right). We show false color textures to illustrate differences to the sigmoidal of Jakob and Hanika [20]. We further show upliftings targeting only part of a texture (bottom-left), as well as enforced metameric behavior (bottom-right).

separated into partial sets of independently uplifted texels, this can be handled. Alternatively, given an acquired image, one can start with a color system where texels are different. Then it is possible to produce metameric matching (e.g., the mug in Figure 3.8). This indicates that the choice of input color system is influential, and can be restrictive.

3.7 Conclusion

Our novel method for spectral uplifting of RGB textures is controllable and allows a user to define material appearance under different illuminants simultaneously. It generates reflectances from a small number of metameric mismatching constraints and uplifts through a simple interpolation. The latter results in a compressed representation for spectral textures with minimal roundtrip error.

Previous uplifting without control cannot target metameric behavior or color constancy. Integrating our work into content pipelines can help with such issues, and we have published our toolkit to support widespread use.

4 Controlled Spectral Uplifting for Indirect-Light-Metamerism

Abstract

Spectral rendering has received increasing attention in recent years. Yet, solutions to define spectral reflectances are mostly limited to uplifting techniques which deterministically augment existing RGB inputs. Only recently has uplifting been able to ensure a certain surface appearance under direct illuminants. Yet, prior work in this area limits artist expressiveness and is not well suited for designing the appearance of a scene, as indirect illumination is ignored entirely.

In this chapter*, we present an uplifting technique with fine-grained spectral appearance control under direct and indirect illumination, even enabling the placement of spectral constraints in a specific scene. Our approach allows for a flexible authoring process, and solves for the resulting spectra efficiently. Additionally, we show that our method's memory overhead during rendering is kept small, by introducing a compact spectral texture format.

* This chapter is based on "*Controlled Spectral Uplifting for Indirect-Light-Metamerism*", previously published in SIGGRAPH Asia Conferece Proceedings (SIGGRAPH Asia 2024) [77].

4.1 Introduction

Spectral rendering has started taking a critical role in production in recent years. Trichromatic systems approximate radiance and reflectance as RGB values, but spectral representations are required for accurate color reproduction under different illuminants. However, spectral surface reflectance data is challenging to acquire at the scales required for textures, and impose a significant memory impact during rendering. One common option is spectral uplifting, which generates spectra from existing RGB inputs. This process is ill-posed due to *metamerism*; different reflectances can produce the same color under a given light source. Hence, uplifting typically establishes a one-to-one mapping, e.g., opting for the smoothest reflectance, and hereby inherits problems of trichromatic rendering, as a produced reflectance might exhibit a wanted appearance under one illuminant only. Therefore, working in a full spectral pipeline remains challenging.

Recent work introduced constrained spectral uplifting [53, 78], focusing in particular on appearance control under various illuminants. Yet, in practice, there are limitations, including performance impacts, limited control, and ignored interreflections, which can significantly impact the appearance of a scene.

Our approach to spectral uplifting extends prior work, while being simple and efficient. Our method supports reproducing spectral measurements, direct-illumination and direct-observer color constraints, and, for the first time, scene constraints on color appearance under complex indirect illumination. We uplift input color data via a $\mathbb{R}^3$ convex polytope en-globing the data with specific spectra on each vertex. The polytope's interior is tessellated into simplices, and interior vertices can be added for fine-grained control. We retrieve the reflectance of a particular color value by localizing the surrounding simplex, and performing interpolation between the spectra of the simplex' vertices. We further extend prior work on metamer mismatch volumes [22, 42, 43], incorporating estimated light transport to handle indirect illumination in a scene environment. Using a small basis, we "bake" the uplifting into an efficient texture format, which supports localized constraints applied to parts of the scene, and is practical for rendering. Finally, we make our source code available online⋆.

In short, we make the following contributions:

- A controllable spectral uplifting method via several types of constraints;
- A solution for metamer color control, particularly indirect light;
- A compact spectral texture format suitable for spectral rendering.

In the following, we cover related material (Section 4.2) and our method (Section 4.3). We evaluate roundtrip error of relevant methods and apply spectral authoring (Section 4.4), discuss our findings (Section 4.5), and conclude (Section 4.6).

⋆ https://graphics.tudelft.nl/indirect_uplifting

4.2 Background

Color Theory While spectra consist of various wavelengths, human color perception builds mainly upon three sensors. Consequently, colors are equated to a sensor trio observing the spectral distribution of light, described by observer functions [37]. Denoting three observer functions as e, and given illuminant distribution i, we express the response to surface reflectance r under direct illumination as

Eq 4.1
$$\Phi(r) = \int_{\Lambda} e(\lambda)\ r(\lambda)\ i(\lambda)\ d\lambda,$$

where Λ describes typically the visible spectrum (e.g. $360 - 830\mathrm{nm}$). This is formalized in the CIE XYZ standard observers and derived color spaces such as sRGB. A *color system* combining observer and illuminant describes a linear transformation $\Phi(r) : \mathbb{X} \to \mathbb{R}^3$. Of particular interest is the valid region of responses $\{r \in \mathbb{X} \mid \Phi(r) \neq 0\}$, which forms a convex region called the *object color solid*.

The illuminant i describes radiant energy over wavelength, while reflectance r describes a surface's efficacy in reflecting said energy. Illuminants are positively unbounded and vary in shapes dependent on the underlying processes. Reflectances are $[0, 1]$-bounded and are generally low-banded in the visible spectrum. This property holds for most pigments, but not for structural colors [48]. We, as most related work, restrict ourselves to smooth reflectances.

Metamer Mismatching We briefly cover metamerism and metamer mismatching; for extended overviews, please refer to [42, 43]. Given some color system, consider the problem of inverting Equation 4.1, i.e. $\Phi^{-1}(\Phi(r)) = r$. It is ill-posed; as it is typically a set of reflectances:

$$\Phi^{-1}(\Phi(r)) = \{\ r' \in \mathbb{X} \mid \Phi(r) = \Phi(r')\ \},$$

i.e. many reflectances achieve a particular color signal. This *metamer set* is convex. Metamers produce the same signal under Φ, but in a secondary color system Ψ (differing in observer or illuminant) they produce different responses. Mapping the set to Ψ, we find a non-singleton color solid called a *metamer mismatch region*. We describe the method of Mackiewicz et al. [22] to find mismatch region boundaries, as we extend this method to incorporate indirect illumination (subsection 4.3.3).

Consider mapping $\mathcal{T} : \mathbb{X} \to \mathbb{R}^6$, $\mathcal{T}(r) = (\phi, \psi)$, where $\phi = \Phi(r)$, $\psi = \Psi(r)$ form color signals in two color systems. For given signal ϕ, the set of *mismatched signals* under Ψ is a cross-section of $\mathcal{T}$, i.e.:

$$\mathcal{M}(\phi, \Phi, \Psi) = \{\ \psi \in R^3 \mid (\phi, \psi) \in \mathcal{T}\ \},$$

where we simplify notations by identifying $\mathcal{T}$ with its image. While this region's interior can be complex, boundary spectra are unique step-wise functions con-

sisting only of transitions between zeroes and ones. These *optimal spectra* are extrema [44, 46]. Finding the mismatch boundary $\delta\mathscr{M}$ thus reduces to extremizing spectra mapped to $\mathscr{T}$, subject to $\Phi(r) = \phi$. In practice, one spherically samples unit vector $\hat{u} \in \mathscr{R}^6$, and projects the color system spectra in $\mathscr{T}$ along $\hat{u}$. In the discrete case, reflectance r is represented as a k-dimensional vector, Φ, Ψ are $k \times 3$ color system matrices, and we can solve for the boundary with a linear program:

Eq 4.2
$$\max_{r \in \Phi^{-1}(\phi)} \left(\begin{bmatrix} \Phi \\ \Psi \end{bmatrix} \hat{u} \right)^{\mathsf{T}} r$$

This produces a discrete reflectance on the boundary $\delta\mathscr{M}(\phi, \Phi, \Psi)$.

Spectral Uplifting A recent overview of the scope of spectral uplifting is given by Weidlich et al. [49]. Uplifting is having a known color signal ϕ and then finding a metamer $r \in \Phi^{-1}(\phi)$. Across different methods, three criteria are identifiable: smoothness of produced reflectances, boundedness, and roundtrip error.

The earliest approaches to spectral uplifting are now of mostly theoretical interest, as they produce blocky distributions or break boundedness [16, 17]. The later approach of Meng et al. [18] precomputes spectra on a grid spanning the xy-chromaticity plane, uplifting colors through interpolation of these spectra. It produces smooth reflectances, but requires scaling for values above or below the plane, introducing errors. Otsu et al. [19] address this by clustering sets of spectra into a spatial hierarchy covering the xy-chromaticty plane. Inside clusters, the authors store localized bases for spectrum recovery. While efficient, this introduces discontinuities between clusters, uplifting gradients poorly. More recently, van de Ruit and Eisemann [78] precompute spectra on a convex polytope enclosing an input texture, ensuring correct recovery if such a polytope can be found.

A separate class of techniques started with Jakob and Hanika [20], who demonstrate a low-dimensional parameterization of a sigmoidal function space, for which they precompute coefficients across a three-dimensional color lookup table. Subsequent function reconstruction is fast, and produces smooth reflectances with minimal roundtrip error. This approach has been extended to handle out-of-gamut spectra [51, 52]. Afterwards, a more complex, Fourier-space approach is introduced by Peters et al. [21], addressing issues of the sigmoidal.

As previously stated, most methods establish 1-to-1 mappings, associating a specific metamer with a color. Two techniques differ in this aspect. Tódová et al [53, 54] extend the Fourier-space representation [21], seeding that method's coefficients s.t. specific spectra are reproduced. Van de Ruit and Eisemann [78] uplift based on specified color constraints. While their approach targets specific color behavior, control is limited. We detail these limitations later (subsection 4.3.1), as we extend their work. Our solution is the first to integrate indirect illumination.

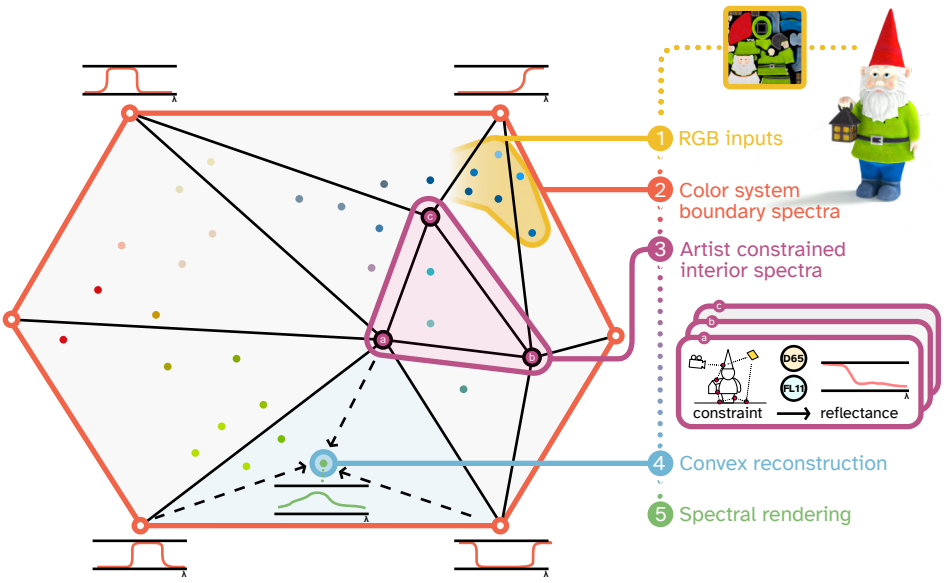


Fig 4.1 **Uplifting prodecure.** We treat input colors as $\mathbb{R}^3$ positions and sample an enclosing boundary on the input's color system, while artist-specified constraints then specify spectra in the interior. Boundary and interior spectra are connected using a Delaunay tessellation, and inputs are then uplifted using barycentric interpolations inside the tessellation's simplices.

4.3 Method

We now present our constrained spectral uplifting. We first cover our method's foundation (subsection 4.3.1), followed by the basic uplifting procedure (subsection 4.3.2). We then derive a color system to constrain the uplifting under indirect illumination (subsection 4.3.3), and finally specify a practical texture format for rendering (subsection 4.3.4).

4.3.1 Foundation

A color system (Equation 4.1) describes a linear transformation, preserving convexity. Let $r_1, \dots, r_n$ be n reflectances with corresponding mappings $\Phi(r_1), \dots, \Phi(r_n)$. If we combine these mappings using convex weights $w_1, \dots, w_n \ : \ \forall_i w_i \geq 0 \cap \sum_i w_i = 1$, we observe:

Eq 4.3
$$\sum w_i \Phi(r_i) = \Phi\left(\sum w_i r_i\right),$$

i.e., the color signal of linearly combined reflectances equals the linear combination of the corresponding color signals. When interpolating two metamers, the result itself is a metamer - but it also holds for arbitrary reflectances. This principle is employed in most prior work. Van de Ruit and Eisemann [78] note that, minimally, interpolation of $\mathscr{R}^3$ color signals must occur within a 3-simplex, as planar methods

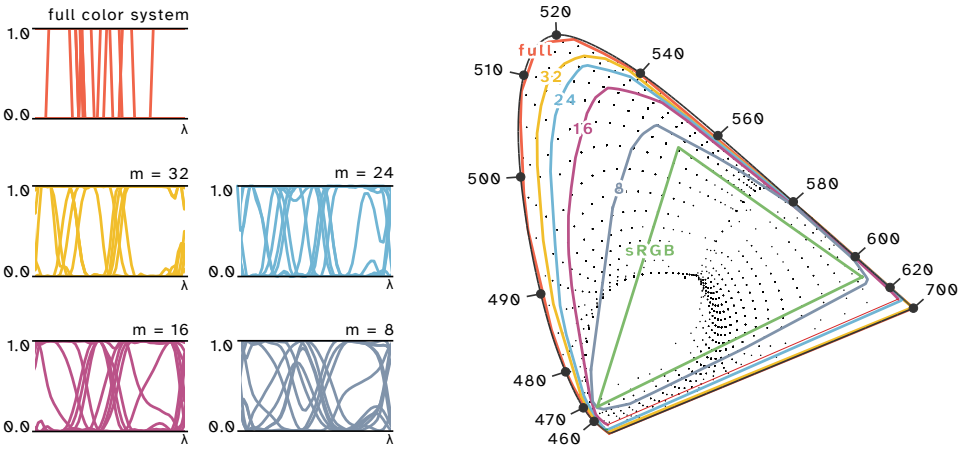


Fig 4.2 **Color system support.** We show sample boundary spectra and the support of our basis with $m = 32, 24, 16, 8$ components. We further show support of a discrete system ($k = 64$) of sampled MacAdam limits, as well as the sRGB gamut, which is enclosed for all choices of m.

unavoidably struggle with error [18, 19, 50]. However, instead of a simplex, they solve for a complex polytope enclosing an input RGB texture. Uplifting then reduces to a linear mixture of the polytope's uplifted vertices. The authors enable targeted uplifting by constraining the vertex spectra, but this is limited in effect; vertices necessarily lie on the polytope, away from the input, and therefore constraints never affect or modify the input's uplifting directly. In contrast, our method allows for interior constraints, which can directly affect the uplifting of specific RGB inputs.

4.3.2 Tessellated Color System

Convex reconstruction of a polytope interior avoids roundtrip error, which is why we also use this as our foundation. However, we select a polytope that describes the color-system boundary, which encloses all color inputs. Given the polytope vertices, we can construct a Delaunay tessellation, which results in a set of 3-simplices, each associated with four vertices. Each vertex contains a spectrum producing the color encoded by the vertex position under the illuminant of the color system. We detail spectrum generation further below. Uplifting a color input then reduces to (a) localizing the enclosing simplex for the color within the tessellation, and (b) a barycentric interpolation of the simplex' associated spectra (Equation 4.3). Further, we can constrain the interior by inserting vertices into the tessellation, which then affect the uplifting. Figure 4.1 shows an overview of the uplifting procedure.

Boundary spectra Reflectances on the color-system boundary are extrema, being the MacAdam limits. Interpolation between such saturated distributions results in correct but physically implausible spectra. We therefore find a smaller boundary formed by smooth spectra as outlined below.

Like prior methods, we use a weighted PCA basis to construct low-banded spectra [19, 62–65, 78]. We use a dataset of $\sim 41M$ measured reflectances $(400-700\text{nm})$ from a variety of materials [47]. Let reflectance r be a discretized k-dimensional vector. We apply PCA to the dataset and retain the first m principal components as a $k \times m$ matrix B. Then we represent a reflectance as $r = Bw$, where w are m-dimensional coefficients. Though $m = 3$ suffices to reproduce most colors [62], we retain $m > 3$ as in prior work [78]. Otherwise, if the system were fully determined, we would eliminate the ability to output metamers.

To generate a boundary within this basis, we employ the method of Mackiewicz et al. [22]. We spherically sample a unit vector $\hat{u} \in \mathscr{R}^3$, and project the $k \times 3$ discretized color system Φ onto $\hat{u}$. Maximizing this projection necessarily results in a position on the system boundary. Expressed in our basis, this becomes:

Eq 4.4
$$\max_{w} (\Phi\hat{u})^{\mathsf{T}} Bw, \quad \text{with } \forall_i 0 \le (Bw)_i \le 1.$$

As PCA acts as a dimensionality reduction, resultant spectra become low-banded, enabling their use for interpolation. Figure 4.2 illustrates color system support and boundary spectra for choices of m.

Interior spectra In the polytope interior, we allow user-specified constraints. Each provides a color signal - in effect a vertex position - and an associated reflectance, uplifted or provided. We detail three supported constraint types.

1. Measurement: Given a concrete spectral reflectance r, we project it into the PCA basis, which gives a w, minimizing $||Bw - r||$, and then insert vertex $\Phi(Bw)$. Similar to Tódová et al. [53], we can reproduce the representations of spectral measurements. We investigate full spectral texture reproduction in subsection 4.4.2.

2. Direct color: Given input of n color constraints $\{\psi_1, \dots, \psi_n\}$ under n color systems $\{\Psi_1, \dots, \Psi_n\}$, the linear program

Eq 4.5
$$\min_{w} ||Bw||, \text{ with } \forall_j \Psi_j Bw = \psi_j \text{ and } \forall_i 0 \le (Bw)_i \le 1$$

produces coefficients for a constraint-satisfying metamer, which we insert as vertex $\Phi(Bw)$. Following van de Ruit and Eisemann [78], we restrict inputs to the intersection of relevant mismatch regions.

3. Indirect color: Given a scene with observer, and an observed surface position with surface reflectance r, we enable constraining the observed color ψ at this position under an indirect color system $\Psi(r)$. We derive this color system in the following (subsection 4.3.3).

