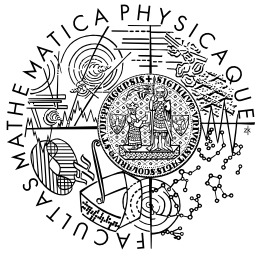


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Lucia Tódová

Constrained Spectral Uplifting

Department of Software and Computer Science Education

Supervisor of the master thesis: doc. Alexander Wilkie, Dr.

Study programme: Computer Science

Study branch: Computer Graphics and Game Development

Prague 2021

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to express my sincere gratitude to my supervisor, doc. Alexander Wilkie, Dr., for his time, guidance and patience over the last year. I would also like to thank my boyfriend, my family and my friends for their support and encouragement.

Title: Constrained Spectral Uplifting

Author: Bc. Lucia Tódová

Department: Department of Software and Computer Science Education

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

Abstract: Physically-based spectral rendering is becoming increasingly popular in both commercial and academic areas due to its ability to accurately simulate natural phenomena. However, the production of materials defined by their spectral properties is a tedious and expensive process, which makes the utilization of RGB-based assets in spectral renderers a desired feature. To convert RGB values to their spectral representations, a process called spectral uplifting is employed. As the RGB color space is a finite subset of the visible gamut, there exist multiple conversion techniques producing distinct results, which may cause color inconsistencies under various lighting conditions. This thesis proposes a method for constraining the spectral uplifting process. To be specific, pre-defined mappings of RGB values to their spectral representations are preserved and the rest of the RGB gamut is plausibly uplifted. In order to assess its correctness, this technique is then implemented and evaluated in a spectral renderer. The renders uplifted via our method show minimal discrepancies when compared to the original textures.

Keywords: spectral uplifting spectral rendering

Contents

Introduction

Over the last few years, the demand for physical accuracy of the rendering process has grown substantially. By providing the renderer with the capability to physically simulate light transport, we can recreate natural phenomena such as metamerism, fluorescence, polarization, etc... Such simulations require the internal color representation to be as close to the visible light's spectral power distribution as possible. However, the advantages of these representations are often not worth implementing due to their complexity.

Therefore, vast majority of the conventional renderers internally represent color as an RGB tristimulus value, mainly due to the simplicity and the robustness of such systems. Additionally, almost all of the already existing assets, such as input textures and materials, are defined in RGB, as their creation and utilization is fairly simple. Spectral assets require a real-life model whose reflectance must be measured with a spectrometer, which is both tedious and, in many cases, even impossible.

To preserve the physical correctness of the spectral rendering process while utilizing RGB textures as its input, a conversion from the color's RGB representation to its spectral variant is needed. Such process is called *spectral uplifting*, also known as *spectral upsampling*.

However, as the RGB color space is intrinsically smaller than the gamut of visible light, there exist multiple spectral representations of the same RGB color. Such spectral variations are called *metamers*, and can cause color discrepancies under different lighting conditions. Various uplifting techniques might therefore provide different results, none of which necessarily have to match the real measured spectra.

The goal of this thesis to create an uplifting model with the capability to constrain itself with pre-defined mappings from RGB to specific spectral shapes, which must be preserved during the uplifting process. All the other RGB input values should return plausible synthetic data which smoothly interpolates the measured spectra.

Related work

Currently, there exist several approaches to spectral uplifting. They differ in the type of spectra they are capable of reconstructing, the gamut they can uplift, the color error caused by round trips, etc...

Unfortunately, many of them have issues. The technique by MacAdam [25] is capable of creating only blocky spectra, which are unsuitable for smooth reflectances usually found in nature. The proposal by Smits [41], although more widely used, is prone to slight round-trip errors, which arise from out-of-range spectra. One of the first approaches that produced smooth spectra was proposed by Meng et al. [28]. However, they did not take energy conservation into account, which resulted in colors with no real physical counterpart, i.e. no real material could produce such color. Otsu et al. [33] introduced a technique that is capable of outperforming most of the existing approaches under specific conditions. Its drawback is its inability to satisfy the spectral range restrictions, which again

causes color errors upon round trips.

In this thesis, we focus mainly on the technique proposed by Jakob and Hanika [19], which employs a pre-built model that is based on spectral representation using sigmoids. This algorithm produces smooth spectra satisfying spectral range restrictions with negligible error.

Jung et al. [20] further improve this technique for wide gamut spectral uplifting by introducing new parameters for fluorescence. Currently, this approach can be considered as state of the art.

None of the existing techniques propose a way in which to constrain the uplifting process.

Layout of this thesis

This thesis is structured as follows:

In Chapter 1, we introduce the reader to light transport and color science. We explain the basic principles of human color perception, overview the existing color representations and focus on their importance in rendering.

Chapter 2 summarizes the already existing spectral uplifting algorithms, placing special emphasis on the technique by Jakob and Hanika [19]. Furthermore, it discusses the theory behind representation of spectra using moments.

Chapter 3 details the implementation of our constrained spectral uplifting model and its utilization in a rendering software. It often refers to chapter 4, where, in addition to discussing our results and comparing them to the non-constrained spectral uplifting, we overview multiple experiments that assess the correctness of various possible implementation choices.

1. Color Science

Color science, or colorimetry, is a branch of science that concerns itself with human perception of color. It researches the relations between human vision and physical properties of color, and analyzes options for both its capturing and reconstruction.

We begin this chapter by describing the physical properties of light and their subsequent meaning in terms of color. We then provide multiple options for quantifying said color for further possible reconstruction in the digital world. Lastly, we show the importance of color representation in modern-day renderers, and its effects on reconstruction of physically based phenomena.

1.1 Light and Color

The term *human visual perception* refers to the ability of the human eye to interpret the surrounding environment. It is based on our capability to detect *electromagnetic radiation*, which is a form of energy that consists of waves which propagate through space and transmit radiant energy.

An *electromagnetic wave* is characterized by its *amplitude* and *frequency*. Amplitude is defined as the distance between the central axis and either the *crest* (the highest point of the wave) or the *trough* (the lowest point of the wave), while frequency specifies how many wave cycles occur in a second. Together, these properties give rise to the term *wavelength*, denoted λ, which measures the length of the wave — the distance between either two subsequent crests, troughs or any two following spots with the same height.

Every electromagnetic wave can be unambiguously defined by its wavelength. Arranging them according to this criterion creates a classification known as *electromagnetic spectrum* (see fig. 1.1). As the electromagnetic spectrum contains all existing types of electromagnetic radiation, it covers wavelengths in the range from fractions of nanometers to thousands of kilometers. This range can be divided into bands to distinguish known types of electromagnetic waves, from low frequency gamma or X-rays to high frequency radio waves.