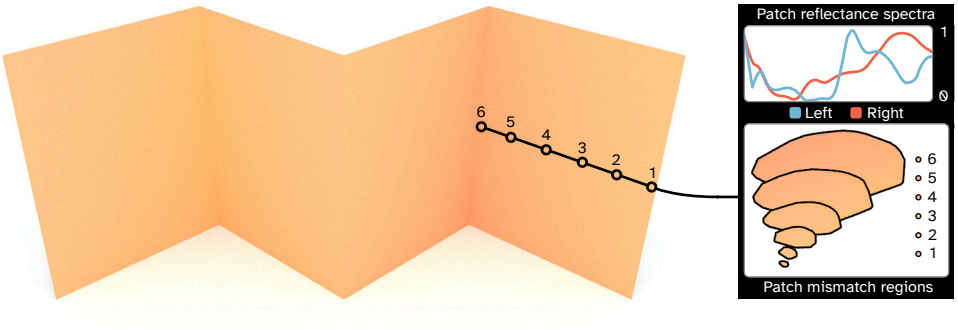


Fig 4.3 Shown are two folded patches on a white plane, illuminated under $D65$. While identical under direct light, these concave surfaces differ in appearance due to interreflections. We show the used reflectance spectra (top-left), and indirect mismatching regions generated at equidistant points on a patch (bottom-left).

4.3.3 Indirect Color System

While prior constraint types enable uplifting control, they can only account for mismatching in a linear form. Illuminant- and observer-induced mismatching, in this way, occur when directly observed. Yet, in complex scenes, illuminant-induced mismatching can occur due to complex light transport and interreflections (Figure 4.3). To control this effect, we derive a convex formulation of a non-linear color system, w.r.t. a constraint reflectance r at some surface position in a scene.

In the following, we estimate the indirect light transport from this point and then factor out r. We use this factorization to formulate a maximization that finds the metamer mismatch boundary under indirect illumination.

Path-integral formulation For an environment with a particular constrained reflectance r at a given surface point, we define a color system over a per-wavelength incident radiance I_r and observer function e:

Eq 4.6
$$\Psi(r) = \int_\Lambda e(\lambda)\, I_r(\lambda)\, d\lambda.$$

The radiance measure I_r can be expressed using a path-integral formulation of light transport [79]:

Eq 4.7
$$I_r(\lambda) = \int_\Omega f_r(\bar{x}, \lambda)\, d\mu\bar{x}$$

where μ forms a measure over samples in the path domain Ω, and $f_r : (\Omega, \Lambda) \to \mathscr{R}$ denotes the per-wavelength measurement contribution along a light path of length n, $\bar{x} = \{x_1, \dots, x_n\}$. This contribution describes light throughput (geometric terms, cosine attenuation, bidirectional reflectance, illuminant) along the path.

Let $\{r_1, \dots, r_{n-1}\}$ be underlying reflectances at path vertices not on a light source. We can factor out these reflectances using a secondary function $f_0 : \Omega \to \mathbb{R}$, which

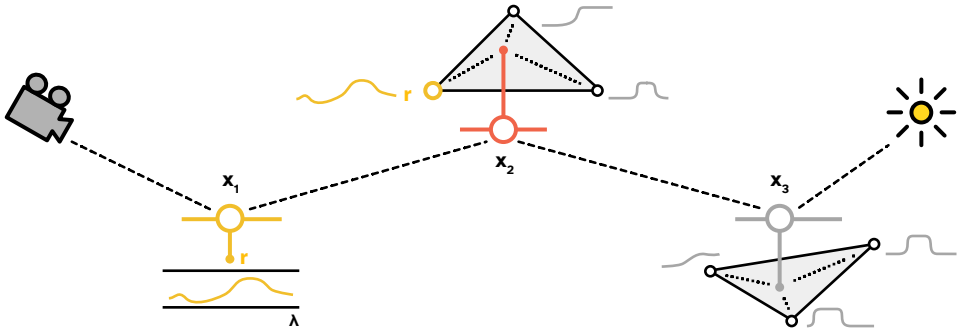


Fig 4.4 **Formulation.** We show a light path $\{x_1, x_2, x_3\}$, and constraint reflectance r at surface x_1. Further reflectances are expressed as convex combinations of spectra in simplices from our tessellation, which can contain r.

is independent of reflectances ($\forall_{r \in \mathbb{X}}\ r(\lambda) = 1$). We employ f_0 to express

Eq 4.8
$$f_r(\bar{x}, \lambda) = f_0(\bar{x}) \prod_{i=1}^{n-1} r_i(\lambda),$$

which implies that BRDFs in our formulation need to allow for associative rearrangement of surface reflectances.

Following this, we can express path reflectance r_i w.r.t. the constraint reflectance r. Recall that uplifting is the convex combination of four spectra, associated with vertices of a 3-simplex. As r is one vertex in our tessellated color system, each encountered spectrum r_i on surfaces along the path either uses r or not. Figure 4.4 illustrates an example.

Leveraging this, we denote simplex spectra and associated convex weights as $(s_1, a_1), \ldots, (s_4, a_4)$, such that $r_i = \sum_{j=1}^{4} s_j a_j$. It follows that:

$$r_i = \begin{cases} a_k r + \sum_{j \neq k}^{4} s_j a_j & (\exists_k\ s_k = r) \\ 0r\ + \sum_{j=1}^{4} s_j a_j & (else) \end{cases} \quad \rightarrow \quad r_i = a_i r + w_i$$

where $a_i \geq 0$ is either zero or a convex weight, and w_i thereby sums the weighted remainder spectra. This enables us to expand Equation 4.8 as:

Eq 4.9
$$\begin{aligned} f_0(\bar{x}) \prod_{i=1}^{n-1} (a_i r + w_i) = f_0(\bar{x}) (\quad & r^{n-1}(a_1 a_2 \cdots a_{n-1}) \\ + \quad & r^{n-2}(w_1 a_2 \cdots a_{n-1} + a_1 w_2 \cdots a_{n-1}) \\ + \quad & r^{n-3}(w_1 w_2 a_3 \cdots a_{n-1} + \cdots) \\ & \vdots \\ + \quad & r^0(w_1 w_2 \cdots w_{n-1})) \end{aligned}$$

For a power $b < n$, we define t_b as the coefficient of r^b in Equation 4.9. The

contribution then simplifies to a truncated power series:

$$f_r(\bar{x}, \lambda) = f_0(\bar{x}) \sum_{b=0}^{n-1} t_b(\bar{x}, \lambda)\ r^b(\lambda)$$

We now revisit the path-integral formulation (Equation 4.7). Given the complexity of light-transport, we apply Monte Carlo integration, introducing the estimator

$$I_r(\lambda) \approx \hat{I}_r(\lambda) = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\bar{x}_i, \lambda)}{p(\bar{x}_i)},$$

converging to the expected value as $N \to \infty$. Here $p(\bar{x}_i)$ is the probability density of sample $\bar{x}_i$. Given a known, finite length n across paths - which we note as a potential approximation - the estimator becomes

Eq 4.10

$$\begin{aligned}
\hat{I}_r(\lambda) &= \frac{1}{N} \sum_{i=1}^{N} \frac{f_0(\bar{x}_i)}{p(\bar{x}_i)} \sum_{b=0}^{n-1} t_b(\bar{x}_i, \lambda)\ r^b(\lambda) \\
&= \sum_{b=0}^{n-1} \left(\frac{1}{N} \sum_{i=1}^{N} \frac{f_0(\bar{x}_i)\ t_b(\bar{x}_i, \lambda)}{p(\bar{x}_i)} \right) r^b(\lambda) \\
&= \sum_{b=0}^{n-1} c_b(\lambda)\ r^b(\lambda) \quad \text{, where } \ c_b(\lambda) := \frac{1}{N} \sum_{i=1}^{N} \frac{f_0(\bar{x}_i)\ t_b(\bar{x}_i, \lambda)}{p(\bar{x}_i)},
\end{aligned}$$

producing a power series as a simplified expression.

In practice, we estimate the coefficient spectra $c_0, \ldots, c_{n-1}$ by accumulating incident radiance along N paths and, for each path expressed thus, factoring out b interreflections of constraint reflectance r. Given these coefficients, we can employ Equation 4.10 to express light transport under mismatching constraint reflectances. This follows prior approximations of interreflections, which treat surfaces as a finite number of patches [80].

Mismatch volume boundary We next employ Equation 4.10 to specify an estimate of an indirect color system (Equation 4.6), which becomes

$$\Psi(r) \approx \hat{\Psi}(r) = \int_{\Lambda} e(\lambda)\ \hat{I}_i(\lambda)\ d\lambda = \int_{\Lambda} e(\lambda) \sum_{b=0}^{n-1} (c_b(\lambda)\ r^b(\lambda))\ d\lambda,$$

which forms a non-linear system.

We build upon the method of Mackiewicz et al. [22] to determine mismatch boundary $\delta\mathcal{M}$. First, we discretize the color systems $(3 \times k)$, specifically the uplifting's color system Φ with known signal ϕ, and define discretized indirect color system spectra as $\tilde{\Psi}_0, \ldots, \tilde{\Psi}_{n-1}$, where $\Psi_b = e \circ c_b$; here $\circ$ denotes a component-wise multiplication. We then spherically sample a unit vector $\hat{u} \in \mathbb{R}^6$, along which we project color system spectra. As with the linear form (Section 4.2), maximizing in the direction of this projection results in a position on the region's boundary. We denote $\hat{u} := (\alpha, \beta)$, where $\alpha, \beta \in \mathbb{R}^3$, forming

Eq 4.11
$$\max_{r \in \Phi^{-1}(\phi)} (\Phi\alpha)^{\mathsf{T}} r + \sum_{b=0}^{n-1} (\tilde{\Psi}_b \beta)^{\mathsf{T}} r^b$$

which adopted to our basis becomes

$$\max_{w} (\Phi\alpha)^{\mathsf{T}} (Bw) + \sum_{b=0}^{n-1} (\tilde{\Psi}_b \beta)^{\mathsf{T}} (Bw)^b$$
$$\text{with } \Phi^{\mathsf{T}}(Bw) = \phi \text{ and } \forall_i 0 \le (Bw)_i \le 1.$$

We show mismatch volumes generated over a patch with varying interreflections in Figure 4.3. Note that this problem is convex as the solution spans r, which is $[0,1]$-bounded, while $b \ge 0$ and $\forall_b c_b \ge 0$. However, as our basis uses negative components, this convexity does not necessarily hold. We may thus find mismatch boundaries that are interior to the exact boundary, which is sufficient for our application in practice.

Indirect reflectance generation We next generate a metamer in the mismatch volume. Given indirect color constraint ψ and discretized color system spectra $\hat{\Psi}_1, \ldots, \hat{\Psi}_{n-1}$, we extend Equation 4.5, solving:

$$\min_{w} \|Bw\|, \; with \; \forall_i 0 \le (Bw)_i \le 1$$
$$\text{with } \phi = \Phi^T (Bw) \text{ and } \psi = \textstyle\sum_b^{n-1} \hat{\Psi}_b^T \, (Bw)^b.$$

Note that, as the mismatch volume we present to the user to specify ϕ is a color solid, we can avoid this minimization. Instead, we can tessellate this solid in $\mathbb{r}^3$, localize ψ's enclosing simplex within the tessellation, and perform interpolation of the simplex' vertex coefficients. This w then encodes the wanted metamer Bw.

4.3.4 Texture format

Our uplifting is a reconstruction localized to a simplex in the tessellation. A direct approach to storing an uplifted input, is the index to its enclosing simplex, together with three convex weights, as the fourth can be deduced ($\sum a_i = 1$). Given that weights are in $[0,1]$ and the number of simplices in the tessellation is low, a low-bit RGBA texture typically suffices. However, users may define conflicting constraints, as we discuss in Section 4.5, implying additional operations during rendering.

Instead, we preprocess and "bake" our reflectances, representing each reflectance in the orthonormal basis B. We already rely on coefficients in this basis throughout our entire pipeline, including the reflectances stored in vertices of the tessellation. Hence, to encode an uplifted reflectance, we find the enclosing simplex and use the interpolated basis coefficients. This representation is compact and conversion

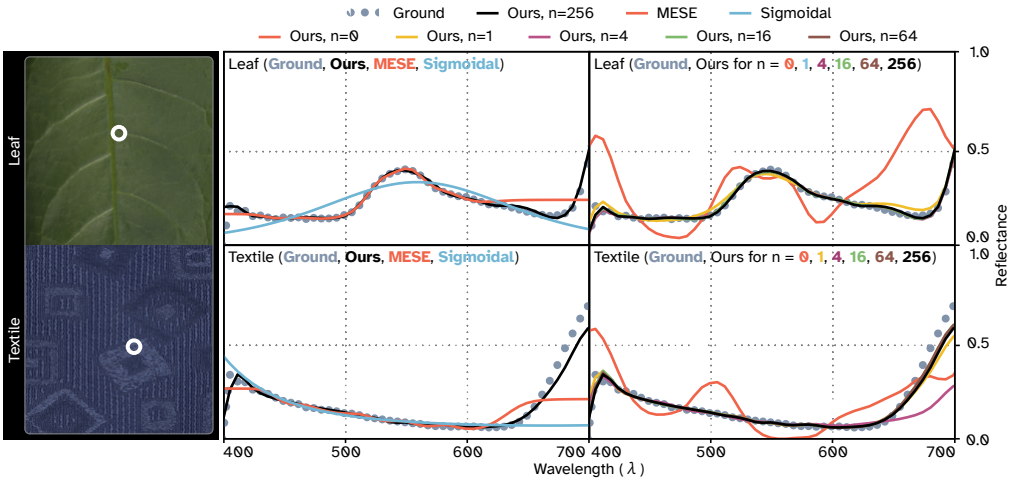


Fig 4.5 **Constrained reflectances.** We show sample spectra from constrained spectral texture reconstruction (Figure 4.9) for our method ($m = 12$), the *sigmoidal* [20] and the bounded *MESE* [21]. Our uplifting is constrained with n random samples from the input texture.

is easily parallelized. We scale our basis to a $[-1, 1]$ boundary on the projection of PCA inputs; recovered coefficients are thus $[0, 1]$-bounded. This enables a fixed-point representation; in practice, we store 128 bits per pixel, packing 8, 12 or 16 coefficients at 16, 10 or 8 bits respectively. Note that texture filtering is applied to coefficients after unpacking; the filtered result is then used for uplifting. We evaluate variants of our representation in subsection 4.4.1.

4.4 Evaluation

In the following, we discuss implementation, evaluate uplifting quality (subsection 4.4.1) and spectral texture recovery (subsection 4.4.2). Afterwards, we demonstrate the indirect color system (subsection 4.4.3).

Our implementation relies on a sequential quadratic programming [81] algorithm in the *NLopt* framework [82] for the constrained optimization and *Qhull* [83] for the Delaunay tessellations. Uplifting and rendering use *OpenGL*, and we employ continuous wavelength hero sampling [13]. During uplifting, discrete spectra use $k = 64$ bins to handle high-frequency illuminants. For color solid sampling, we use 128 spherical samples in all cases. Our method supports any spectral range, but uses $400 - 700\text{nm}$ due to the dataset underlying our basis. A supplemental video shows real-time spectral modifications on a *RTX 3070*. When a user first places an indirect constraint in a scene, we trace *65K* paths GPU-side, and reduce to a power series CPU-side, which takes under a second. Texture baking occurs whenever a relevant constraint is edited, and takes 12.5 ms for a 4K texture.

Throughout the evaluation, we compare with the *sigmoidal* uplifting [20] and the Fourier-space bounded *MESE* [21]. For the latter, we select $m = 12$ coefficients

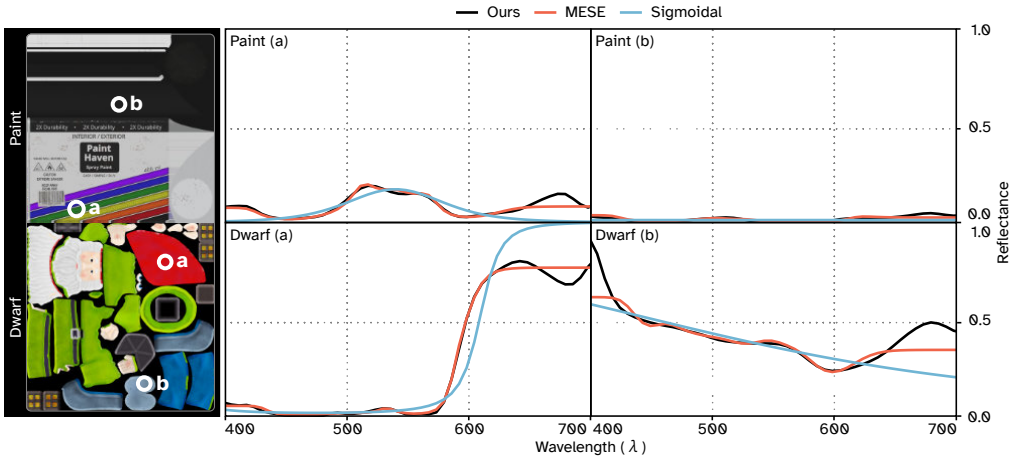


Fig 4.6 **Unconstrained reflectances.** We show sample spectra from unconstrained uplifting (Figure 4.8) for our method ($m = 12$), the *sigmoidal* [20] and the bounded *MESE* [21]. All spectra accurately reproduce the RGB input under $D65$.

(11 moments). This method has high accuracy on RGB roundtrip with fewer coefficients; however, it proved insufficient to reproduce complex metamers on mismatch boundaries. Note that the method's reconstruction time is quadratic w.r.t. the choice of m. In the authors' implementation, we use their *mirrored, warped* configuration.

We evaluate color accuracy for the different methods using CIE ΔE_{00} color difference [84]. Note the following thresholds: $\Delta E \leq 1$ implies no perceptible difference; $\Delta E \in (1, 2]$ implies a very close match; $\Delta E > 2$ implies close but visible mismatching.

4.4.1 Reconstruction of RGB data

We first evaluate unconstrained uplifting of RGB input data. We measure color reproduction of *BabelColor Average* patches [38] under $D65$. Prior methods achieve sufficient accuracy here; ideally we produce the same roundtrip error or less. For our method, we test bases with $m = 8, 12, 16$ principal components at 32 bit, and low-bit *packed* variants storing $16, 10, 8$ bits per coefficient. For the bounded *MESE*, we likewise test a variant storing 10 bits per coefficient, using the same code as our method.

Figure 4.7 shows roundtrip results for all methods and Figure 4.1 lists mean and maximum ΔE_{00}. While roundtrip of full-precision variants of our method consistently improves on prior methods, none produces perceptible error. The exception is our method's $m = 16$ packed variant, which visibly mismatches darker colors due to the low bitrate. As packed variants are intended for practical rendering we discard $m = 16$ variants in the following.

Figure 4.8 shows unconstrained uplifting of RGB textures, achieving similar error. All methods uplift correctly, though in packed variants our method avoids mis-

matching, while the bounded *MESE* struggles with darker colors. We further show uplifted spectra from these textures (Figure 4.6); evidently, the basis can introduce an oscillating behavior near spectral range boundaries, compared to the smoother outputs of prior methods. This is explained by the use of warped coefficients, which was presented by the authors of bounded *MESE*; i.e. greater precision is given to the centre of the visible range.