In this thesis, we focus on *visible light*, which covers only a mere fraction of the electromagnetic spectrum. Its waves are roughly in the 380-780nm range.

To sum up, electromagnetic waves specify the way in which light travels. To, however, describe the interaction between light and matter, the term *photon* is used.

Photons are elementary particles of light moving in a manner specified by their wavelengths. They make up electromagnetic radiation and can be emitted or absorbed by atoms and molecules. During this process, they transfer energy either from the object that emitted them or to the object that absorbed them. This change in energy (denoted E) is proportional to the wave frequency of the absorbed/emitted photon and can be computed as follows [12]:

$$E = hf = \frac{hc}{\lambda}, \tag{1.1}$$

where h is the Planck's constant, f is the wave frequency and c is the speed of

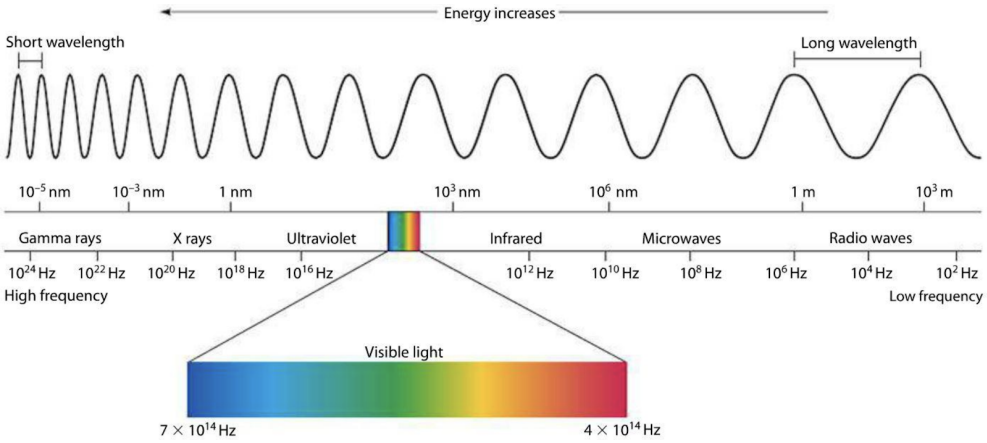


Figure 1.1: An illustration of the electromagnetic spectrum [3]

light. Therefore, generally speaking, the human eye identifies light when atoms and molecules in the retina absorb photons.

To specify this process, we first describe the retina. The retina consists of millions of light-sensitive cells, also called *photoreceptors*, which pass a visual signal via an optic nerve to the brain, giving the notion of light and color. There are two types of photoreceptors in the human eye — rods and cones.

Rods make up most of the receptor cells (around 91 million according to Purves et al. [36], but other sources state that their number could be as high as 125 million [47]). They are usually located around the boundary of the retina, and are responsible for low light (*scotopic*) vision. However, they possess very little notion of color, which is also the reason why the human eye has trouble recognizing colors during the night.

Cones are located mainly in the center of the retina and their numbers are a lot lower (from around 4.5 million [36] to 6 million [47]). In contrast to rods, they are active at daylight levels (responsible for *photopic* vision) and have the notion of color. To be specific, different types of cones differ in their sensitivity to photon energies at concrete wavelengths. The final color is then composed by the brain from the stimulation signals sent by each cone.

The human eye has three types of cones:

- *L-cones*, which are the most responsive to longer wavelengths at around 560nm. When stimulated, they correspond to the red color.
- *M-cones*, which are the most sensitive to medium wavelengths at around 530nm and correspond to green color.
- *S-cones*, which respond the most to small wavelengths that peak at around 420nm and correspond to blue color.

Their relative response to stimulation can be seen in fig. 1.2.

This type of color perception is called *trichromatic*, as it uses three types of receptors to create the whole color space. The attempts to simulate such perception in the digital world give rise to tristimulus color representations, which have been widely adapted in color science. We discuss these thoroughly in section 1.2.

Up until now, we have discussed the interaction of light with the human eye. Photons, however, also interact with objects. As established by the relationship

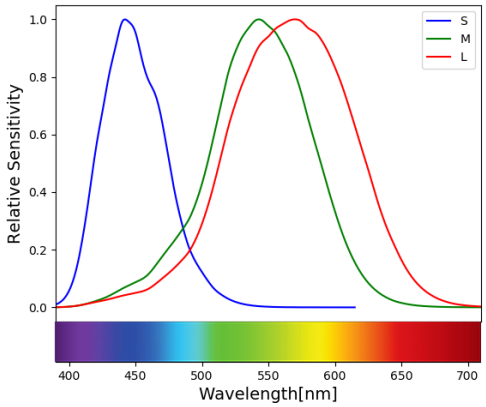


Figure 1.2: Relative sensitivity of S, M and L-cones plotted according to the data measured by Stockman and Sharpe [42]

defined in eq. (1.1), the energy transferred to an object upon light interaction is dependent on the photon wavelength. This means that objects might absorb some wavelengths and reflect others.

Object color is defined by the wavelengths it *reflects*. For example, if it reflects all the wavelengths, the resulting color is white, while absorbing all the wavelengths would render the object black. Naturally, human perception of object color is not only dependent on its reflective properties, but also on the lighting of the scene. If the only present light is red, wavelengths corresponding to colors other than red never hit the object. Therefore, the object might reflect only a subset of wavelengths than it would under white light, which may subsequently result in a change of the perceived color.

1.2 Color representation

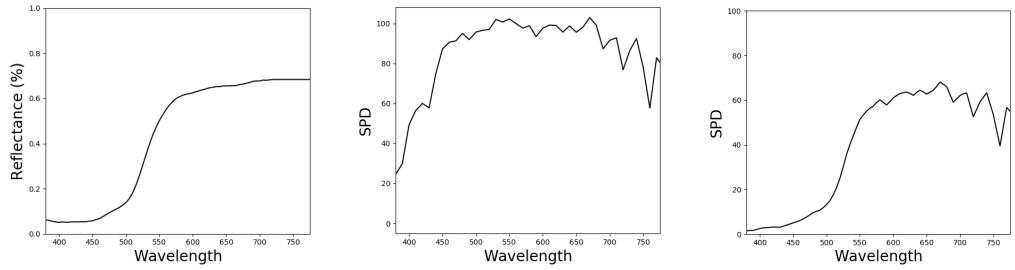
The question of how to discretely represent color has been posed ever since the introduction of the first graphical user interface. For use in computer science, representations are required to be compact, precise, and the operations on colors are required to be simple and easily executed.

In this section, we describe the spectral representation, which is based primarily on the physical properties of color. Then, we overview the basic properties of the most popular tristimulus systems.

1.2.1 Spectral representation

When defining the color of an object, we must not only specify the wavelengths it reflects, but also the ratio between the incoming and the outgoing energy at these wavelengths. The dependency of reflectance on the wavelength is called a *reflectance spectrum*, and is usually a smooth, continuous curve (see example in fig. 1.3a).

Although this definition might be sufficient for reflective surfaces, describing the color emitted by a light source requires the knowledge of the source's power rather than reflectance. For these purposes, *spectral power distribution* (SPD)



(a) Reflectance spectrum (b) Emission spectrum (c) Combination

Figure 1.3: The behavior of a reflectance of a material under an illuminant specified by an emission spectrum

is used. Generally, SPD is a function describing the relationship between wavelength and any radiometric or photometric quantity (radiant energy, luminance, luminous flux, irradiance, etc. . .). In this thesis, however, we use SPD to describe the emissive properties of light sources, and therefore consider SPD to be a function of wavelength and power. We provide an example of an emission spectrum in fig. 1.3b.

As the reflectance spectrum serves as a throughput for the emission spectrum, the color of an object illuminated by a light source can be determined by multiplying the light source's SPD curve with the reflectance curve of the object, as shown in an example in fig. 1.3. This way the physical properties of color are preserved and the results are the same as they would be in nature.

1.2.2 Tristimulus representation

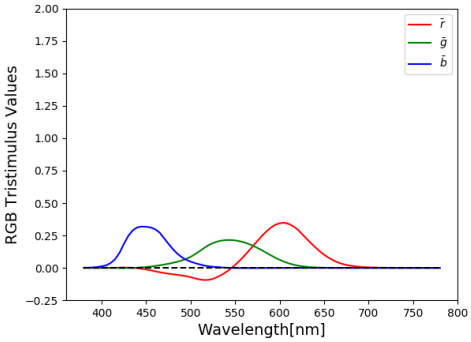
The obvious drawback of the spectral representation is the difficulty of its discretization, since there is an infinite number of possible spectral curves, but only a finite number of digital colors that can be displayed by a monitor. For the purposes of color visualization, a distinct, discrete color representation is required.

Tristimulus representation approaches this issue by saving the color as a set of three values. Although the original idea was to simulate the trichromatic perception of human eye (i.e. save values that specify how much have the red, green and blue cones been stimulated), over time, multiple other tristimulus color spaces have been created. They differ mostly in the range of colors they are capable of representing and in their practical use. Following, we provide an overview of some of the most popular ones.

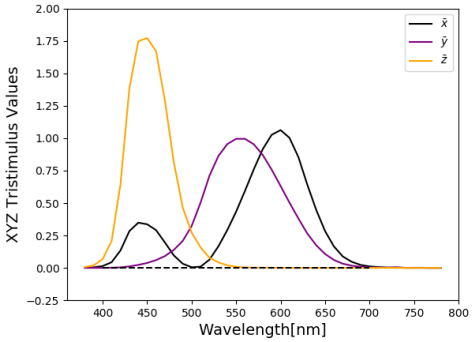
RGB color space

The RGB color space is an additive space employing three primaries — red, green and blue. In other words, if three lasers with red, green and blue chromacities are used to illuminate a single point, any color within the RGB color space can be created solely by changing the lights' intensities.

An RGB value can therefore be thought of as a point in a 3-dimensional Euclidean space with each of the coordinate axes representing one of the primaries. As the lights' intensities must be bounded, this space is narrowed down to a cube



(a) $\overline{r}(\lambda)$, $\overline{g}(\lambda)$ and $\overline{b}(\lambda)$ functions plotted with data by Broadbent [5]



(b) $\overline{x}(\lambda)$, $\overline{y}(\lambda)$ and $\overline{z}(\lambda)$ functions plotted with data by Choudhury [8]

Figure 1.4: Color matching functions

starting at the base of the coordinate system. Usually, the range for each value is defined within 0 and 255, but a normalized $[0, 1]$ range is also used.

Various implementations of the RGB color space exist. They differ in the specifications of the RGB primaries, and therefore in their *color gamut*, which is the subset of colors they are capable of representing. Some examples (named in ascending order with respect to the size of their color gamut) include ISO RGB, sRGB, Adobe RGB, Adobe Wide Gamut RGB and ProPhoto RGB. An illustrative comparison of the sRGB and Adobe RGB gamut in the chromaticity diagram (described thoroughly in section 1.2.2) can be seen in fig. 1.5.

RGB color spaces are commonly used in everyday world, e.g. in LED displays, digital cameras, scanners and even in computer graphics rendering. Their main downside has, however, been discovered when designing color matching functions [11].

A *color matching function* is a function designed to simulate the response of a certain type of cone in the human eye. In 1931, CIE designed a set of three color matching functions that could be used for spectral to RGB conversion [11]. Denoted $\overline{r}(\lambda)$, $\overline{g}(\lambda)$ and $\overline{b}(\lambda)$, they approximate the response of the L, M and S cones respectively. However, as seen in figure fig. 1.4a, the functions may also acquire negative values. At the time, this posed a problem due to calculation errors. Therefore, to eliminate these negative portions of functions, CIE designed a new color space — the XYZ color space.

XYZ color space

The XYZ color space is a reference color space capable of encompassing all colors perceptible by the human eye. Its color matching functions, $\overline{x}(\lambda)$, $\overline{y}(\lambda)$ and $\overline{z}(\lambda)$, were specifically designed for the purposes of SPD to tristimulus conversion, which is computed using the following equations:

$$\begin{aligned} X &= \int P(\lambda)\overline{x}(\lambda)d\lambda, \\ Y &= \int P(\lambda)\overline{y}(\lambda)d\lambda, \\ Z &= \int P(\lambda)\overline{z}(\lambda)d\lambda, \end{aligned} \tag{1.2}$$

where X, Y and Z are the resulting tristimulus values and $P(\lambda)$ is the spectral power distribution.

Although the X, Y and Z primaries were designed so that the Y primary closely matches luminance and X and Z primaries give color information, they are only imaginary, i.e. they do not correspond to any spectral distribution of wavelengths. This property renders the whole XYZ space imaginary, which means that it cannot be used for visualization purposes. Its main function is to serve as a "middle step" when performing a conversion from SPD to an arbitrary tristimulus space, which eliminates the need for creating color matching functions for other tristimulus spaces. The conversion from XYZ to an arbitrary tristimulus space can then be performed by a simple space-specific $3x3$ matrix transformation.

xyY color space

In addition to the impossible visualization process, another downside of the XYZ color space is that its values are practically unbounded and do not have any real meaning (such as the RGB triplets have). Therefore, a more intuitive color space has been created, which considers the relative proportions of the X, Y and Z values rather than their unbounded versions — the xyY color space [21]. It is based on the assumption that color can be regarded as a quantity with two properties: *luminance* and *chromaticity*.