4.4.2 Reconstruction of spectral data

We next demonstrate recovery of hyperspectral textures. Given their size, such data is typically impractical for rendering. We approximate the textures in a compact format. We repeat a prior experiment [78], fitting textures from the *HyTexila* dataset [76]; each 1024^2 texture stores 186 spectral channels over $400 - 1000\text{nm}$ ($744MB$). We then sample $n = 0, 1, 4, 16, 64, 256$ spectra from the texture, inserting these in our tessellation as *measurement* constraints. As n increases, we expect our method to increasingly resemble the input texture. We could fit all pixels into our basis directly instead. However, as spectral texture capture is challenging, we test whether a smaller input suffices. For the bounded *MESE*, we fit per pixel, though we note that the work of Tódová et al. [53, 54] enables a compact fitting of this method.

Figure 4.6 compares outputs under standard illuminants $D65$, $FL11$, and $LED - RGB1$. All unconstrained methods correctly handle recovery under $D65$, but visibly mismatch under one of the other illuminants. Given any number of constraints, our method's output strongly improves. For $n \geq 16$, all outputs achieve mean $\Delta E_{00} \leq 1.2$, outperforming prior methods. We show several output spectra for all methods and sample counts in Figure 4.5.

4.4.3 Reconstruction of indirect color constraints

Finally, we evaluate indirect color constraining (subsection 4.3.3). We set up a simple scenario; a neutral-gray surface, illuminated by a constant $D65$ environment. The surface consists of flat and folded parts, perpendicular to an orthogonal camera. We constrain the flat part to simply reproduce the input RGB color. We then constrain the center of the fold, where interreflections occur and metameric mismatching is possible. We generate a mismatch volume for the constraint, and select constraint values that we subsequently apply to the surface.

Figure 4.10 displays results; for each position, we render with the $m = 12, packed$ variant of our method. All variants of the surface correctly reproduce the intended colors; roundtrip error remains below perceptible limits. Error on the flat patch (mean $\Delta E_{00} = 0.34$) is partially due to the low-bit representation used for rendering.

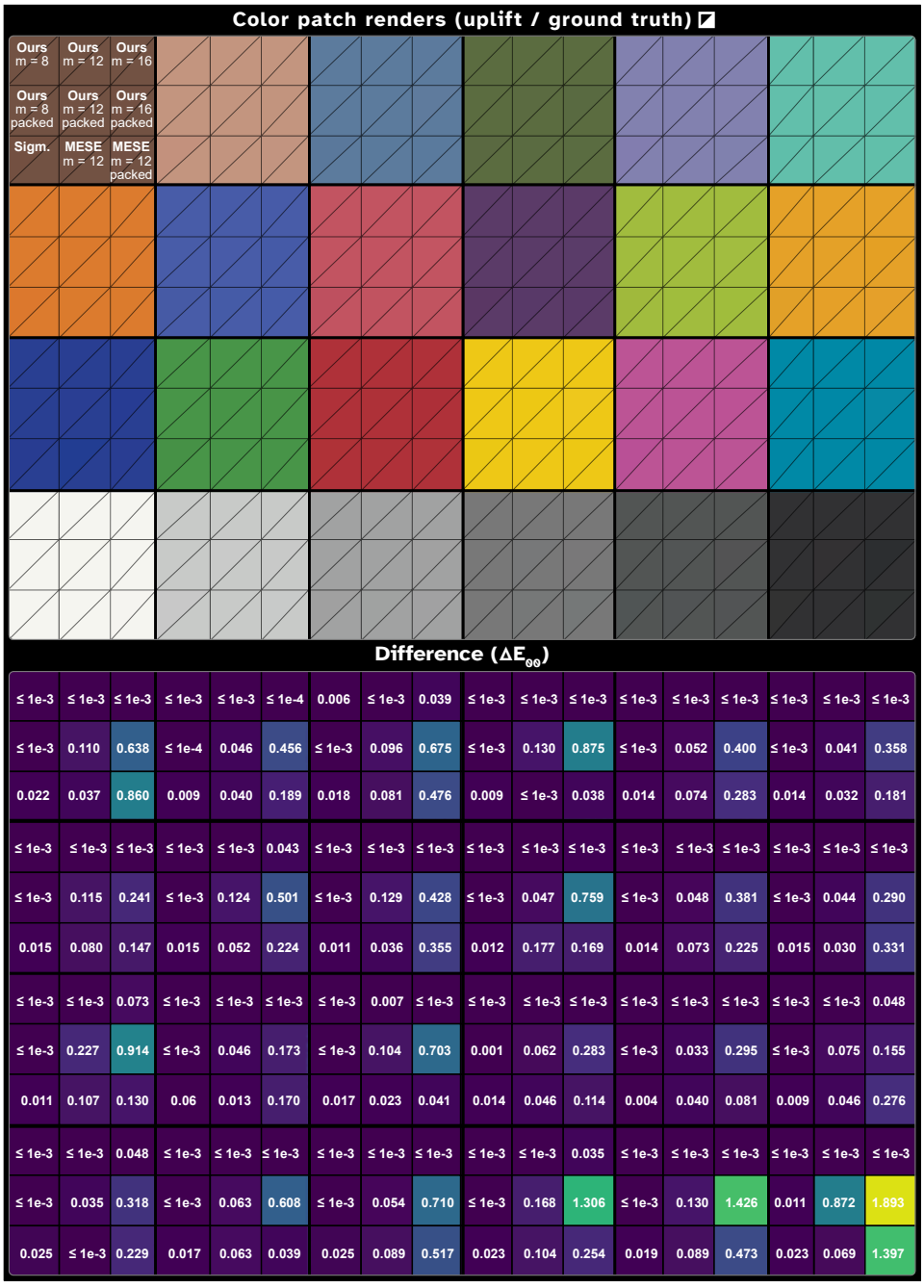


Fig 4.7 **Unconstrained color uplifting.** We uplift color patches [38] and render under $D65$. *Left.* Output for our method ($m = 8, 12, 16$), low-bit variants, the sigmoidal [20] and the bounded *MESE* [21]. *Right.* ΔE_{00} values are listed for all results.

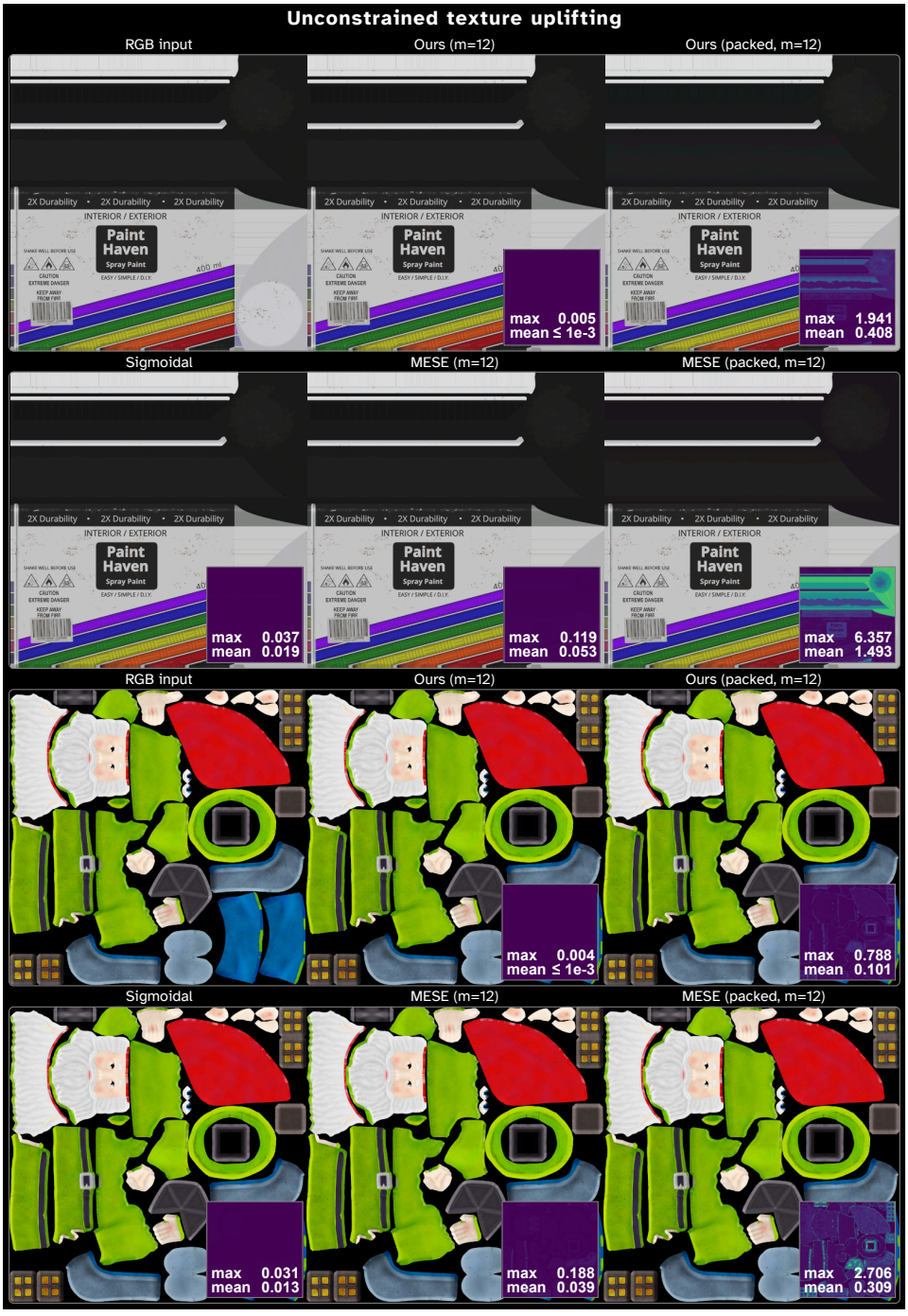
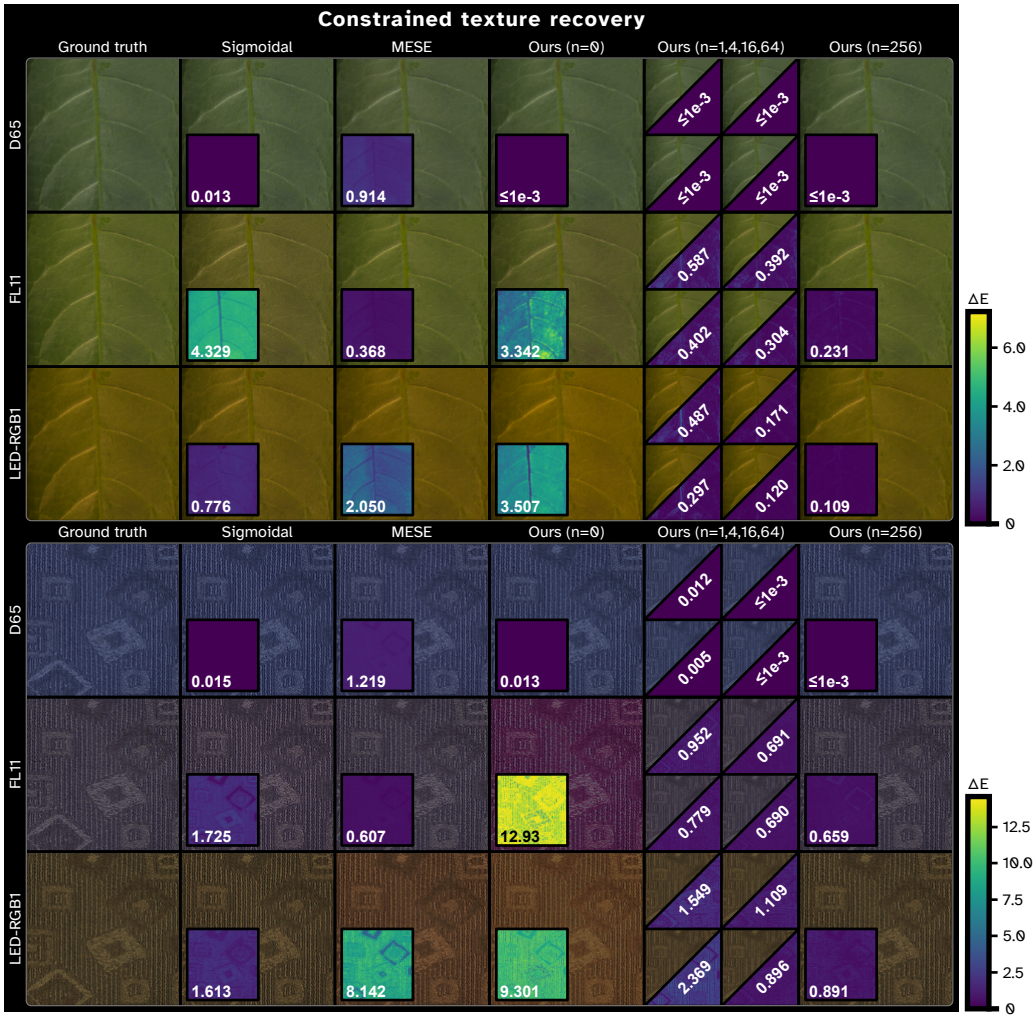


Fig 4.8 **Unconstrained texture uplifting.** We uplift RGB textures and render under $D65$. We show output for our method ($m = 12$ coefficients), a low-bit packed variant, the *sigmoidal* [20] and bounded *MESE* [21]. Mean and maximum ΔE_{00} are listed for all outputs.

	Ours (m=8)	Ours (m=12)	Ours (m=16)
$\Delta E_{00}, \mu$	0.00046	0.00056	0.01024
$\Delta E_{00}, max$	0.00588	0.00725	0.07291
	Ours (m=8, packed)	Ours (m=12, packed)	Ours (m=16, packed)
$\Delta E_{00}, \mu$	0.00048	0.11862	0.60964
$\Delta E_{00}, max$	0.01079	0.87168	1.89340
	Sigmoidal	MESE (m=12, packed)	MESE (m=12, packed)
$\Delta E_{00}, \mu$	0.01467	0.05829	0.29709
$\Delta E_{00}, max$	0.03181	0.17747	1.39676

Tab 4.1 **Perceptual error metrics.** We list mean and maximum CIE LAB ΔE_{00} for unconstrained RGB texture uplifting Figure 4.8. We compare output for our method ($m = 8, 12, 16$), low-bit variants, the *sigmoidal* [20] and the bounded *MESE* [21].



(a)

(b)

Fig 4.9 **Constrained texture recovery.** We reproduce a spectral texture dataset [76] under three standard illuminants for the *sigmoidal* [20], the bounded *MESE* [21], and our method ($m = 12$) using $n = 0, \ldots, 256$ constraints. Mean ΔE_{00} is listed for all outputs.

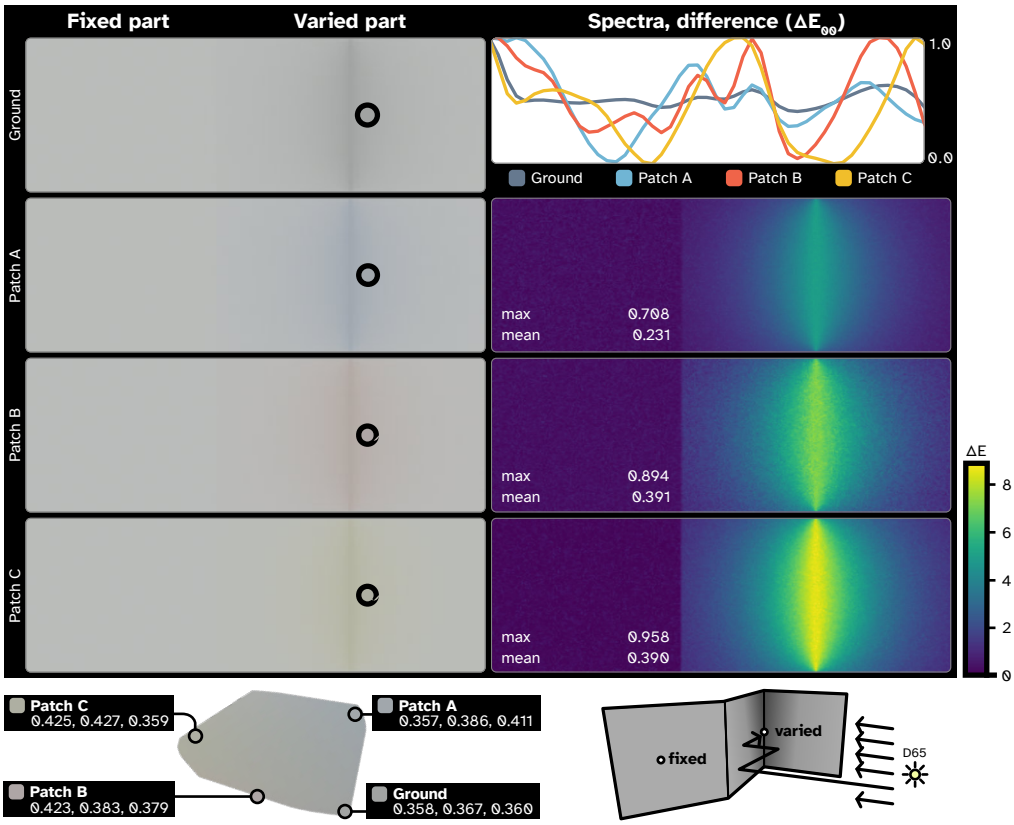


Fig 4.10 **Indirect color constraining.** *Bottom-right.* Scene setup; flat and folded patches under constant $D65$, constrained at the indicated positions. *Bottom-left.* A mismatch volume showing potential colors under interreflections in the fold. We select four color constraints. *Top.* Renders with the constrained metamers. We show ΔE_{00} for the unchanged flat patch.

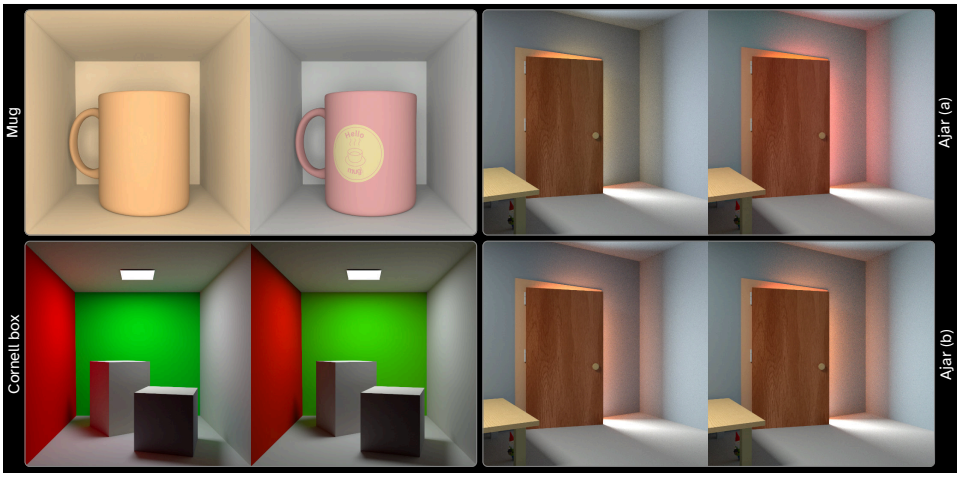


Fig 4.11 **Examples renders.** *Mug.* We enforce metamerism under $FL2$, hiding a surface texture visible under $D65$. *Cornell box.* We constrain a neutral gray surface to cancel red scattering from the left wall. *Ajar (a).* We constrain wall reflectances to obtain differing interreflections under $D65$. *Ajar (b).* We constrain the door's wood texture, instead of the wall.

4.5 Discussion

Reflectance reconstruction Our method recovers metamers with roundtrip under $D65$, outperforming or matching prior methods even in the ($m = 12$, packed) variant. While produced reflectances remain smooth, prior methods better target a low-banded output through their smaller basis or warped coefficients, or nonlinearity, though at the cost of potential mismatching under secondary illuminants. Specifically in Figure 4.9, all unconstrained methods mismatch for one or more illuminants. We hence emphasize the necessity of constrainable uplifting, as even a single interior constraint allows us to recover smooth, correctly matching spectra. If no constraints are present, one can even insert smooth spectra into the interior.

Our theory supports specular materials. However, as their view-independence makes constraint placement intuitive, implementing general BRDFs is future work.

Scene constraint specification In our implementation, users specify constraints by directly clicking in the scene. We then sample the underlying surface and construct an interior vertex in the tessellation. This enables significant user control; Figure 4.11 demonstrates example scene modifications.

However, as users edit parts of a scene, they may define conflicting constraints, e.g., two metamers sharing the same tessellation vertex and affecting the same object. This can be supported with a texture atlas; a user then guides the process via texture masks, encoding weights for each constraint which are blended during uplifting. While this implies additional operations are needed during uplifting, it is irrelevant in practice as our texture has constant evaluation cost.

Basis function encoding We employ $m = 12$ basis functions in most results, enabling metamerism. To store basis coefficients in our texture, we scale the basis, enabling low-bit representation of $[-1, 1]$ coefficients. Instead of a basis encoding a full color system, we can compute a targeted basis for just the spectra in a texture. This may allow us to improve the precision or size of our texture.

4.6 Conclusion

Our solution enables authorable spectral rendering. Tessellated uplifting improves upon prior solutions in expressivity, while remaining compact for rendering. Further, we introduced indirect illuminant constraints, forming a novel artistic tool. Our representation is efficient - our implementation reaches interactive frame rates - while opening up new avenues for spectral material design.

In this work, we targeted accurate reflectance representations. In the future, we may focus on perceptual aspects. For one, representations may be further compressed without visible difference. For another, our work shows that, despite the human visual system's limitations, we are capable of differentiating reflectances in the right scene context. Yet, during material design, this context is often lacking. We hope that novel interfaces relying on our solution may alleviate this problem.

5 Conclusion

We asserted in Chapter 1 that production rendering is increasingly physically based, in part due to a demand for virtual cloning. As such, the path forward appears to forego the tristimulus approximation. It was from this perspective that we deconstructed several challenges posed by spectral rendering, and proposed potential solutions to each.

In Chapter 2, we addressed wavelength sampling. Spectral light transport necessitates handling of wavelengths in addition to light paths. Given this, existing implementations show poor convergence behavior in scenarios with high-variance spectral radiance. We proposed a multi-pass approach. Before rendering, we built coarse estimates of camera-incident radiance. By employing these priors for importance sampling, we could favor wavelengths contributing more to the final image. Our evaluation demonstrated improved convergence on instances of non-uniform illuminant and material data. This is complementary to existing techniques. When combined with prior spectral sampling methods [12–14], we generally achieve performance not much worse than in tristimulus rendering. We discuss several remaining cases in Section 5.1.

Spectral material data is challenging to acquire and costly to render. Further, little tooling exists to handle spectral data, as production workflows generally target tristimulus rendering. For surface reflectances, this problem can be bypassed through color-to-spectrum uplifting. However, since existing methods target 1-to-1 uplifting, color metamerism was until now not considered. In Chapter 3 and Chapter 4, we proposed efficient forms of controllable spectral uplifting that lift this restriction. Our methods aimed at practical applications, enabling the use of acquired spectral data, while exposing uplifted color behavior through a scene editor. Particularly, in Chapter 4, we demonstrated control over metameric mismatching under indirect scene illumination.

Curiously, spectral uplifting lessens the disconnect between tristimulus and spectral workflows, as scene input can be entirely trichromatic independent of the system. However, as we expose mismatching behavior to the artist, the spectral workflow is more capable. This may enable the design of novel scene appearances, and we hope this can become a powerful addition to the rendering toolbox.

5.1 Future challenges

Many challenges have been addressed in the existing body of work, but several avenues of research remain.

While we demonstrated an improved wavelength sampling approach for non-uniform camera-incident radiance (Chapter 2), related issues remain. Advances in guiding and visibility sampling have seen little expansion to the spectral do-

main [35, 85]. A related problem is that of *fluorescence*. Prior work addresses efficient representations and sampling techniques [28, 86], but performance remains subpar and the selection of specific wavelengths requires further work.

Fluorescence has been addressed in the context of uplifting [51, 52]. However, to our knowledge, no tooling effectively exposes this effect to lighting or material designers. Our work on mismatching under indirect light (subsection 4.3.3) demonstrates the benefits of scene-focused tools, and could incorporate such fluorescent effects. Likewise, tooling can be adapted to control layered spectral materials; paint mixings and coatings are common examples. The uplifting of layered materials can show unexpected color behavior due to repeated scattering. Other sources of repeated scattering (participating media) also show this behavior.

Further, as uplifting of tristimulus inputs expands a single material into many potential options, novel interfaces could be developed to explore the many combinations of spectral lights and uplifted materials that are now possible. This may be a problem of data generation and exploration. Explorative interfaces have been demonstrated in the context of texture generation [87]. Given the potential complexity of the underlying domain, interactive exploration of such data may require efficient embedding algorithms (Appendix A).

Finally, perceptual aspects have been little explored in spectral rendering. While color metamerism is subtle, our work demonstrates that humans are sensitive to some metamer changes in a scene. Given the right approach, the spectral data may be further compressed. For example, we might exploit subjective effects, such as color constancy. Human vision is insensitive to minute changes in illuminant or material conditions, enabling simplification of spectral data across repeated renders in a manner similar to temporal filtering methods.

5.2 Closing words

As we stated previously, there is an inherent disconnect between tristimulus and spectral rendering. The latter methodology is rarely used in production. In this dissertation, we hope to have contributed to closing this gap, so that a single future methodology can emerge. Perhaps, just as the distinctions between notions of light and color once seemed unclear, so may this methodology appear trichromatic, but be spectral. Indistinguishable in performance, yet all the more able.

It is a good thing to have two ways of looking at a subject,
and to admit that there are two ways of looking at it.

James Clerk Maxwell

6 Acknowledgements

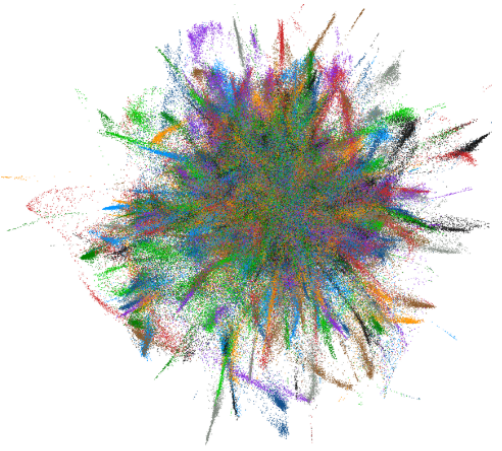
Reaching the end of this dissertation, I am uncertain where my interest in computer graphics originates. Probably my father, as each of his sons inherited a share of his interests to some degree. I remember him upgrading our GPU (a GT 6800, the latest in 18th century technology) and running a benchmark with a virtual waterfall. Laughably outdated now, but to a ten year old it was quite magical.

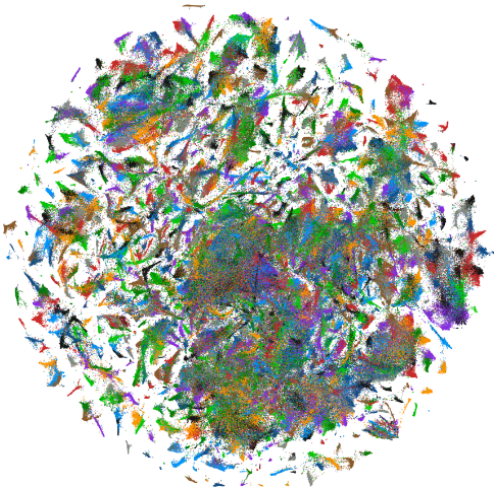
I didn't really develop my interests until Delft, when I took Elmar Eisemann's graphics course. We had to implement several ray tracing algorithms in a largely French codebase. I loved it, and would spend much of my spare time in university building renderers. In 2019 I discussed doing a PhD with Elmar. I didn't fully realize the complexity of this endeavor at the time, but he pushed me to work on problems I'd never even heard of, and in hindsight I'm nothing but grateful to him for seemingly pulling funding out of nowhere just to keep me around.

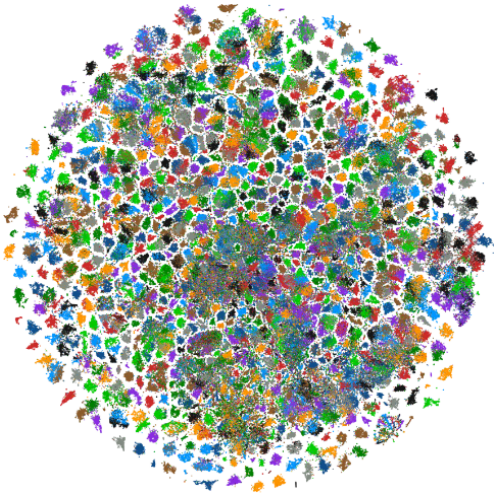
The start of my PhD coincided with the onset of the global pandemic. Strange times, but I had my research to sink my teeth into, and was surrounded by a cozy group of like-minded people. I developed an interest in the overlap between rendering and color science, and Elmar was happy to let me explore both fields. We fleshed out some ideas, and finally managed to publish those pesky SIGGRAPH papers. I wasn't always sure these things would work out (usually quite the opposite) but Elmar taught me that it takes integrity, joyfulness, a confident supervisor, and every now and then a foolhardy "See, I told you it'd work!" to do a PhD.

Of course, it also takes the people around you, like all the colleagues I've met over these five years. Whether it's hallway chair races, reading groups where no one reads the paper, or someone scheming to steal my desk (you know who you are). Many thanks to Ricardo, for helping me cross the finish line. Benno and Ali for making an office-mate laugh a lot. Amir, for being a great group organizer (and I hope he enjoyed the ttrpg). Lukas and Jackson for the peak garden gnome gag. Lauretta for helping me whenever I was bad at paperwork. Mathijs, Mika, Christoph, Bojan, Xuemei, Ruud, Fengshi, Mrinal, Priyanka, Soumyadeep, Nicolas, Chen-Chi, Celine, and Peter, for many things over the years. The staff: Rafa, Klaus, Thomas, Petr, Michael, and Martin, for all your support.

Finally, I have to recognize the people who were there along the way. My mother, who has been a helpful ear on every problem I got stuck on. My brothers and their families, who occasionally keep an eye on their brother. Dirk-Jan, Anna, Sieb, Bart, and Menno, for sometimes pulling me away from my spectra. Alex, Annemieke, Baran, and Ruben, for their many insights into the PhD process, and occasionally blowing up Oldtown. Maaike, Jaap, and Bart, for their boundless hospitality despite all my shenanigans. Berend, for just being awesome and telling me to apply to Blender as if that'd work. Jesper, Bohdan, Sarah, Shatha, and Vicente, for their boundless creativity and making me question whether crowbars exist.







A Dual-Hierarchy t-SNE as Linear-Time Minimization

Abstract

t-distributed Stochastic Neighbour Embedding (t-SNE) has become a standard for exploratory data analysis, as it is capable of revealing clusters even in complex data while requiring minimal user input. While its run-time complexity limited it to small datasets in the past, recent efforts improved upon the expensive similarity computations and the previously quadratic minimization. Nevertheless, t-SNE still has high runtime and memory costs when operating on millions of points.

In this appendix⋆, we present a novel method for executing the t-SNE minimization. While our method overall retains a linear runtime complexity, we obtain a significant performance increase in the most expensive part of the minimization. We achieve a significant improvement without a noticeable decrease in accuracy even when targeting a 3D embedding. Our method constructs a pair of spatial hierarchies over the embedding, which are simultaneously traversed to approximate many N-body interactions at once. We demonstrate an efficient GPGPU implementation and evaluate its performance against state-of-the-art methods on a variety of datasets.

⋆ This appendix is based on "*An Efficient Dual-Hierarchy t-SNE Minimization*", previously published in IEEE TVCG (IEEE VIS 2021) [88].

A.1 Introduction

The exploration of high-dimensional data has received significant interest. Non-linear dimensionality reduction techniques have made it possible to visualize structures in large-scale high-dimensional datasets, leading to discoveries in many different domains, such as immunology [89] and forensic analysis [90]. The ability to successfully preserve local structures in the data is especially important. The *t-Distributed Stotachstic Neighbour Embedding* (t-SNE) algorithm [91] achieves this goal by matching pairwise similarity distributions, representing the original data in the high-dimensional space and a possible embedding in a low-dimensional space. The algorithm consists of two phases. First, a similarity distribution is constructed over the high-dimensional data. Second, a minimization is performed using the Kullback-Leibler (KL) divergence [92] between this distribution and a low-dimensional distribution, which is initially constructed over a random embedding.

Both phases of the t-SNE computation are costly operations, becoming impractical for very large datasets. While significant effort has been invested into lowering the computational cost of the similarity computations [93–97], the minimization remains costly. Here, efforts have focused on efficiently mapping the minimization to GPU hardware [98, 99] or on reducing the $\mathcal{O}(N^2)$ runtime complexity; the commonly used Barnes-Hut t-SNE (BH-SNE) [97] initially obtained a $\mathcal{O}(N \log N)$ runtime complexity, and $\mathcal{O}(N)$ complexities were achieved afterwards by both Linderman et al. [100] and Pezotti et al. [101]. While effective for smaller 2D embeddings, millions of points remain costly and there is a significant overhead in 3D.

Our work introduces a pair of sparsely constructed spatial hierarchies to accelerate the t-SNE minimization. The first hierarchy is constructed over the embedding, and the second over a discretization of the embedding's space. We approximate N-body computations, a costly part of the t-SNE minimization, by computing interactions between the two hierarchies using a dual-hierarchy traversal. During traversal, we eliminate the majority of these interactions using an improved formulation of the BH-SNE approximation [97]. While our minimization retains a $\mathcal{O}(N)$ runtime complexity, the number of considered interactions is significantly reduced. As N-body computations previously dominated the runtime of t-SNE for two- and especially three-dimensional embeddings, our method provides a strong improvement, significantly outperforming the state-of-the-art while generating high-quality embeddings. Further, our method is designed with GPGPU programming in mind, leveraging the compute capabilities of modern GPUs.

We first formally introduce t-SNE (Section A.2) and related work (Section A.3). We then cover our method (Section A.4), its implementation details (Section A.5), and evaluation (Section A.6), before concluding (Section A.7).

A.2 t-SNE

t-SNE models a dataset of points $X = x_1, \dots, x_N$ in a high-dimensional space through pairwise similarities, represented as a symmetric joint probability distribution P. Likewise, a randomly initialized *embedding* of low-dimensional points $Y = y_1, \dots, y_N$ is represented in a similarity distribution Q. The goal of t-SNE is to minimize the difference between P and Q according to a cost function.

The distribution P, defined over the high-dimensional data points, represents the joint similarity p_{ij} between all pairs x_i and x_j. This similarity can be interpreted as the probability of these data points being near to each other in high-dimensional space. In a similar manner, the similarity between representative low-dimensional embedding points y_i and y_j is represented as q_{ij}. To minimize the difference between P and Q, the cost function C is used

Eq A.1
$$C(P,Q) = KL(P \parallel Q) = \sum_{i=1}^{N} \sum_{j \neq i}^{N} p_{ij} \ln(\frac{p_{ij}}{q_{ij}}),$$

which is the KL-Divergence between P and Q. During minimization the positions of embedding points are updated to minimize this cost. The joint similarity p_{ij} is modeled through centering of a pair of Gaussian kernels on either high-dimensional data point as

Eq A.2
$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N},$$

where

Eq A.3
$$p_{j|i} = \frac{\exp(-(\|x_i - x_j\|^2)/(2\sigma_i^2))}{\sum_{k \neq i}^{N} \exp(-(\|x_i - x_k\|^2)/(2\sigma_i^2))}$$

and variance σ_i is defined according to the local density in the high-dimensional space around x_i. As $p_{j|i}$ acts on a local neighbourhood outside of which influence diminishes rapidly, the effective number of considered points is typically much lower than N. It is instead based on a user-controlled *perplexity* value μ, and σ_i is then chosen such that

Eq A.4
$$\mu = 2^{-\sum_j^N p_{j|i} \log p_{j|i}}$$

holds for each i. For the low-dimensional similarity q_{ij}, a *Student's t-Distribution* with one degree of freedom is used instead of a Gaussian distribution. q_{ij} is defined as

Eq A.5
$$q_{ij} = \left((1 + \left\|y_i - y_j\right\|^2)Z\right)^{-1},$$

where

Eq A.6
$$Z = \sum_{k=1}^{N} \sum_{l \neq k}^{N} \left(1 + \|y_k - y_l\|^2\right)^{-1}.$$

Intuitively, to ensure that distribution Q closely represents P, their local neighbourhoods should match each other. Hence, the algorithm iteratively moves randomly-initialized embedding points around to match this criterion. This movement stems from a gradient descent applied to C. In each iteration, the gradient is computed and subsequently used to update the positions of the embedding points relying on its analytical formulation over y_i:

Eq A.7
$$\begin{aligned} \frac{\delta C}{\delta y_i} &= 4(Z \sum_{j \neq i}^{N} p_{ij} q_{ij} (y_i - y_j) - \sum_{j \neq i}^{N} q_{ij}^2 Z (y_i - y_j)) \\ &= 4(F_i^{attr} - F_i^{rep}). \end{aligned}$$

As shown, the gradient is decomposed into F^{attr} and F^{rep}, which allows for a potential reformulation as an *N-body problem*, where each of the N embedding points exerts attractive and repulsive forces on surrounding points. As is typical for N-body problems, the computational complexity is $\mathcal{O}(N^2)$.

A.3 Related Work

After the introduction of t-SNE [91], *Barnes Hut SNE* (BH-SNE) [97], reduced the runtime complexity to $\mathcal{O}(N \log N)$, and memory complexity to $\mathcal{O}(N)$. It models the similarity computation in Equation A.3 as a k-nearest-neighbour (KNN) graph problem, computed using *Vantage Point trees* [102]. In addition, a Barnes-Hut approximation [103], previously used in physics calculations, significantly reduces the number of force computations in the N-body problem.