The conversion from the XYZ to the xyY color space is performed as follows — first, the X, Y and Z values are converted to their bounded versions (also called *chromaticity coordinates*) as defined in eq. (1.3) [13].

$$\begin{aligned} x &= \frac{X}{X+Y+Z} \\ y &= \frac{Y}{X+Y+Z} \\ z &= \frac{Z}{X+Y+Z} \end{aligned} \tag{1.3}$$

Since $x+y+z = 1$, the z term can be expressed as $z = 1-x-y$. This means that z does not give us any additional information about the current color and therefore can be dropped from the representation. It also implies that some information has been lost during the conversion, i.e. we cannot reconstruct the original XYZ triplet using only the x and y values and therefore cannot obtain the initial color. At least one of the original values is needed for this purpose — CIE [9] decided to use the Y component, as it already specifies luminance.

Plotting the values of the x and y chromaticity coordinates creates a *chromaticity diagram*, shown in fig. 1.5. Each point of the curved boundary line (also called the *spectral locus*) corresponds to an XYZ value that is the result of a monochromatic radiation (i.e. a single-wavelength stimulus). All other chromaticities visible to the standard observer lie within a region bounded by the spectral locus.

L*a*b*

Although some of the color spaces mentioned so far are already quite intuitive in terms of human color perception (e.g. xyY), neither of them regards for the

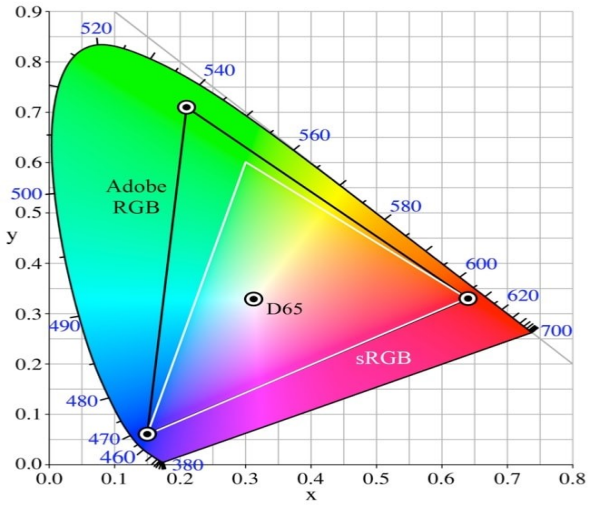


Figure 1.5: An illustrative comparison of the sRGB and Adobe RGB gamut in the chromaticity diagram based on images created by Choi et al. [7]

human perception of color differences. The human eye is, for example, more prone to spotting differences when comparing lighter pastel colors and neglecting them upon interaction with darker color (e.g. dark blue). If we wish to accurately describe color differences in a color space, we must regard for this factor and aim for *perceptual uniformity*, i.e. for the difference between two colors as perceived by the human eye to be proportional to their Euclidean distance within the color space.

The Hunter's Lab color space [49] addresses this issue and is designed so that the distance between its two triplets characterizes roughly how different they are in chromaticity and luminance. It is based on the Opponent color theory [22], which suggests that the cones in the human eye are linked together in opposing pairs and that the visual system records the *difference* between the stimulation of the pairs rather than the cones' individual responses.

As the Hunter's Lab color space does not achieve perfect uniform spacing of values, CIE *L*a*b** color space (CIELAB) [17] has been proposed in an attempt to improve some of its shortcomings and is now more widely used. However, neither of the systems are completely accurate in terms of perceptual uniformity [49].

The three opponent channels used to specify color in the CIE L*a*b* color space are defined as follows [17]:

- *L** — indicates lightness, i.e. the difference between *light* and *dark*. Its values range from 0 (yielding black color) to 100 (indicating diffuse white color).
- *a** — defines the difference between *green* and *red*. Positives values of this component indicate the object's color to be more green, while negative values signify the domination of red.
- *b** — defines the difference between *yellow* and *blue*. Positive values indicate the object to be more yellow, while negative values indicate the domination of blue.

Neither the range of the a* nor the b* component has any specific numerical limits [17].

The L*a*b* color space is a *reference system* — an abstract, non-intuitive space encompassing all human-perceptible colors. Due to its perceptual uniformity, it can be used for color balance corrections by modifying the a* and b* components, and for lightness adjustments by modifying the L* component. However, as mentioned earlier, its main purpose is the determining of *color differences*.

In 1976, CIE introduced the concept of *Delta E*, which is the measure of change in visual perception of two colors [16]. Denoted ΔE^*_{ab}, it is computed as an Euclidean distance between two sample points, i.e.:

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2}, \tag{1.4}$$

where (L^*_1, a^*_1, b^*_1) and (L^*_2, a^*_2, b^*_2) are the L*a*b* coordinates of the sample points.

However, the ΔE^*_{ab} error is prone to exaggerating the differences occurring when comparing two highly saturated colors of similar hue. This is due to the fact that the L*a*b* color space is not perfectly perceptually uniform, which can be perceived upon observation of these regions. To improve upon these shortcomings, other measuring techniques for computing Delta E, such as *Delta94* and *Delta2000*, have been proposed over the years.

Delta94 is computed by first modifying the original L*a*b* values of both colors to compensate for perceptual distortions in the color space and then by computing the Euclidean distance from the modified values. Although the results match the human color difference perception more closely, the Delta94 error metric still lacks some accuracy in the blue-violet region [16].

Delta2000 attempts to remove these inaccuracies. Including the corrections added to Delta94, Delta2000 overall adds five correctional factors to the original ΔE^*_{ab} — compensation factors for lightness, hue and chroma, compensation for neutral colors and, lastly, a hue rotation term for the problematic blue-violet region.

From the listed Delta E equations, the Delta2000 error measurements are the most accurate in terms of human color difference perception [16]. However, in addition to being computationally intensive, the Delta2000 equation is discontinous [40]. Therefore, it is not a preferred measure to use in gradient-based optimizers, which is an observation we use in the practical part of this thesis.

Other color spaces

In addition to the already mentioned tristimulus color spaces, there exist many more used for various purposes. Following, we briefly overview some of them:

- *L*u*v** — Similarly to the CIELAB system, L*u*v* (or CIELUV) aims for perceptual uniformity. As a matter of fact, the $L*$ value is defined in the same manner as in the CIELAB system, while u and v values are evaluated by certain projections of the x and y coordinates of the chromaticity diagram. When comparing their Euclidean error measure, the most important distinction between the two spaces is that while the CIELAB generally outperforms CIELUV in terms of perceptual uniformity [27], CIELUV does not have as many inaccuracies in the dark regions [39]. Therefore, it is often

recommended to use the CIELUV color space for characterization of color displays and the CIELAB color space for the characterization of colored surfaces and dyes.