More recent developments can be divided into two areas: improving similarity computations and improvements/replacements of the minimization algorithm. Early on, *Approximated tSNE* (A-SNE) [95], relied on principles of *Progressive Visual Analytics* [104, 105] to selectively refine parts of approximate embeddings during the optimization, while replacing a precise KNN-graph with an approximate graph relying on a forest of randomized KD-trees. A similar approach was demonstrated with *LargeViz* [96], which instead leverages randomized projection trees to obtain similarities. In addition, it links the minimization's objective function to a probabilistic graph-visualization model, which is optimized through an asynchronous stochastic gradient descent. A rather different approach is *Uniform Manifold Approximation and Projection* (UMAP) [93], which instead performs a minimization between topological representations of the high-dimensional and low-dimensional spaces. While it provides superior performance to all t-SNE variants described so far, it has been shown to suffer from many of the same down-

sides [106]. Despite the improvements, current available implementations are orders of magnitude slower than more recent GPGPU solutions.

A fast GPU-based approach is CUDA-SNE [99]. The method approximates KNN in a manner similar to A-SNE [95] with the GPU-based FAISS library [107], and maps the BH-SNE [97] optimization to a GPGPU programming environment. Although this approach achieves good performance on large datasets, it largely relates to engineering optimizations and remains bound by $\mathcal{O}(N \log N)$ runtime complexity.

More recently, linear runtime complexity was reached by *Fast Fourier transform accelerated interpolation-based t-SNE* (FIt-SNE) [100], demonstrated with a CPU-based implementation. It uses an alternative approximation for computing repulsive forces by redefining them in terms of a convolution over an equispaced grid, which is subsequently interpolated to recover repulsive forces.

A similar GPU-based approach was developed by Pezotti et al. [101], named *GPGPU linear complexity t-SNE* (L-SNE). The authors rewrite Equation A.8 as a function of *scalar* and *vector fields* — continuous functions assigning scalar or vector values to positions in space — which are then approximated in a discrete format using a GPU texture in $\mathcal{O}(FN)$ time (where F is the size of the discrete texture). Afterwards, force components are recovered through texture interpolation, which is highly efficient on GPUs. The method's runtime is dominated by the computation of this *field texture*, which suffers from scaling in either F or N and is particularly inefficient for 3D embeddings. In the following, we briefly cover this field-based formulation before presenting our approach, which avoids these shortcomings.

Given are the scalar and vector fields $\mathcal{S} : \mathbb{R}^d \to \mathbb{R}$ and $\mathcal{V} : \mathbb{R}^d \to \mathbb{R}^d$, d being the dimensionality of the embedding, typically 2 or 3. At an arbitrary position p the fields are defined as

Eq A.8
$$\mathcal{S}(p) = \sum_{i}^{N} \left(1 + \|y_i - p\|^2\right)^{-1},$$

Eq A.9
$$\mathcal{V}(p) = \sum_{i}^{N} \frac{y_i - p}{\left(1 + \|y_i - p\|^2\right)^2}$$

Based on the Student's t-distribution, $\mathcal{S}$ represents the effective density of the embedding space, while $\mathcal{V}$ represents the gradient of the repulsive forces applied. Assuming for now that these fields are available, attractive forces can be approximated in a restricted neighbourhood as

Eq A.10
$$\hat{F}_i^{attr} = \hat{Z} \sum_{\ell \in kNN(i)} p_{i\ell} q_{i\ell} (y_i - y_\ell),$$

as seen in BH-SNE [97]. The normalization factor $\hat{Z}$ is now approximated in linear time by consulting the scalar field:

Eq A.11
$$\hat{Z} = \sum_{\ell=1}^{N} (\mathcal{S}(y_\ell) - 1).$$

The repulsive force for a single point is approximated as

Eq A.12
$$\hat{F}_i^{rep} = \mathcal{V}(y_i)/\hat{Z}.$$

Computing an approximate gradient now requires linear runtime, as the fields are queried in constant time, approximated in a discrete texture format, and separately computed through a summation of the contributions of all embedding positions. Formally, positions in the fields sum kernels S and V as follows:

Eq A.13
$$\mathcal{S}(p) = \sum_i^N S(y_i - p), \quad S(t) = \left(1 + \|t\|^2\right)^{-1},$$

Eq A.14
$$\mathcal{V}(p) = \sum_i^N V(y_i - p), \quad V(t) = t\left(1 + \|t\|^2\right)^{-2}.$$

While this leads to a linear runtime, there are two observations. The kernels S and V are again based on a Student's t-distribution and have limited effects on far-away positions, but are applied to all positions with full accuracy. In addition, as the kernels have a fixed support in the embedding space, the field's discrete representation must grow with the embedding as the minimization progresses, gradually becoming larger. Pezotti et al. [101] propose that $F \ll N$ generally holds. However, while the texture grows slowly in two dimensions, the addition of a third dimension (which implies a cubic scaling of F) strongly reduces potential effectiveness. While theoretically of linear runtime, the solution is not optimal when F becomes large.

A.4 Dual-Hierarchy t-SNE Minimization

Here, we present our approach to an efficient t-SNE minimization using the field-based formulation [101]. Our approach reduces the field texture's construction time, which dominates the original runtime and renders the solution impractical for higher embedding dimensionalities. We observe that this discrete representation in form of a texture requires evaluating many small regions with varying local interactions, but similar global interactions. We propose to represent both the embedding and the discrete field as spatial hierarchies, henceforth referred to as the *embedding hierarchy* and the *field hierarchy* respectively. We perform a dual traversal over these hierarchies, during which we employ an improvement of the approximation criterion used in BH-SNE [97] to selectively compute interactions between hierarchy nodes, which represent large regions in the embedding and the field (Figure A.1). These interactions between the regions are not directly transferred to data points but are first stored in the hierarchy itself; specifically, for a region, the interaction is added to its corresponding node of the hierarchy. Hereby, we benefit from both hierarchies. After dual traversal, we accumulate these interactions that are stored throughout the hierarchy to form a complete, yet sparsely-

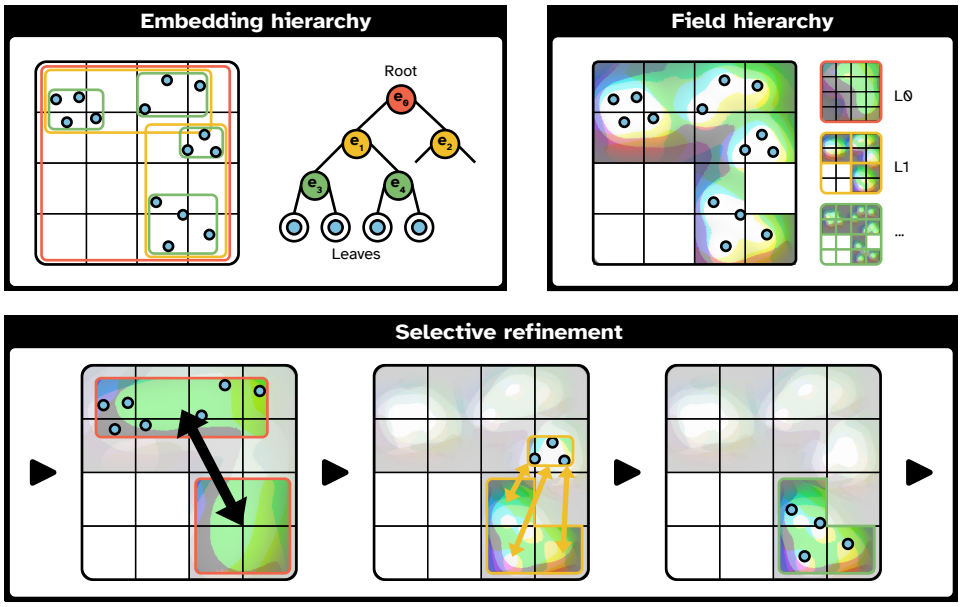


Fig A.1 We use embedding and field hierarchies (top), comparing their nodes to compute interactions between many points and large portions of the field in a single step (bottom left). Where refinement is necessary, we descend one or both hierarchies (bottom mid), continuing until points interact with a full-resolution field (bottom right).

computed approximated field. In this way, we improve upon the original $\mathcal{O}(FN)$ complexity of the field computation, as our cost approaches $\mathcal{O}(N)$. We provide a proof in the supplementary material, but suggest to first follow the algorithm in this section to ease understanding. Figure A.2 shows an overview of our method, divided into three steps: *hierarchy construction*, *dual traversal*, and *field accumulation*. We detail each step in the following.

A.4.1 Hierarchy Construction

We construct hierarchies over the embedding and field (Figure A.2, first part). Meyer et al. [108] showed that a careful choice of the spatial hierarchy provides performance improvements to BH-SNE [97]. We chose our structures with efficient execution on the GPU in mind.

As embedding hierarchy, we select an implicit linear *bounding volume hierarchy* (BVH), constructed in linear time on the GPU (Section A.5). In a BVH, each node stores an *axis-aligned bounding box* (AABB) encompassing the child-node bounding boxes, while leaf nodes directly contain one or more objects (i.e., embedding points). BVHs provide a close fit around contained data, and allow for refitting of AABBs without fully rebuilding the hierarchy. The latter is an important cost-saving measure, made possible because embedding positions move slowly during the minimization. As with BH-SNE [97], nodes in the embedding hierarchy track

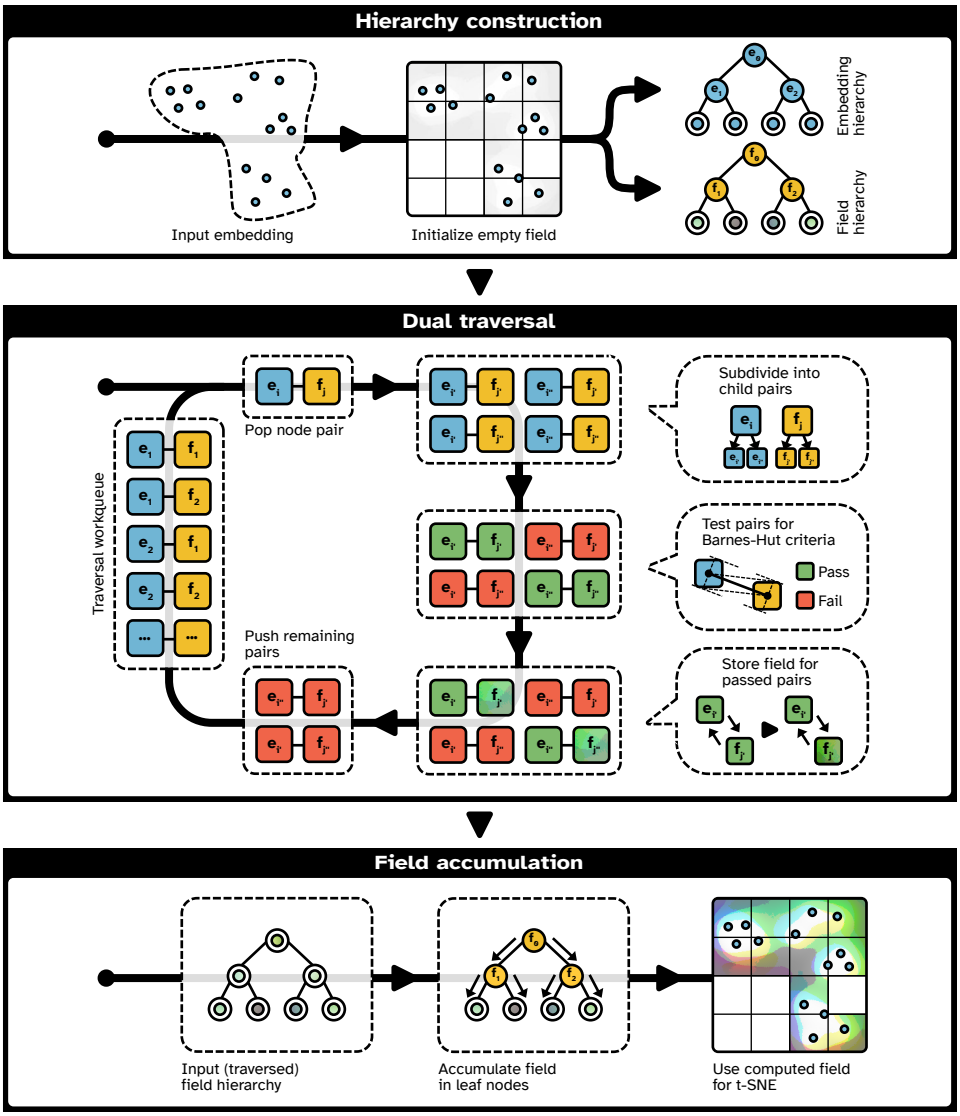


Fig A.2 We generate embedding and field hierarchies (a), and dual traverse these using a work queue (b). When approximation of the interactions between node pairs suffices, we cull these pairs. Further, we evaluate and store interactions between pairs at different levels of detail in the hierarchies. A final traversal (c) constructs the field used in the t-SNE minimization.

their center of mass, defined as the average of the contained embedding points. The center of mass c_i of a node e_i with mass m_i is simply

Eq A.15
$$c_i = \frac{1}{m_i} \sum_{j \in emb(e_i)} y_j,$$

where $emb(e_i)$ defines the indices of the points in the node.

Observing the discrete grid nature of the field texture, we select a sparse implicit quad-/octree for the field hierarchy. We mark cells of interest in the grid that we build our hierarchy upon. This is done in $\mathcal{O}(F \log N)$ time, but marking costs are negligible in practice ($< 1\%$ of total compute time). For each grid cell, we descend the previously generated embedding hierarchy to determine if the cell overlaps or borders embedding points. We then construct the sparse field hierarchy with the marked cells as leaf nodes. Computing the field for these locations suffices, as it will only be accessed here during the minimization. Each node f_j in the hierarchy has scalar and vector field entries $\hat{\mathcal{S}}_j$ and $\hat{\mathcal{V}}_j$, which are initialized as 0 at the start of every iteration of the minimization and used as intermediate storage during traversal. Contrary to the embedding hierarchy, nodes in the field hierarchy represent regions and their center of mass c_j is simply their region's geometric center.

A.4.2 Dual Traversal

With both hierarchies available, we perform a dual traversal (Figure A.2, second part), formulated as a top-down breadth-first traversal of a single, larger tree. This tree consists of nodes representing node pairs (e_i, f_j), where e_i and f_j are respectively nodes in the embedding and field hierarchies. Each node pair represents a potential interaction between the embedding points and field regions described by the two contained nodes. We model traversal using a work queue, in which we store node pairs in the dual hierarchy that still have to be traversed. At the start of traversal, a root node pair, i.e., (e_0, f_0), is pushed on the queue. During traversal, a node pair is popped from the queue, and is subsequently subdivided. We descend one level in both hierarchies under each node if possible. If both nodes are leaves, we compute the underlying interactions directly. Otherwise, the different possible pairs of child nodes from both hierarchies are tested via an approximation criterion (subsection A.4.3), to determine if they represent interactions with a sufficient accuracy. If this criterion fails for a child node pair, it is pushed on the work queue for further subdivision. If it holds, we will not further descend into the dual hierarchy underneath this child node pair but process them directly.

To process a node pair, we compute the interactions by using an approximation of the kernels in Equation A.13 and Equation A.14:

Eq A.16
$$\hat{\mathcal{S}}(e_i, f_j) = m_i \; S(c_i - c_j),$$

Eq A.17
$$\hat{\mathcal{V}}(e_i, f_j) = m_i \; V(c_i - c_j).$$

Both values are computed once and will be used for all m_i points in the embedding node and all regions under the field node instead of evaluating m_i values for potentially many field cells. These values are atomically added to $\hat{\mathcal{S}}_j$ and $\hat{\mathcal{V}}_j$ in the

field node f_j.

Traversal is finished once the work queue is empty. We purposefully subdivide node pairs before testing an approximation criterion — as opposed to the inverse — for implementation reasons (Section A.5).

A.4.3 Barnes-Hut Approximation

We modify the dual-hierarchy approximation criterion of BH-SNE [97] to determine whether a node pair can be processed. Originally, given lengths r_i, r_j of the diagonals of two nodes' AABBs, and node centers c_i, c_j, the following term is evaluated:

Eq A.18
$$\frac{max(r_i, r_j)}{\|c_i - c_j\|} < \theta.$$

The parameter θ defines a maximum allowed ratio, interpreted as the tangent of an angle in a triangle whose opposite and adjacent edges have lengths $max(r_i, r_j)$ and $\|c_i - c_j\|$ respectively. A larger θ means larger bounding boxes closer to each other pass the test, leading to earlier processing in the hierarchy (faster traversal) but a coarser approximation. Similarly, if $\theta = 0$, the hierarchies are traversed fully, leading to an inefficient but accurate computation. For single-hierarchy traversals, θ typically lies between 0.1 and 0.5 [97]. The condition is simple as it is evaluated many times, assuming that all bounding boxes in both hierarchies have regular sides (as is the case for quad-/octrees).

Hierarchies such as a linear BVH tend to produce irregular bounding boxes that closely fit the contained data. Here, the Barnes-Hut criterion is suboptimal, as it considers a bounding box based on its diagonal, which is not a good representative of all sides when having a highly irregular bounding box. Hence, we modify Equation A.18 to project the diagonals d_i, d_j of the nodes' bounding boxes so the approximation criterion accurately matches this irregularity, leading to projected diagonal lengths r_i', r_j'. We visualize our approach in Figure A.3.

To obtain projected diagonal lengths, we first compute a unit vector $\hat{t}$ along the difference $c_i - c_j$, but reflected across axes so it is near-orthogonal to the diagonals. In two dimensions, this is simply:

Eq A.19
$$\hat{t} = \left| \frac{c_i - c_j}{||c_i - c_j||} \right| \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

A suitable length is then obtained through vector rejection as:

Eq A.20
$$r_i' = \|d_i - \hat{t}\ (d_i \cdot \hat{t})\|.$$

We compute r_j' in the same manner and use $\max(r_i', r_j')$ for the comparison in Equation A.18. We evaluate this criterion in Section A.6.

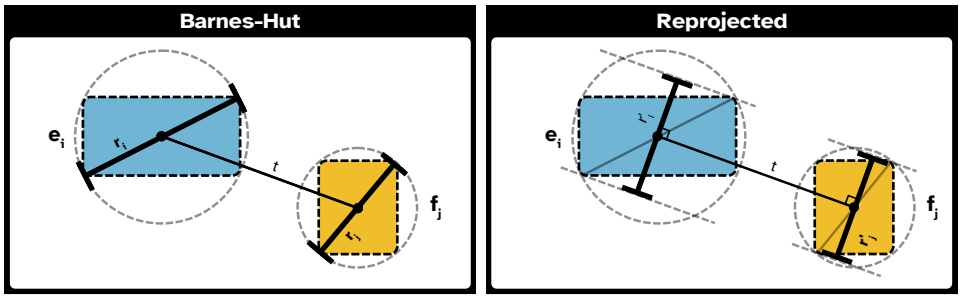


Fig A.3 Simple modification to Barnes-Hut approximation [103]. Instead of constant radii based on bounding box diagonals, we reproject diagonals, handling irregular bounding boxes regardless of their respective positions to each other.