- *HSL* and *HSI* color spaces define color by its *hue*, *saturation* and *lightness* (or *intensity*). They are alternative representations of the RGB color space and must therefore be defined purely with reference to an RGB space [18]. As their components correlate more intuitively with human perception of color than those of the RGB system, they are often used in image processing applications, e.g. for processes such as feature detection (edge detection [45], object recognition) or image segmentation (which can be performed solely by using the hue component) [18].

- *CMYK* model is a subtractive color model commonly used in color printing. This means that assigning zero values to all components renders white light, and increasing the value of a component specifies how much of the respective color is *subtracted* from the white light. It is based on RGB's complementary colors — *cyan*, *magenta* and *yellow* (respectively). Although the premise is that maximizing CMY values should render perfect black, in reality, the printing inks are not 100% pure CMY and their combinations therefore cannot produce rich black. For this purpose, a fourth component, *black* (K), is often added.

Other color spaces include the Munsell color system, RAL, Natural Color System, Pantone Matching System, $\text{CIELCH}_{\text{ab}}$, $\text{CIELCH}_{\text{uv}}$, etc...

1.2.3 Color representation in rendering

Rendering is an image synthesis process which internally utilizes a light transport simulation in order to mimic natural behavior of light and material colors based on a given scene description. The output of such process is an image called a *render*.

Accurate color representation is the core of rendering softwares. Although most of today's renderers support multiple color spaces, we can still divide them into two main categories according to the space used during evaluation of light transfer equations — *tristimulus* and *spectral* renderers.

Tristimulus renderers are usually based on the RGB color space, although they often offer conversions to other tristimulus spaces. Due to the ease of use and simplicity of representation, RGB renderers are more common in commercial rendering software. They provide realistically looking images, often indistinguishable from a photograph, and are more robust, memory efficient and easy to implement.

However, light in real world does not travel as a tristimulus value, but rather as a distribution of wavelengths. Therefore, RGB renderers cannot properly simulate the physical properties of color during reflections or refractions.

Spectral rendering, on the other hand, uses full-spectral information of all materials and lights in the scene throughout the whole rendering process. Obviously, before visualization occurs, spectral information must be converted into tristimulus (usually RGB) values, but this does not pose a problem as, at the moment of conversion, all the physically based simulations have already taken

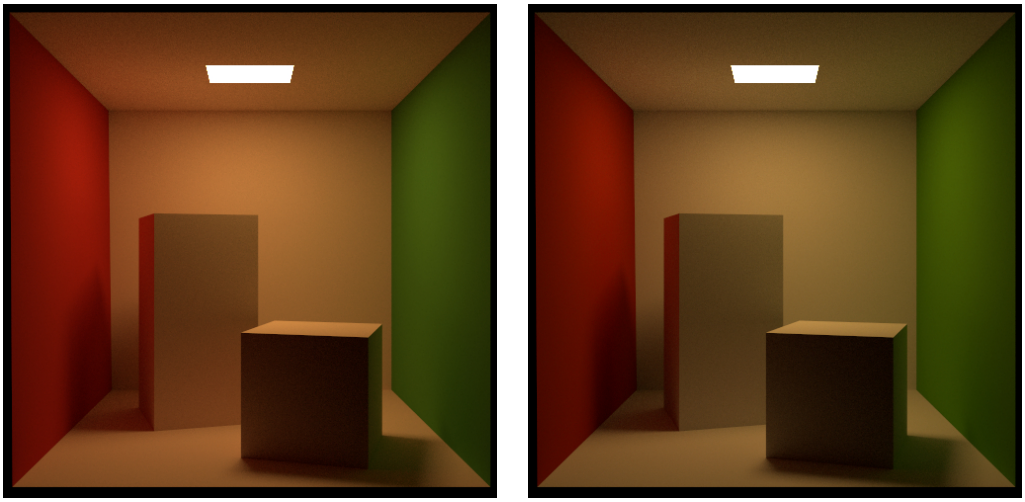


Figure 1.6: Comparison of an RGB-based rendering and spectral-based rendering as presented in the documentation of Mitsuba2 [46]. Left: Spectral reflectance data of all materials is first converted to RGB and the scene is then rendered in the RGB mode. Right: Scene is rendered directly in the spectral mode.

place. Therefore, the rendered scene appears more realistic. We demonstrate this difference in fig. 1.6, on a scene already rendered by Mitsuba2 [46]. As can be observed, while the scene rendered in the RGB mode produces an unnaturally saturated image, the spectrally-based render results in more realistic colors.

In addition to a more convincing rendering of reflections and refractions, another reason for using spectral rendering is its capability to simulate physically based phenomena that arise due to the interaction of light with specific materials. Following, we overview some of the most common ones:

- *Metamerism*

  As already mentioned in section 1.2.2, the human tristimulus perception has a significantly lower domain than the spectral domain. Therefore, two different spectra can trigger the same cone response in the human eye and appear to have the same color (and, subsequently, to have the same RGB values), giving rise to a phenomenon called *metamerism*. The two spectra evaluating to the same tristimulus values are called *metamers*.

  In real world, metamerism is often perceived when the lighting conditions under which we observe metamers change. An example of this can be seen in fig. 1.7, where the color of the presented spheres is similar under the D65 illuminant, but clearly differs under the fluorescent F11 illuminant.

  As an RGB renderer does not possess spectral information, it cannot replicate the behavior of spectral reflectance under an illuminant, and is therefore unable to reproduce metamerism.

- *Fluorescence*

  By definition, fluorescence occurs when light from one excitation wavelength λ_0 is absorbed by an object and is almost immediately re-emitted at a different, usually longer, wavelength λ_1 [15]. Specifically interesting is the fact that the absorbed light can come from outside of the visible spectrum and