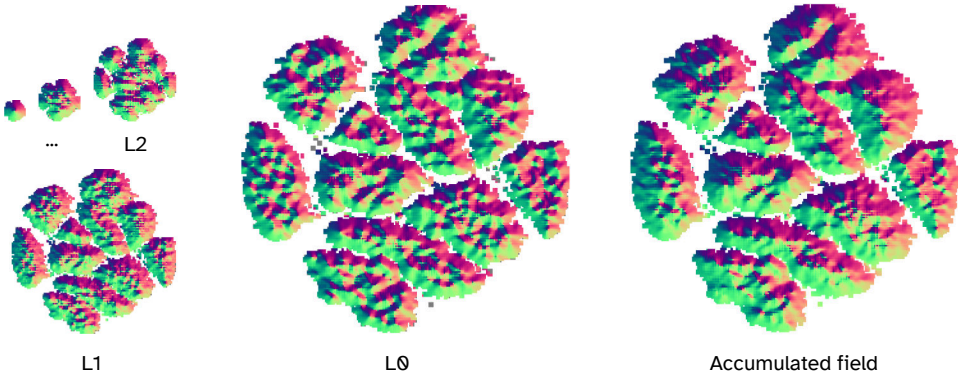


Fig A.4 The field hierarchy is ascended from leaf nodes (L0) and intermediate values in consecutive smaller levels (L1, L2, ...) are added, resulting in an approximate field. A sparse vector field is visualized, red and green colors marking x- and y-directions of the vectors.

A.4.4 Field accumulation

After dual traversal, we collect the approximate interactions stored in the hierarchies (Figure A.2, third part). In particular, field hierarchy nodes now store intermediate parts of the actual fields in $\hat{\mathscr{S}}_j$ and $\hat{\mathscr{V}}_j$. As in [101], we want to interpolate the discrete field to obtain approximate field values at embedding positions, which is difficult in a hierarchy. Hence, we flatten it to recover a coarsely approximated texture, i.e., we ascend the field hierarchy upwards once for each non-empty leaf node, accumulating encountered field scalar and vector values and storing their sum in the respective texture position of said leaf node. This requires $\mathcal{O}(F \log F)$ time when gathering upwards from a leaf to the root. Performing the operation in reverse would lead to $\mathcal{O}(F)$ time but is less practical on GPU hardware. Afterwards, the field can be queried for interpolated scalar and vector field values per point. Figure A.4 displays an accumulation of different levels of the field hierarchy.

A.5 Implementation

Our technique is implemented following a GPGPU approach. We develop our implementation in a combination of the OpenGL 4.6 API and CUDA 11, although the concepts we described can be applied on other APIs. Our implementation is available online*.

Mirroring the algorithmic description, our t-SNE implementation consists of two parts. We first generate the joint similarity distribution P in the same manner as Chan et al. [99], using approximate KNN information with $k = 3\mu$ obtained through the GPU-based FAISS library [107]. The formulation of the distributions remains the same as BH-SNE [97]. Second, we mirror the matrix-based minimization used by Pezotti et al. [101]. During the minimization, we invest time at the start of each iteration to rebuild or refit our spatial hierarchies, and then perform a dual-hierarchy traversal, replacing the expensive field computation.

A.5.1 Hierarchy Construction

As mentioned, we implement the embedding hierarchy as an implicit linear BVH, constructed on the GPU in $\mathcal{O}(N)$ time. We outline the general method, but refer the reader to Lauterbach et al. [109] for a full description. In short, the linear BVH method reduces BVH construction to a single sorting operation. Each of the N embedding points is assigned a Morton code based on their discretized position in 2D/3D space. Based on these codes, the points are bucketed in leaf nodes, which are subsequently arranged along a space-filling z-order or Morton curve in a $\mathcal{O}(N)$ parallel radix sort, using the Morton codes as keys. After sorting, levels of the hierarchy are constructed iteratively by grouping nodes, which share the same high order bits in their respective Morton codes. Our implementation adopts the work-queue based approach of Garanzha et al. [110]. Faster and more recent construction algorithms can be used at the cost of increased code complexity. For a parallel radix sort, we leverage the implementation available in the CUDA-based CUB library [111], which can access specific buffer objects in OpenGL through the included interoperability library.

We implement the field hierarchy as a sparse implicit quad-/octree due to the discrete nature of the field texture. As nodes in this hierarchy are regular, we do not store bounding box information, instead deriving these from node indices when necessary. The only information we store in a node is its type and the mentioned intermediate scalar and vector values used during traversal.

Although the embedding changes rapidly during early iterations, changes are less pronounced later on. Early iterations of t-SNE, typically the first 250, use *early exaggeration*, multiplying p_{ij} by some scalar to aggressively separate clusters. We use this to our advantage to reduce hierarchy-construction costs significantly. While we rebuild hierarchies on every iteration during early exaggeration, we only do so

* https://www.github.com/markvanderuit/dual_hierarchy_tsne

at regular intervals after. We can often simply refit bounding boxes around the newly updated positions, avoiding costly sorting. As the number of leaves in the field hierarchy can change at each iteration — leading to a substantially different spatial hierarchy — we included the cells bordering embedded points as leaves, which enables a reuse.

A.5.2 Dual Traversal

As described in Section A.4, dual-hierarchy traversal is formulated as a breadth-first traversal, in which node pairs are read, subdivided, and tested for further traversal. We leverage a pair of work-queues to track traversal. At the start of traversal, an initial set of node pairs (matching to root nodes) is written to the first, or primary work-queue. During a single step of traversal, all node pairs on the primary work-queue are subdivided and tested for Equation A.18. Node pairs which fail the approximation criterion are pushed on the secondary work-queue, which is subsequently swapped with the primary work-queue for the next traversal step. We repeat this process until the primary work-queue is empty or the leaf levels are reached, at which point traversal has completed.

As root nodes encompass the entire embedding, they will always be subdivided. As an optimization, we start traversal at a lower level in both hierarchies by pushing all pairs corresponding to the selected levels on the work-queue (we use levels 3/2 for a 2D/3D embedding, leading to 4096 node pairs for hierarchies with fan-outs 4/8).

To optimize subdivision, we leverage local cross-communication capabilities of modern GPUs (subgroups in OpenGL/GLSL, warps in CUDA) to test multiple combinations of node pairs per GPU thread (invocation) while minimizing memory operations. To subdivide a single node pair on both sides, we use two threads (four for quadtrees, eight for octrees), having each thread load a single child node from both sides of the hierarchies. The total number of node pairs that must be tested (four for binary trees, 16 for quadtrees, 64 for octrees) can be obtained by rotating the child nodes on one side of the hierarchy along the 2/4/8 threads, using the subgroup capabilities.

A.5.3 Single-Hierarchy Fallback

As described in Section A.3, the discrete field grows in size as the minimization progresses. At its start, the small discrete field implies few regions of interest require computation, leading to a sparse hierarchy. In this scenario, a dual traversal is inefficient as the field hierarchy's levels have too few nodes to fully occupy the GPU. We establish a maximum positive difference in depths d_e, d_f between the embedding and field hierarchies (i.e., $d_e - d_f \leq d_{max}$) to determine when dual hierarchy traversal is used. We empirically established $d_{max} = 4$ as a suitable threshold. Whenever we forego a dual-hierarchy traversal, we only construct the embedding hierarchy, and depth-first traverse it for the entire field in $\mathcal{O}(F \log N)$ time.

Ours			L-SNE [101]		
Component	Time complexity	Space complexity	Component	Time complexity	Space complexity
Hierarchy	$\mathcal{O}(F+N+F\log N)$	$\mathcal{O}(F+N)$			
Dual traversal	$\mathcal{O}(max(F,N))$	$\mathcal{O}(F+N)$			
Field comp. **	$\mathcal{O}(F\log F)$	$\mathcal{O}(F)$	Field comp.	$\mathcal{O}(FN)$	$\mathcal{O}(F+N)$
F^{rep} lookup *	$\mathcal{O}(N)$	$\mathcal{O}(F+N)$	F^{rep} lookup *	$\mathcal{O}(N)$	$\mathcal{O}(F+N)$
F^{attr} comp.	$\mathcal{O}(KN)$	$\mathcal{O}(KN)$	F^{attr} comp.	$\mathcal{O}(KN)$	$\mathcal{O}(KN)$
Apply forces *	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Apply forces *	$\mathcal{O}(N)$	$\mathcal{O}(N)$
FIt-SNE [100]			BH-SNE [97]		
Component	Time complexity	Space complexity	Component	Time complexity	Space complexity
Point-grid	$\mathcal{O}(pN)$	$\mathcal{O}(pN_{int}+N)$			
Grid-grid	$\mathcal{O}(pN_{int}\log pN_{int})$	$\mathcal{O}(pN_{int})$	Hierarchy	$\mathcal{O}(N)$	$\mathcal{O}(N)$
F^{rep} comp.	$\mathcal{O}(pN)$	$\mathcal{O}(pN_{int}+N)$	F^{rep} comp.	$\mathcal{O}(N\log N)$	$\mathcal{O}(N)$
F^{attr} comp.	$\mathcal{O}(KN)$	$\mathcal{O}(KN)$	F^{attr} comp.	$\mathcal{O}(KN)$	$\mathcal{O}(KN)$
Apply forces *	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Apply forces *	$\mathcal{O}(N)$	$\mathcal{O}(N)$

* Component has negligible computational runtime (Figure A.9).
** Component can be performed in $\mathcal{O}(F)$ time (subsection A.4.4).

Tab A.1 Time/space complexities for the stages of our and other methods. N is input size, K is restricted neighbourhood size, and F is field size for our method and L-SNE [101]. N_{int} and p are parameters of FIt-SNE [100]: N_{int} represents a discrete grid size, and p represents a number of equispaced points over said grid. Note that F, N_{int} and p are independent of N.

Dataset	Points	Dims.	Iters.	μ
MNIST	60000	784	1000	50
Fashion	60000	784	1000	50
ImageNet	1250000	128	4000	10
Word2Vec	3000000	300	4000	5

Tab A.2 Sizes, dimensions and selected minimization parameters for each dataset.

A.6 Evaluation

We first evaluate specific choices of our method, and afterwards compare against state-of-the-art solutions in terms of computational cost and embedding quality. All experiments run on a particular dataset use the same configuration and parameters. Further, all experiments are conducted on a single machine with an Intel Core i7-9900 (16 logical threads @3.1 GHz), 16 GB of DDR4 RAM and a GeForce RTX 2080 Ti GPU with access to 11 GB VRAM. For each experiment, we record the minimization runtime and resulting KL-divergence as a direct measure of how far a specific minimization has progressed. As KL-divergence is directly coupled to minimization, we additionally consider an unrelated metric, selecting Nearest-Neighbourhood-Preservation (NNP) as described by Venna et al. [112]. It measures how well local neighbourhoods in the low-dimensional embedding preserve characteristics of their high-dimensional counterparts. In order to obtain correct results for NNP, the gradient descent must be (mostly) converged. Hence, we use a larger number of iterations for larger datasets.

Dataset	CUDA-SNE	L-SNE	Ours
MNIST	5.86s	1.70s	1.36s
Fashion	5.74s	1.83s	1.40s
ImageNet	94.12s	346.94s	51.10s
Word2Vec	94.90s	212.11s	86.90s

Tab A.3 Compared runtimes of methods for a 2D minimization, using the datasets in Table A.2.

For testing, we select four datasets that are frequently applied in the evaluation of dimensionality reduction algorithms such as t-SNE. As different datasets typically differ in size, dimensionality, and structure, we select separate iterations and perplexity μ for each dataset. Specific sizes and parameters selected for each dataset are displayed in Table A.2.

The commonly-used MNIST dataset consists of labeled 28×28px grayscale images of handwritten digits, each represented as a vector storing an image's pixel values. MNIST is often used for this kind of evaluation as it contains 10 clearly-defined classes corresponding to 10 different digits. The similar Fashion-MNIST [113] contains images of 10 different types of clothing, instead of digits, which are sometimes closely related but harder to separate into clusters with an algorithm such as t-SNE. For this reason, we included it in our evaluation.

The ImageNet dataset [114] stores approximately 1000 categories of random images of objects at varying resolutions. We use a reduced and formatted version previously published by Fu et al. [115], processed such that each vector in the dataset has a dimensionality of 128.

The GoogleNews dataset stores a collection of three million words, each represented as a vector generated by Word2Vec [116]. This tool consumes a text corpus — in this case originating from Google News — and assigns words in the corpus a representative vector in such a way that words are closely related if they share a similar context.

A.6.1 Hierarchy Evaluation

We first evaluate our choice of spatial hierarchy. Although our method works with different hierarchies, we focus on the implicit linear BVH [109]. Use of alternatives such as a quad-/octree is possible. However, BVHs have several benefits: they fit the contained data closely, and their bounding volumes can be refitted when data changes. Refitting instead of rebuilding provides a significant reduction in runtime over consective iterations. We compare minimizations of MNIST and ImageNet in four cases: using quad-/octrees, using a BVH rebuilt every iteration, and using BVHs that are rebuilt after four or eight iterations of refitting. No refitting is performed in the first 250 iterations as early exaggeration takes place. Results are displayed in Figure A.5. The quad-/octree has to be rebuilt every iteration. It only matches the BVH performance when the latter is always rebuilt. The benefit of the BVH becomes apparent when refitting is used, e.g., during four iterations. However, this degrades

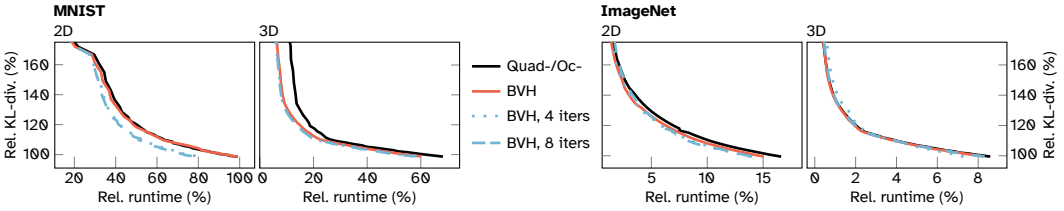


Fig A.5 Comparison of a BVH with/without refitting and a quad-/octree as a spatial hierarchy over 2D/3D embeddings of the MNIST and ImageNet datasets. We minimize for increasing numbers of iterations and plot the resulting relative runtime and KL-divergence. Baseline results are established with GPGPU linear complexity t-SNE [1Ø1].

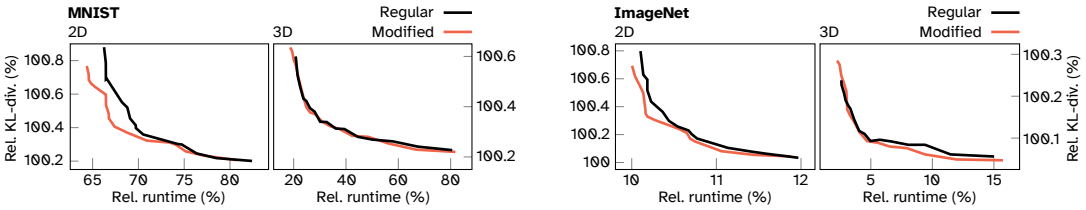


Fig A.6 Comparison of our modified Barnes-Hut approximation and the regular form for generation of 2D/3D embeddings over the MNIST and Imagenet datasets. We minimize for differing $\theta \in [0.2, 0.6]$ and plot the resulting relative runtime and KL-divergence. Baseline results are established with GPGPU linear complexity t-SNE [1Ø1].

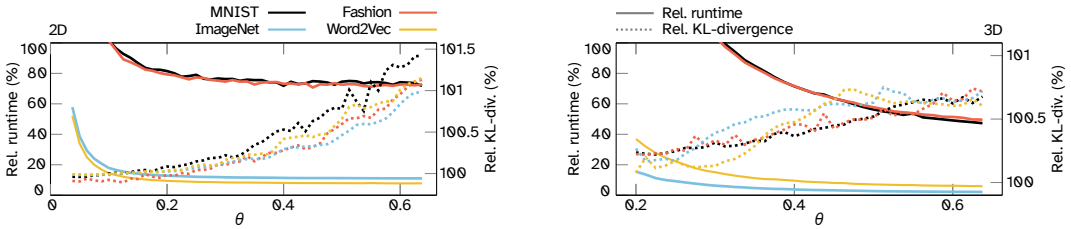


Fig A.7 Evaluation of θ for generation of 2D/3D embeddings. We show the resulting relative runtime and KL-divergence of three datasets. Baseline results ($\theta = 0$) are 100% and are established with GPGPU linear complexity t-SNE [1Ø1]. In larger datasets and small $\theta < 0.2$, 3D minimizations may exceed the memory capacity of our GPU and are not shown.

BVH quality, and refitting for too many iterations results in unpredictable runtimes. This is seen in the ImageNet minimization for 8 iterations of refitting. Also, as the embedding still undergoes significant changes after the early exaggeration phase, refitting degrades the hierarchy quality. We show iteration runtimes in Figure A.1Ø, where these effects are visible. In practice, we employ four consecutive iterations in all other examples.

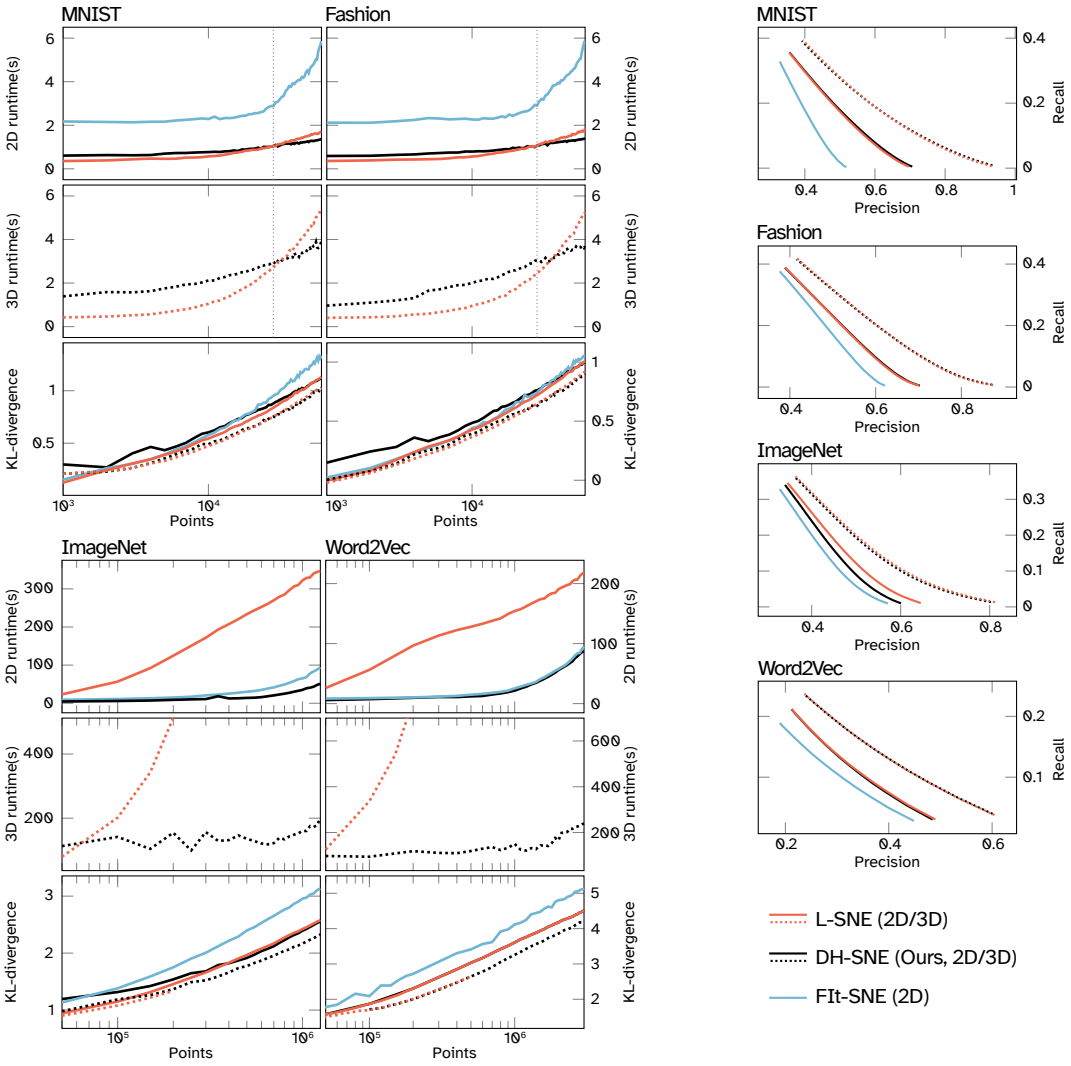


Fig A.8 Comparison of linear complexity t-SNE [101], a CUDA implementation of FIt-SNE [99, 100], and our method (DH-SNE) across four datasets. We show minimization runtimes and of KL-divergence for 2D/3D embeddings over increasing subsets of data (left, horizontal axes are logarithmic). We next show NNP in the form of precision/recall curves (right). Our method outperforms the state-of-the-art on large datasets in terms of runtime for both 2D/3D, while retaining a similar quality to linear tSNE [101].

A.6.2 Barnes-Hut Evaluation

Next, we evaluate our modified Barnes-Hut approximation criterion in conjunction with a BVH. This criterion handles irregular bounding volumes, which occur in a BVH, better than the original. We compare minimizations of MNIST and ImageNet with our criterion and the regular criterion [103]. We test for differing $\theta \in [0.2, 0.6]$

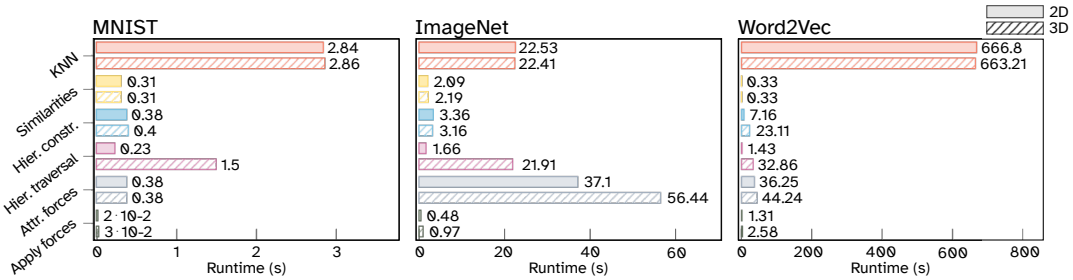


Fig A.9 Comparison of the runtimes of the most expensive components of our method on three datasets of varying sizes, for 2D/3D embeddings. The different perplexity values listed in Table A.2 imply different local neighbourhood sizes, leading to varying attractive force computation costs (Equation A.10). As demonstrated, runtime is dominated by KNN and attractive force computations on larger datasets.

and record resulting (relative) runtime and KL-divergence. Results are displayed in Figure A.6. While the improvements are stronger for larger values of θ, the new criterion outperforms the regular criterion in all cases.

Larger θ leads to a coarser approximation and faster traversal as nodes are culled earlier. While this parameter was evaluated in BH-SNE [97] in the context of single-hierarchy traversal, the established $\theta \leq 0.5$ does not hold for our method. In addition, the parameter's impact on traversal may vary across 2D/3D embeddings. We investigate its effect in both scenarios in Figure A.7. We consider $\theta = 0.25$ a good tradeoff for 2D, and $\theta = 0.4$ for 3D. There is a noticeable increase in KL-divergence for larger θ across datasets, which becomes visible as grid-like patterns.

A.6.3 Comparative Evaluation

Finally, we compare with state-of-the-art techniques with linear runtime complexity. First, we select the field-based L-SNE developed by Pezotti et al [101], with which we generate both 2D/3D comparisons. This technique shows excellent performance on smaller datasets and provides high quality embeddings in comparison with earlier techniques. We use field scalings of 2.0 (2D) and 1.2 (3D) for measurement with both our method and L-SNE. We also select a current version of CUDA-SNE [99] which, instead of a Barnes-Hut approximation, recently adapted the $\mathcal{O}(N)$ FIt-SNE [100] to the GPU. Although their approach incurs an overhead for smaller datasets, it outperforms the original implementation due to a linear runtime. This implementation only generates 2D embeddings, so we only compare it in this regard. Older BH-SNE [97] or baseline $\mathcal{O}(N^2)$ t-SNE algorithms [91] have been omitted, as their practical performance is typically orders of magnitude slower.

As our technique uses a field similar to Pezotti et al. [101], we expect to produce similar embeddings at improved runtime performance. With regards to FIt-SNE [100], we expect to reach similar or improved performance on large datasets, while producing substantially different embeddings, as our minimization differs

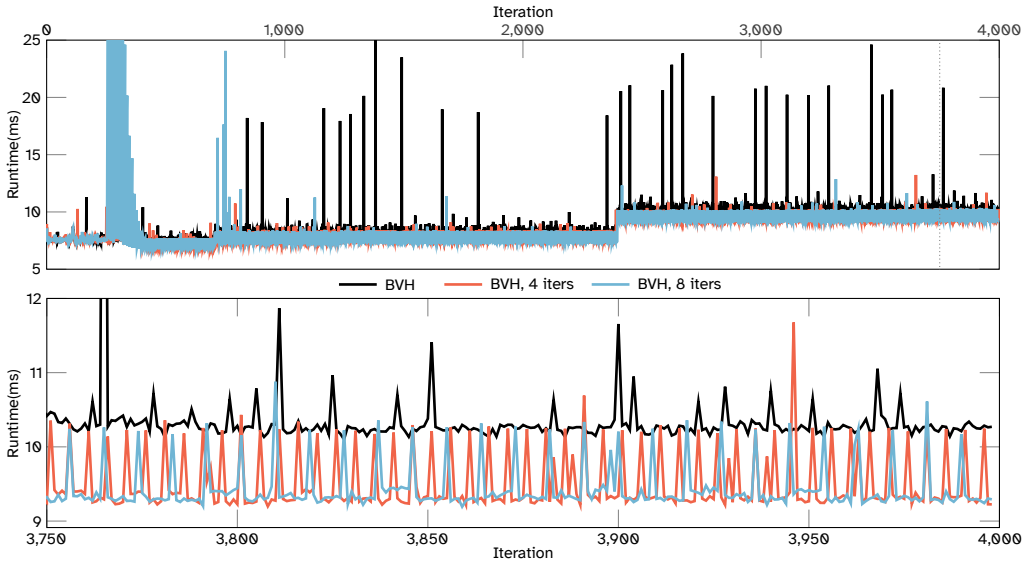


Fig A.10 Influence of refitting a BVH during minimization of the ImageNet dataset. We show runtime per iteration for all 4000 iterations (top) and the last 250 iterations (bottom). Note the spike in runtime after early exaggeration for 8 iterations of refitting.

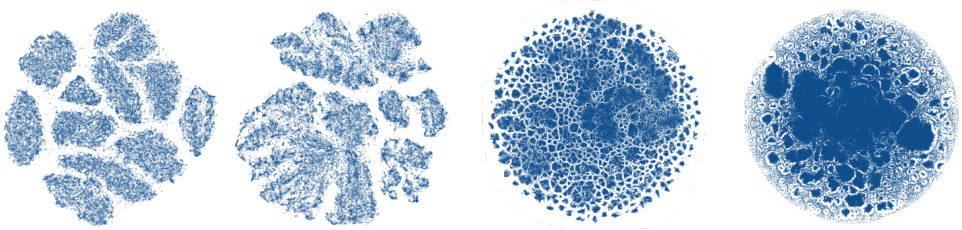


Fig A.11 Embeddings of datasets (Table A.2), generated by our method.

from theirs by definition. To correctly compare the different minimization methods, we ensure they use identical KNN information and an identical joint similarity distribution P. We further ensure all methods use an identical initial embedding, and use identical parameters for their gradient descent. Differences between methods then correspond solely to the differences in their respective complexities. We provide an overview of the different time/space complexities of each method in Table A.1.

The first two rows of Figure A.8 show minimization runtimes for 2D/3D embeddings separately. We run on increasingly large subsets of the datasets to show how minimization progresses. A logarithmic scale is used on horizontal axes to account for large dataset sizes. We list exact runtimes of minimizations on the full dataset in Table A.3.

Our technique performs exceedingly well for sufficiently large datasets; starting at approximately 27K points (indicated by a dotted vertical line) it outperforms compared methods in both the relatively small MNIST and Fashion datasets. On the full datasets, a 1000 iteration minimization requires $1.36s$, compared to $1.70s$ for L-SNE [101]. This gap widens significantly in the 1.2M point ImageNet dataset, where our method completes a 4000 iteration minimization in $51.10s$, down from $346.94s$. On the 3M point Word2Vec dataset, FIt-SNE [100] performs a 4000 iteration minimization in $94.90s$, while our method requires $86.90s$. Convergence between the methods on the Word2Vec dataset is explained by attractive-force computations (Equation A.10), which become exceedingly expensive for denser local neighbourhoods. For comparison: both existing methods perform relatively poorly on the smaller ImageNet dataset, where a higher perplexity value leads to larger neighbourhoods. To confirm this, we display runtimes of separate components in our method in Figure A.9. Evidently, attractive-force computation becomes a dominating factor in the minimization.

Observed scaling for 3D embeddings remains linear in all experiments, though there is a runtime overhead compared to 2D embeddings. This is expected, given the computational overhead involved in a third dimension. Linear complexity tSNE [101] is impractical for large datasets, as runtime spikes around 100K points, while our technique is orders of magnitude faster and completes a full 4000 iteration minimization on the 3M point Word2Vec dataset in $238.67s$.

While our technique improves runtime, embedding quality is another important metric. In the last two rows of Figure A.8, we examine KL-divergence of generated embeddings for increasingly large subsets of the datasets, in addition to computed NNP. The NNP metric is displayed in the form of precision/recall plots. For this, we repeat an experiment performed by Pezotti et al. [101]. For each point in a dataset, we observe points in an increasingly large neighbourhood of a size k based on perplexity. For every value from $k = 1$ to $k = 3\mu$, we compute T, defined as the accurate number of points belonging to both points' neighbourhoods. Precision is computed as T/k and recall is computed as $T/(3\mu)$. By averaging generated curves for each point, a representative curve is obtained for the entire dataset.

As demonstrated, our approximation has a minor impact on embedding quality compared to linear complexity tSNE [101]. The CUDA-SNE [99] implementation of FIt-SNE [100] interestingly delivers a lower quality of embeddings for the specified metrics. As explored by Linderman et al. [100], FIt-SNE reaches comparable levels of quality to BH-SNE [97], so these results are expected. The field-based approximation used by Pezotti et al. [101] is established to be more accurate, which is a quality our method mostly retains. We display embeddings generated with our method in Figure A.11. Further, we show example minimizations in a supplemental video.

A.7 Conclusion

We have presented a novel and improved minimization algorithm for t-SNE, providing significant performance improvements above the state-of-the-art, especially for large datasets and higher dimensional embeddings. The latter point is a crucial step forward, as it can improve embedding quality and could be of high relevance in many applications relying on a 3D visualization. For this reason, we have made an implementation of our method available on Github *.

Our method illustrates that a field-based formulation of t-SNE, previously shown to have linear runtime, can still be significantly accelerated via a dual-hierarchy traversal. This allows us to compute N-body interactions efficiently, as is demonstrated in a GPGPU-based environment on modern graphics hardware. Our experiments reveal significant run-time improvements with regards to linear complexity t-SNE [101] and FIt-SNE [99] for two- and three-dimensional embeddings, while achieving comparable quality.

With these improvements, the t-SNE algorithm is, at this stage, dominated by the required KNN and attractive force computations, which are interesting challenges for future work.

Curriculum Vitæ

Mark van de Ruit

10-09-1993	Born in Bergen op Zoom, the Netherlands.

Education

2006–2012	Develstein College, Zwijndrecht *VWO Atheneum*
2013	Breda University of Applied Sciences Propedeuse Game Design & Architecture
2014-2025	Technische Universiteit Delft BSc, Computer Science (’14-’17) MSc, Computer Science (’17-’19) PhD, Computer Graphics (’20-’25)

List of Publications

4. M. B. Mark van de Ruit and E. Eisemann. “An Efficient Dual-Hierarchy t-SNE Minimization”. In: *IEEE Transactions on Visualization and Computer Graphics* (2021). DOI: 10.1109/TVCG.2021.3114817

3. M. van de Ruit and E. Eisemann. “A Multi-Pass Method for Accelerated Spectral Sampling”. In: *Computer Graphics Forum* (2021). DOI: 10.1111/cgf.14408

2. M. van de Ruit and E. Eisemann. “Metameric: Spectral Uplifting via Controllable Color Constraints”. In: *SIGGRAPH 2023 Conference Proceedings*. 2023. DOI: 10.1145/3588432.3591565

1. M. van de Ruit and E. Eisemann. “Controlled Spectral Uplifting for Indirect-Light-Metamerism”. In: *SIGGRAPH Asia 2024 Conference Papers*. 2024. DOI: 10.1145/3680528.3687698

Bibliography

[1] J. C. Maxwell. *The Scientific Papers of James Clerk Maxwell*. Cambridge University Press, 1890.

[2] T. Young. “The Bakerian Lecture: on the theory of light and colours”. In: *Philosophical Transactions of the Royal Society of London* (1802). DOI: 10.1098/rstl.1802.0004.

[3] T. Young. *Chromatics*. Supplement to the Encyclopaedia Britannica. 1817.

[4] I. Newton. *Opticks, or, a treatise of the reflections, refractions, inflections & colours of light*. 1704.

[5] R. M. Evans. “Some Notes on Maxwell’s Colour Photograph”. In: *The Journal of Photographic Science* (1961).

[6] J. C. Maxwell. *The Scientific Letters and Papers of James Clerk Maxwell: 1846-1862*. Cambridge University Press, 1990.

[7] J. D. Mollon. “The origins of modern color science”. In: *The Science of Color* (2003).

[8] C. F. Borges. “Trichromatic Approximation for Computer Graphics Illumination Models”. In: *SIGGRAPH Comput. Graph.* (1991).

[9] M. B. Hullin, J. Hanika, B. Ajdin, H.-P. Seidel, J. Kautz, and H. P. A. Lensch. “Acquisition and analysis of bispectral bidirectional reflectance and reradiation distribution functions”. In: *ACM Trans. Graph.* (2010).

[10] A. Fichet, L. Belcour, and P. Barla. “Non-Orthogonal Reduction for Rendering Fluorescent Materials in Non-Spectral Engines”. In: *CGF* (2024).

[11] A. Weidlich, A. Forsythe, S. Dyer, T. Mansencal, J. Hanika, A. Wilkie, L. Emrose, and A. Langlands. “Spectral imaging in production”. In: *SIGGRAPH ’21 Courses*. 2021.

[12] M. Radziszewski, K. Boryczko, and W. Alda. “An Improved Technique for Full Spectral Rendering”. In: *Journal of WSCG* 17 (2009).

[13] A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika. “Hero Wavelength Spectral Sampling”. In: *Computer Graphics Forum* 33.4 (2014), pp. 123–131. DOI: 10.1111/cgf.12419.

[14] R. West, I. Georgiev, A. Gruson, and T. Hachisuka. “Continuous Multiple Importance Sampling”. In: *ACM Trans. Graph.* 39.4 (2020). DOI: 10.1145/3386569.3392436.

[15] G. F. Evans and M. D. McCool. “Stratified Wavelength Clusters for Efficient Spectral Monte Carlo Rendering”. In: *Graphics Interface*. 1999, pp. 42–49.

[16] D. L. MacAdam. “Maximum Visual Efficiency of Colored Materials”. In: *J. Opt. Soc. Am.* 25.11 (1935), pp. 361–367. DOI: 10.1364/JOSA.25.000361.

[17] B. Smits. “An RGB-to-Spectrum Conversion for Reflectances”. In: *Journal of Graphics Tools* 4.4 (1999), pp. 11–22. DOI: 10.1080/10867651.1999.10487511.

[18] J. Meng, F. Simon, J. Hanika, and C. Dachsbacher. “Physically Meaningful Rendering using Tristimulus Colours”. In: *Computer Graphics Forum* 34.4 (2015), pp. 31–40. DOI: 10.1111/cgf.12676.

[19] H. Otsu, M. Yamamoto, and T. Hachisuka. “Reproducing Spectral Reflectances From Tristimulus Colours”. In: *Computer Graphics Forum* 37.6 (2018), pp. 370–381. DOI: 10.1111/cgf.13332.

[20] W. Jakob and J. Hanika. “A Low-Dimensional Function Space for Efficient Spectral Upsampling”. In: *Computer Graphics Forum* (2019).

[21] C. Peters, S. Merzbach, J. Hanika, and C. Dachsbacher. “Using Moments to Represent Bounded Signals for Spectral Rendering”. In: *ACM Trans. Graph.* 38.4 (2019). DOI: 10.1145/3306346.3322964.

[22] M. Mackiewicz, H. J. Rivertz, and G. Finlayson. “Spherical sampling methods for the calculation of metamer mismatch volumes”. In: *J. Opt. Soc. Am. A* 36.1 (2019). DOI: 10.1364/JOSAA.36.000096.

[23] M. van de Ruit and E. Eisemann. “A Multi-Pass Method for Accelerated Spectral Sampling”. In: *Computer Graphics Forum* (2021). DOI: 10.1111/cgf.14408.

[24] V. Petitjean, P. Bauszat, and E. Eisemann. “Spectral Gradient Sampling for Path Tracing”. In: *Computer Graphics Forum* 37.4 (2018). DOI: 10.1111/cgf.13474.