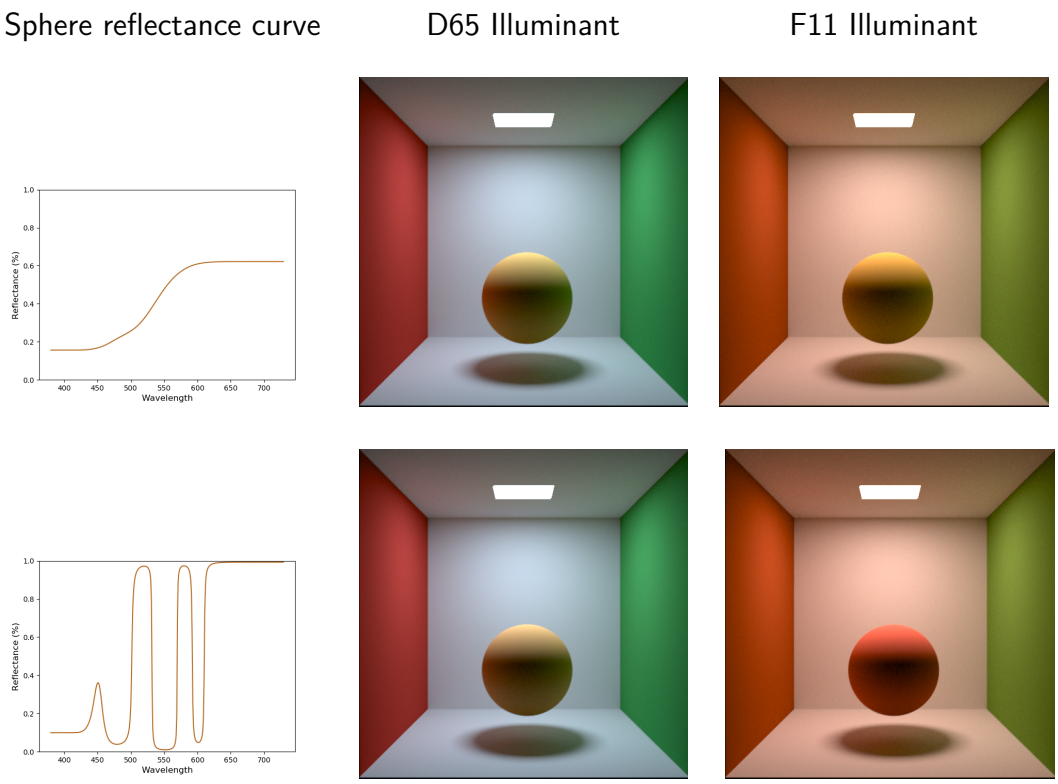


Figure 1.7: The effects of metamerism

be re-emitted inside it, which results in an unrealistically bright appearance of materials. This can be perceived in real world when, for example, fluorescent fish, corals, jellyfish or even minerals are illuminated by a UV light.

RGB renderers attempt to fake this behavior through custom shaders [51]. As they often produce satisfactory results and, in comparison to physical simulation, are immensely easier to implement, physically based fluorescence has received small amount of work. Its support can be found in spectral renderers, added for example to ART by Mojzík [29].

- *Iridescence*

  Iridescence, or goniochromism, is a phenomenon occurring when certain surfaces change color according to the current viewing angle. It arises when the object's physical structure causes interferences between light waves (e.g. inside extremely thin dielectric layers), yielding rich color variations [4]. It can be perceived in nature in certain plants, specific minerals, butterfly wings, peacock's feathers, snakes, but also in man-made products such as oil leaks, soap bubbles or car paints.

  Similarly to fluorescence, iridescent behavior can be faked in an RGB renderer [43]. However, research based on physical properties of iridescence has also been conducted. For further information about the current development, we refer the interested reader to the articles by Belcour and Barla [4], Sadeghi and Jensen [38], or Werner et al. [48].

- *Dispersion*

When light travels from one medium to another (e.g. when light coming from air hits glass or water), its direction of travel changes. This phenomenon is called *refraction* and is closely described by Snell's law, which specifies how the angle of refraction can be computed from the angle of incidence and the *refraction indices* of the two media [10]. However, the refraction index depends not only on the *type* of media, but also on the current *wavelength* [44] — which implies that the resulting direction of photons may vary according to their wavelength.

Probably the most known scenario displaying this phenomenon is white light hitting a dispersive prism. Upon interaction, light is split into spectral bands, creating a "rainbow" effect.

There have been multiple attempts to simulate physically based dispersion. We refer the interested reader to articles by Sun et al. [44] or Wilkie et al. [50].

- *Polarization*

  Electromagnetic waves traveling through space are *transverse waves* — their oscillation is perpendicular to their path of propagation. By default, the directions of oscillations are arbitrary for each photon — this type of light is called an *unpolarized light*. Restrictions to the directions of oscillations (also called *polarization*) render *polarized light*. Such phenomenon usually occurs upon light's interaction with certain materials, called polarizers.

  The polarization process contributes to the overall color only in special cases (e.g. when using polarization filters) [51]. Therefore, it receives little attention in implementation of rendering softwares. However, for physical consistencies (and due to the possibility of unusual scenes) both ART [31] and Mitsuba [46] follow the direction of oscillation during the rendering process.

Other researched phenomena (some of it closely linked to the already mentioned ones) include *phosphorescence*, *bioluminescence*, *dichroism*, *opalescence*, *aventurescence* and many more.

2. Spectral Uplifting

While spectral rendering offers many advantages in terms of physical correctness, it is often neglected and traded off for the RGB color representation due to its ease of use and memory efficiency. Even the physically-based phenomena can be "faked", and, therefore, many conventional rendering systems are RGB-based.

Another reason for the disinclination towards spectral renderers is the deficiency of spectral textures. The process behind their creation is, in comparison to RGB-based textures, a lot more complicated. It usually requires a real-life model whose reflectance spectra must be measured with a spectrometer, which is, in many cases, virtually impossible.

Spectral renderers are, nonetheless, used both in the research and in the commercial sphere (e.g. ART, Mitsuba, Manuka). To make spectral rendering possible while utilizing RGB textures as input, a reliable conversion from RGB to spectral data has been proposed. We refer to this process as *spectral uplifting*, however, other sources also use the term *spectral upsampling* [20].

Converting an RGB value into its respective spectrum poses multiple difficulties. As the relationship between the spectral and the RGB domain is not bijective (specifically, infinitely many spectral distributions render the same RGB values), distinctive approaches to the conversion process may render different spectral distributions. Although all of them might be correct in terms of the resulting RGB value, it is possible that none of them would be identical to spectral distribution measured with a spectrometer.

This does not cause a problem under standard illuminant with regard to which the RGB values were uplifted. However, as already mentioned in section 1.2.3, changing the illuminating conditions may result in a rather significant color change due to metamerism. This may cause issues with spectra created by an uplifting process, which are, with the current technology, arbitrary metamers. Therefore, the appearance of assets defined via RGB can be somewhat unpredictable under changing illumination, which is in turn unacceptable for e.g. workflows with very high demands on visual consistency between plate footage and VFX renders.

We demonstrate this problem in fig. 2.1. The Munsell Book of Color, pages of which are shown in the images, is an old color atlas, which was defined before fluorescent lights were common. The yellow pages of the atlas are known to exhibit noticeable distortions in the color gradient of the samples when viewed under fluorescent or LED light sources. If an RGB texture of an atlas page were used as input, a naive uplifting technique would have no chance of reproducing the fact that under one type of light source there is a smooth gradient, while under others the gradient breaks down.