[25] D. S. Immel, M. F. Cohen, and D. P. Greenberg. “A radiosity method for non-diffuse environments”. In: *ACM SIGGRAPH Computer Graphics* 20 (1986). DOI: 10.1145/15886.15901.

[26] J. T. Kajiya. “The Rendering Equation”. In: *SIGGRAPH Comput. Graph.* 20.4 (1986). DOI: 10.1145/15886.15902.

[27] E. Veach. “Robust Monte Carlo Methods for Light Transport Simulation”. PhD thesis. Stanford University, 1998.

[28] M. Mojzík, A. Fichet, and A. Wilkie. “Handling Fluorescence in a Uni-directional Spectral Path Tracer”. In: *Computer Graphics Forum* 37.4 (2018). DOI: 10.1111/cgf.13477.

[29] J. Lehtinen, T. Karras, S. Laine, M. Aittala, F. Durand, and T. Aila. “Gradient-domain metropolis light transport”. In: *ACM Transactions on Graphics* 32 (2013). DOI: 10.1145/2461912.2461943.

[30] M. Kettunen, M. Manzi, M. Aittala, J. Lehtinen, F. Durand, and M. Zwicker. “Gradient-domain path tracing”. In: *ACM Transactions on Graphics* 34 (July 2015). DOI: 10.1145/2766997.

[31] E. Eisemann and F. Durand. “Flash photography enhancement via intrinsic relighting”. In: *ACM Transactions on Graphics* 23 (2004). DOI: 10.1145/1015706.1015778.

[32] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. “Digital photography with flash and no-flash image pairs”. In: *ACM Transactions on Graphics* 23 (2004). DOI: 10.1145/1015706.1015777.

[33] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. “Joint bilateral upsampling”. In: *ACM Transactions on Graphics* 26 (July 2007). DOI: 10.1145/1276377.1276497.

[34] A. Buades, B. Coll, and J. Morel. “A non-local algorithm for image denoising”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2005.

[35] J. Vorba, J. Hanika, S. Herholz, T. Müller, J. Křivánek, and A. Keller. “Path Guiding in Production”. In: *ACM SIGGRAPH 2019 Courses*. 2019. DOI: 10.1145/3305366.3328091.

[36] W. Jakob. *Mitsuba - Physically Based Renderer*. https://www.mitsuba-renderer.org. 2010.

[37] G. Wyszecki and W. S. Stiles. *Color science: Concepts and Methods, Quantative Data and Formulae*. Wiley, 1982.

[38] BabelColor Company. *The Colorchecker Pages*. 2019. URL: http://www.babelcolor.com/colorchecker.htm.

[39] J. Roby and M. Aubé. *Lamp Spectral Power Distribution Database (LSPDD)*. https://lspdd.org. 2019.

[40] S. G. Narasimhan, M. Gupta, C. Donner, R. Ramamoorthi, S. K. Nayar, and H. W. Jensen. “Acquiring scattering properties of participating media by dilution”. In: *ACM Transactions on Graphics* 25 (July 2006). DOI: 10.1145/1141911.1141986.

[41] M. van de Ruit and E. Eisemann. “Metameric: Spectral Uplifting via Controllable Color Constraints”. In: *SIGGRAPH 2023 Conference Proceedings*. 2023. DOI: 10.1145/3588432.3591565.

[42] G. D. Finlayson and P. Morovic. “Metamer sets”. In: *J. Opt. Soc. Am. A* 22.5 (2005), pp. 810–819. DOI: 10.1364/JOSAA.22.000810.

[43] A. D. Logvinenko, B. Funt, and C. Godau. “Metamer mismatching”. In: *IEEE Transactions on Image Processing* 23.1 (2013), pp. 34–43. DOI: 10.1109/TIP.2013.2283148.

[44] E. Schrödinger. “Theorie der Pigmente von grösster Leuchtkraft”. In: *Annal. Der Phys.* 62 (1920), pp. 603–622.

[45] G. West and M. H. Brill. “Conditions under which Schrödinger object colors are optimal”. In: *J. Opt. Soc. Am.* 73.9 (1983), pp. 1223–1225. DOI: 10.1364/JOSA.73.001223.

[46] A. D. Logvinenko. “An object-color space”. In: *Journal of Vision* 9.11 (2009), pp. 5–5. DOI: 10.1167/9.11.5.

[47] X. Zhang, B. Funt, and H. Mirzaei. “Metamer mismatching in practice versus theory”. In: *J. Opt. Soc. Am. A* (2016), A238–A247. DOI: 10.1364/JOSAA.33.00A238.

[48] L. T. Maloney. “Evaluation of linear models of surface spectral reflectance with small numbers of parameters”. In: *J. Opt. Soc. Am. A* 3.10 (1986), pp. 1673–1683. DOI: 10.1364/JOSAA.3.001673.

[49] A. Weidlich, A. Forsythe, S. Dyer, T. Mansencal, J. Hanika, A. Wilkie, L. Emrose, and A. Langlands. “Spectral Imaging in Production: Course Notes Siggraph 2021”. In: *ACM SIGGRAPH 2021 Courses*. SIGGRAPH ’21. 2021. DOI: 10.1145/3450508.3464582.

[50] I. Mallett and C. Yuksel. “Spectral Primary Decomposition for Rendering with sRGB Reflectance”. In: The Eurographics Association, 2019. DOI: 10.2312/sr.20191216.

[51] A. Jung, A. Wilkie, J. Hanika, W. Jakob, and C. Dachsbacher. “Wide Gamut Spectral Upsampling with Fluorescence”. In: *Computer Graphics Forum* 38.4 (2019), pp. 87–96. DOI: 10.1111/cgf.13773.

[52] L. König, A. Jung, and C. Dachsbacher. “Improving Spectral Upsampling with Fluorescence”. In: *Workshop on Material Appearance Modeling*. The Eurographics Association, 2020. DOI: 10.2312/mam.20201139.

[53] L. Tódová, A. Wilkie, and L. Fascione. “Moment-based Constrained Spectral Uplifting”. In: *Eurographics Symposium on Rendering*. 2021. ISBN: 978-3-03868-157-1. DOI: 10.2312/sr.20211304.

[54] L. Tódová, A. Wilkie, and L. Fascione. “Wide Gamut Moment-based Constrained Spectral Uplifting”. In: *Computer Graphics Forum* 41.6 (2022), pp. 258–272. DOI: 10.1111/cgf.14617.

[55] M. S. Floater. “Mean value coordinates”. In: *Computer aided geometric design* 20 (2003).

[56] T. Ju, S. Schaefer, and J. Warren. “Mean value coordinates for closed triangular meshes”. In: *ACM Siggraph 2005 Papers*. 2005.

[57] J. Tan, J.-M. Lien, and Y. Gingold. “Decomposing Images into Layers via RGB-Space Geometry”. In: *ACM Trans. Graph.* 36 (2016). DOI: 10.1145/2988229.

[58] J. Tan, J. Echevarria, and Y. Gingold. “Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry”. In: *ACM Transactions on Graphics (TOG)* 37 (2018). DOI: 10.1145/3272127.3275054.

[59] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. “The quickhull algorithm for convex hulls”. In: *ACM Transactions on Mathematical Software (TOMS)* 22 (1996).

[60] H. Hoppe. “Progressive meshes”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 99–108.

[61] P. V. Sander, X. Gu, S. J. Gortler, H. Hoppe, and J. Snyder. “Silhouette clipping”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000.

[62] J. Cohen. “Dependency of the spectral reflectance curves of the Munsell color chips”. In: *Psychonomic science* 1.1 (1964), pp. 369–370.

[63] J. P. Parkkinen, J. Hallikainen, and T. Jaaskelainen. “Characteristic spectra of Munsell colors”. In: *JOSA A* 6.2 (1989), pp. 318–322.

[64] H. S. Fairman and M. H. Brill. “The principal components of reflectances”. In: *Color Research & Application* 29.2 (2004), pp. 104–110.

[65] D.-Y. Tzeng and R. S. Berns. “A review of principal component analysis and its applications to color technology”. In: *Color Research & Applications* 30.2 (2005), pp. 84–98.

[66] D. H. Foster, K. Amano, S. M. Nascimento, and M. J. Foster. “Frequency of metamerism in natural scenes”. In: *Josa a* 23.10 (2006), pp. 2359–2372.

[67] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar. “Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum”. In: *IEEE transactions on image processing* 19 (2010).

[68] S. Le Moan, S. T. George, M. Pedersen, J. Blahová, and J. Y. Hardeberg. “A database for spectral image quality”. In: *Image Quality and System Performance XII*. Vol. 9396. SPIE. 2015.

[69] C. Li, M. R. Luo, M. Pointer, and P. Green. “Comparison of real colour gamuts using a new reflectance database”. In: *Color Research & Application* 39.5 (2014), pp. 442–451.

[70] G. B. Dantzig. “Reminiscences about the origins of linear programming”. In: *Operations Research Letters* 1.2 (1982), pp. 43–48. ISSN: 0167-6377. DOI: https://IDOI.org/10.1016/0167-6377(82)90043-8.

[71] B. S. Bischoff, M. Botsch, S. Steinberg, S. Bischoff, L. Kobbelt, and R. Aachen. “OpenMesh–a generic and efficient polygon mesh data structure”. In: *In OpenSG Symposium*. 2002.

[72] J. Forrest, S. Vigerske, T. Ralphs, L. Hafer, J. Forrest, jpfasano, H. G. Santos, M. Saltzman, Jan-Willem, B. Kristjansson, h-i-gassmann, A. King, pobonomo, S. Brito, and to-st. *coin-or/Clp: Release releases/1.17.7*. Version releases/1.17.7. Jan. 2022. DOI: 10.5281/zenodo.5839302.

[73] O. Cornut. *Dear ImGui*. Version releases/1.89.2. Dec. 2022. URL: https://github.com/ocornut/imgui.

[74] P. Deutsch and J.-L. Gailly. *Zlib compressed data format specification version 3.3*. Tech. rep. 1996.

[75] W. Jakob, S. Speierer, N. Roussel, M. Nimier-David, D. Vicini, T. Zeltner, B. Nicolet, M. Crespo, V. Leroy, and Z. Zhang. *Mitsuba 3*. Version 3.0.1. https://mitsuba-renderer.org. 2022.

[76] H. A. Khan, S. Mihoubi, B. Mathon, J.-B. Thomas, and J. Y. Hardeberg. "HyTexiLa: High Resolution Visible and Near Infrared Hyperspectral Texture Images". In: *Sensors* 18.7 (2018). DOI: 10.3390/s18072045.

[77] M. van de Ruit and E. Eisemann. "Controlled Spectral Uplifting for Indirect-Light-Metamerism". In: *SIGGRAPH Asia 2024 Conference Papers*. 2024. DOI: 10.1145/3680528.3687698.

[78] M. van de Ruit and E. Eisemann. "Metameric: Spectral Uplifting via Controllable Color Constraints". In: *ACM SIGGRAPH 2023 Conference Proceedings*. 2023. DOI: 10.1145/3588432.3591565.

[79] E. Veach and L. J. Guibas. "Optimally combining sampling techniques for Monte Carlo rendering". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 1995.

[80] S. K. Nayar, K. Ikeuchi, and T. Kanade. "Shape from interreflections". In: *International Journal of Computer Vision* 6 (1991), pp. 173–195.

[81] D. Kraft. "Algorithm 733: TOMP–Fortran modules for optimal control calculations". In: *ACM Transactions on Mathematical Software* 20 (1994), pp. 262–281. DOI: 10.1145/192115.192124.

[82] S. G. Johnson. *The NLopt nonlinear-optimization package*. 2007. URL: https://github.com/stevengj/nlopt.

[83] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. "Qhull: Quickhull algorithm for computing the convex hull". In: *Astrophysics Source Code Library* (2013).

[84] G. Sharma, W. Wu, and E. N. Dalal. "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations". In: *Color Research & Application* 30.1 (2005), pp. 21–30. DOI: 10.1002/col.20070.

[85] C. Wyman, M. Kettunen, D. Lin, B. Bitterli, C. Yuksel, W. Jarosz, P. Kozlowski, and G. D. Francesco. "A Gentle Introduction to ReSTIR: Path Reuse in Real-time". In: *ACM SIGGRAPH 2023 Courses*. 2023. DOI: 10.1145/3587423.3595511.

[86] A. Jung, J. Hanika, and C. Dachsbacher. "Spectral Mollification for Bidirectional Fluorescence". In: *Computer Graphics Forum* (2020). DOI: 10.1111/cgf.13937.

[87] X. Luo, L. Scandolo, and E. Eisemann. "Texture Browser: Feature-based Texture Exploration". In: *Computer Graphics Forum*. 2021.

[88] M. B. Mark van de Ruit and E. Eisemann. “An Efficient Dual-Hierarchy t-SNE Minimization”. In: *IEEE Transactions on Visualization and Computer Graphics* (2021). DOI: 10.1109/TVCG.2021.3114817.

[89] V. v. Unen, T. Höllt, N. Pezzotti, N. Li, M. Reinders, E. Eisemann, A. Vilanova, F. Koning, and B. P. Lelieveldt. “Visual Analysis of Mass Cytometry Data by Hierarchical Stochastic Neighbour Embedding Reveals Rare Cell Types”. In: *Nature Communications* 8.1740 (2017). DOI: 10.1038/s41467-017-01689-9.

[90] B. Melit Devassy and S. George. “Dimensionality reduction and visualisation of hyperspectral ink data using t-SNE”. In: *Forensic Science International* 311 (2020). ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2020.110194.

[91] L. van der Maaten and G. Hinton. “Viualizing data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: https://www.jmlr.org/papers/v9/vandermaaten08a.html.

[92] S. Kullback. *Information Theory and Statistics*. New York: Wiley, 1959. ISBN: 0486696847.

[93] L. McInnes, J. Healy, and J. Melville. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: (2018). arXiv:1802.03426. arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426.

[94] N. Pezzotti, T. Höllt, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. “Hierarchical Stochastic Neighbor Embedding”. In: *Computer Graphics Forum* 35.3 (2016), pp. 21–30. DOI: 10.1111/cgf.12878. URL: http://graphics.tudelft.nl/Publications-new/2016/PHLEV16.

[95] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Höllt, E. Eisemann, and A. Vilanova. “Approximated and User Steerable tSNE for Progressive Visual Analytics”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.7 (2017), pp. 1739–1752. DOI: 10.1109/TVCG.2016.2570755.

[96] J. Tang, J. Liu, M. Zhang, and Q. Mei. “Visualization of Large-scale and High-dimensional Data”. In: *CoRR* abs/1602.00370 (2016). URL: http://arxiv.org/abs/1602.00370.

[97] L. van der Maaten. “Accelerating t-SNE using Tree-Based Algorithms”. In: *Journal of Machine Learning Research* 15.93 (2014), pp. 3221–3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.

[98] M. Kim, M. Choi, S. Lee, J. Tang, H. Park, and J. Choo. “PixelSNE: Pixel-Aligned Stochastic Neighbor Embedding for Efficient 2D Visualization with Screen-Resolution Precision”. In: *Computer Graphics Forum* 37 (2018), pp. 267–276. DOI: https://doi.org/10.1111/cgf.13418.

[99] D. M. Chan, R. Rao, F. Huang, and J. F. Canny. “T-SNE-CUDA: GPU-Accelerated T-SNE and its Applications to Modern Data”. In: *30th International Symposium Computer Architecture and High Performance Computing*. 2018, pp. 330–338. DOI: 10.1109/CAHPC.2018.8645912.

[100] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, and Y. Kluger. “Efficient Algorithms for t-distributed Stochastic Neighborhood Embedding”. In: *CoRR* abs/1712.09005 (2017). arXiv: 1712.09005. URL: http://arxiv.org/abs/1712.09005.

[101] N. Pezzotti, A. Mordvintsev, T. Höllt, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova. “Linear tSNE optimization for the Web”. In: *CoRR* abs/1805.10817 (2018). URL: http://arxiv.org/abs/1805.10817.

[102] P. Yianilos. “Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces”. In: *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. Vol. 93. Society for Industrial and Applied Mathematics, 1993. URL: https://dl.acm.org/doi/10.5555/313559.313789.

[103] J. Barnes and P. Hut. “A hierarchical O(N log N) force-calculation algorithm”. In: *Nature* 324 (1986), pp. 446–449. DOI: 10.1038/324446a0.

[104] C. D. Stolper, A. Perer, and D. Gotz. “Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1653–1662. DOI: 10.1109/TVCG.2014.2346574.

[105] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. “Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1643–1652. DOI: 10.1109/TVCG.2014.2346578.

[106] D. Kobak and G. C. Linderman. “UMAP does not preserve global structure any better than t-SNE when using the same initialization”. In: *bioRxiv* (2019). DOI: 10.1101/2019.12.19.877522. eprint: https://www.biorxiv.org/content/early/2019/12/19/2019.12.19.877522.full.pdf. URL: https://www.biorxiv.org/content/early/2019/12/19/2019.12.19.877522.

[107] J. Johnson, M. Douze, and H. Jégou. “Billion-scale similarity search with GPUs”. In: *IEEE Transactions on Big Data* (2019). arXiv:1702.08734. DOI: 10.1109/TBDATA.2019.2921572.

[108] B. H. Meyer, A. T. R. Pozo, and W. M. N. Zola. “Improving Barnes-Hut t-SNE Scalability in GPU with Efficient Memory Access Strategies”. In: *2020 International Joint Conference on Neural Networks*. 2020. DOI: 10.1109/IJCNN48605.2020.9206962.

[109] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha. “Fast BVH construction on GPUs”. In: *Computer Graphics Forum* 28 (2009), pp. 375–384. DOI: 10.1111/j.1467-8659.2009.01377.x.

[110] K. Garanzha, J. Pantaleoni, and D. McAllister. “Simpler and Faster HLBVH with Work Queues”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. Association for Computing Machinery, 2011, pp. 59–64. DOI: 10.1145/2018323.2018333.

[111] NVIDIA. *CUB: Cooperative primitives for CUDA C++*. Accessed: 2021-06-24. [Online]. Available: https://github.com/NVIDIA/cub. 2010. URL: https://github.com/NVIDIA/cub.

[112] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. “Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization”. In: *Journal of Machine Learning Research* 11 (2010), pp. 451–490. ISSN: 1532-4435. URL: https://dl.acm.org/doi/10.5555/1756006.1756019.

[113] H. Xiao, K. Rasul, and R. Vollgraf. “Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: *CoRR* abs/1708.07747 (2017). URL: http://arxiv.org/abs/1708.07747.

[114] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009. DOI: 10.1109/CVPR.2009.5206848.

[115] C. Fu, Y. Zhang, D. Cai, and X. Ren. “AtSNE: Efficient and Robust Visualization on GPU through Hierarchical Optimization”. In: *25th International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, 2019, pp. 176–186. ISBN: 9781450362016. DOI: 10.1145/3292500.3330834.

[116] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Vol. 2. 2013, pp. 3111–3119. DOI: 10.5555/2999792.2999959.