We begin this chapter by reviewing the already existing approaches to spectral uplifting. We then talk about a novel technique, *constrained spectral uplifting*, which provides means for preserving the original metameric artifacts during uplifting and the implementation of which is the goal of this thesis.

Figure 2.1: Photos of four Munsell Book of Color pages (050B, 075Y, 100Y and 075R) in an xRite Judge QC viewing booth. Left: daylight; Right: fluorescent light. The photos are white balanced so that the neutral patches on the color checker match.

2.1 Uplifting methods

Although there have been multiple attempts at spectral uplifting, not many meet all the criteria required for a successful and complete conversion. Some of these methods pose various issues, such as outputting reflectance spectra with values outside the $[0, 1]$ range, working for saturated colors only, causing noticeable round-trip errors (i.e. there is a visible difference between the original RGB and the RGB of the uplifted spectrum), etc...

Following, we overview the most popular approaches to spectral uplifting and call attention to their advantages and disadvantages. We base most of this section on an article by Jung et al. [20], which, in addition to providing a survey of existing techniques, proposes a new one that is considered to be the current state of the art.

MacAdam. One of the first techniques was proposed in an article by MacAdam [25], which primarily focused on achieving the highest possible brightness for a given color saturation in printing. The uplifting process was only a byproduct of a proof of the limits to the colors' brightness, created especially for representing the reflectance of the researched colors (i.e colors of maximum brightness for any given saturation). Although this method is not limited to a specific input, it produces spectra that are box-shaped and only consist of rising and falling edges. This type of representation is unsuitable for reflectances usually found in nature, which tend to have smoother curves.

Smits. Another technique was proposed by Smits [41]. In this case, the uplifting is based on a box basis split into 10 discrete bins, which are derived using an optimization algorithm that accounts for energy conservation and aims for overall smoothness of the spectra. This approach is practically implemented and widely used, as it provides satisfactory results in the sRGB gamut [19]. However, in some cases, the uplifted spectra acquire values above 1, which does not satisfy the $[0, 1]$ range criterion. Furthermore, round trips consisting of uplifting an RGB value and converting the resulting spectrum back to RGB produce slight color differences, which are only amplified in scenes with multiple reflections.

Lastly, this approach becomes unstable when used for wider gamuts, as it was not designed for this purpose.

Meng et al. The goal of the method proposed by Meng et al. [28] is wide-gamut uplifting. It also concentrates on optimizing the uplifting algorithm for spectral smoothness. However, it does not take energy conservation into account, which results in images with colors that have no physical counterpart (i.e. no real material could produce such colors). Meng et al. [28] try to solve this by introducing a set of scaling methods for mapping the uplifted spectra to valid reflectances. These, however, fail upon uplifting bright colors.

Otsu et al. One of the most recent uplifting techniques has been proposed by Otsu et al. [33]. It is based on the observation that a typical measured reflectance spectrum can be represented with only a few principle components. The method uses clustered principal component analysis (PCA) and, unlike many other approaches, does not presume that spectra must necessarily be smooth. Such a simplification both eliminates the requirement of having a smoothness heuristic and enables the reconstructed spectra to match the actual measured spectra pretty well. This approach, however, has its downsides. Firstly, the method does not necessarily satisfy the $[0, 1]$ range criterion. Therefore, the values must be clamped, which results in color reproduction errors. Moreover, since there is no interpolation across clusters, similar RGB values may produce very different spectra, which might lead to discontinuities in rendering. However, in multiple cases, this method has been shown to outperform all of the already mentioned ones [19].

Jakob and Hanika. Jakob and Hanika [19] propose another approach, which discusses a parametric function space for efficient representation of spectral reflectance curves and its subsequent utilization in spectral uplifting. Based on their observation, a research software, *Borgtool*, has been created. Originally developed by collaboration of Charles University and Weta Digital, Borgtool possesses the capability of creating uplifting models in accordance to the work of Jakob and Hanika [19]. As the goal of this thesis is to extend Borgtool, we describe the theory behind its uplifting model in more detail.

The motivation was to create a spectral representation that would be both energy-conserving and would have a successful round-trip, e.g. the Delta E difference between the original RGB and the RGB obtained by conversion to spectra and back would be as small as possible. Based on the equation specifying the Delta E error, a simple analytical model has been created. Spectra in accordance with this model are represented as follows:

$$f(\lambda) = S(c_0\lambda^2 + c_1\lambda + c_2), \tag{2.1}$$

where $f(\lambda)$ is the resulting spectrum, S is a simple sigmoid function and c_i are coefficients of a second-order polynomial. Therefore, all spectra in this space are represented by three parameters.

In addition to energy conservation, the resulting spectra do not violate the $[0, 1]$ range constraint. They are smooth and simple, similarly to reflectance

Algorithm 1 Construction of a sigmoid-based uplift cube

create $RGBCube$ with empty RGB:spectra mappings
$unmappedPoints \leftarrow$ a list of all lattice points in $RGBCube$
$centerPoint \leftarrow$ index of the middle of $RGBCube$
▷ $RGBCube[centerPoint].rgb \simeq (0.5, 0.5, 0.5)$
$centerPoint.coefficients \leftarrow (0, 0, 0)$
▷ "guess" the coefficients at $centerPoint$
run the CERES optimizer for $RGBCube[centerPoint]$
remove $RGBCube[centerPoint]$ from $unmappedPoints$
while $unmappedPoints$ is not empty **do**
    for all $point \in unmappedPoints$ **do**
        if $point$ has a neighbor v with defined coefficients **then**
            $point.coefficients \leftarrow v.coefficients$
            run the CERES optimizer for $point$
            if optimization was successful **then**
                remove $point$ from $unmappedPoints$

spectra of real-life surfaces. Another advantage is memory efficiency, as storing one spectrum requires only three values.

However, representing spectra as such also has its drawbacks. For example, there is currently no straightforward, well-defined computation of the RGB $\rightarrow$ spectrum conversion in such a domain. Therefore, in order to find a coefficient triplet that evaluates to the desired RGB, an iterative optimization process is utilized.

The optimization is performed by the CERES solver [1], which requires only an initial "guess" of the coefficient triplet and a metric according to which it improves the guess (i.e. the Delta E error originating from round-trips). As it requires only a few iterations to converge to 0, the conversion of one RGB triplet is rather fast.

However, utilizing the CERES Solver during the rendering process would result in substantial performance overhead. Therefore, RGB:spectra mappings are pre-computed beforehand, and stored in a texture which then serves as a lookup structure during rendering.

Obviously, it is impossible to store mappings for every RGB triplet — the RGB space needs to be discretized as efficiently as possible. Borgtool approaches this problem by storing the mappings in the vertices of a regular 3D grid inside an RGB cube. Note that this is a simplification of the original proposition by Jakob and Hanika [19], which divides the sRGB space into three quadrilateral regions in which the coefficients are very smooth. For satisfactory results, only three 3D cubes of size 64^3 would be required.

We describe the pseudo-algorithm used in Borgtool in order to create a sigmoid-based uplifting model in algorithm 1.

Discretization of the RGB space does not, however, eliminate the need for uplifting RGB values that have no mapping in the spectral uplifting model. In such case, the unknown spectral reflectance values must be computed from the already existing mappings. As the RGB value we wish to uplift falls into a specific voxel in the RGB cube, without much elaboration, three straightforward methods

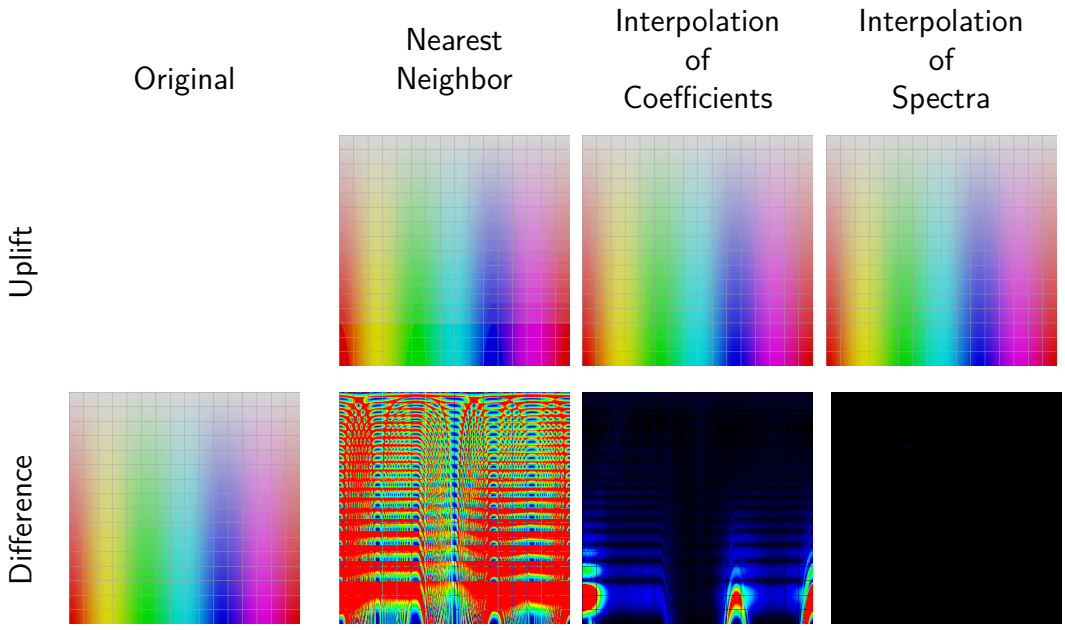


Figure 2.2: Comparison of techniques for obtaining spectra for non-mapped RGB triplets in the RGB cube as created by Borgtool. The Delta E in the difference images is relative to $\Delta E = 1$.

of how to uplift it come to mind:

1. trilinear interpolation of spectra in voxel corners
2. trilinear interpolation of coefficients in voxel corners
3. using the coefficients of the nearest voxel corner (*nearest neighbor* approach)

We present the results of these approaches when used for uplifting an RGB texture in fig. 2.2. We also compare them to the original texture and provide the difference images.

Although the original paper suggests that interpolating coefficients should, within limits, produce reasonable spectra without unexpected artifacts, we observe that the interpolation of spectra provides higher round trip accuracy. Therefore, the spectral uplifting tool in ART opts for interpolating spectra even despite the increased time complexity [32]. However, neither of the two approaches produce significant inaccuracies.

The nearest neighbor approach, on the other hand, is susceptible to visible artifacts, especially in the dark blue region. Therefore, it is not widely used.

An obvious case in which it is possible to avoid any kind of interpolation or optimization is when the RGB values that will be required during the uplift are known beforehand. They can then be added as lattice points to the RGB cube. This is especially useful when attempting to uplift specific textures or materials, which is, in fact, what this method was originally intended for.

This uplifting model is, in many cases, superior to the ones already mentioned above. First of all, the round trip error yields 0 in the sRGB gamut. In other gamuts within the spectral locus, it outperforms other models as well. Moreover, the execution speed is by far the best, even with the uplifting process running beforehand.

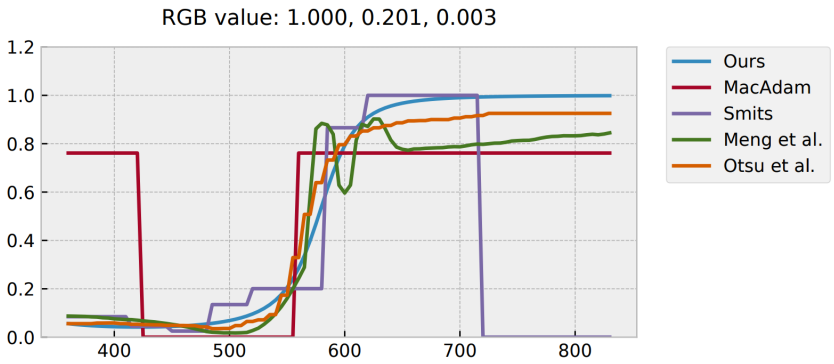


Figure 2.3: Comparison of spectral uplifting techniques in terms of reconstructed spectral shapes as shown by Jakob and Hanika [19]. Note that the "Ours" approach, in this case, refers to the approach by Jakob and Hanika [19].

The smoothness of the spectra can be considered both an advantage and a limitation. Although such spectra closely resemble real-life spectra and are suitable for the interpolation process, they cannot describe extremely bright and saturated spectra, as these tend to be more blocky.

Jung et al. The approach by Jung et al. [20] tries to solve this issue by extending the set of three sigmoid coefficients with three additional ones specifically designed for handling fluorescence. Its goal is to avoid creating blocky spectra at gamut boundaries and rather to create smooth spectra with added fluorescent dyes to compensate for the lack of saturation. The implementation of this method was added to Borgtool by Karlsruhe Institute of Technology. Similarly to the previous sigmoid-based technique, the uplifting model is an RGB cube created prior to the rendering process. In order to obtain the coefficients for the individual entries, CERES solver is utilized. As expected, the construction of the cube takes longer, as it has 6 coefficients to consider for each cube entry. However, the method is the first one capable of simulating fluorescent spectra, which is especially useful for wide-gamut input textures.

The problem arising with a requirement to uplift RGB values that do not have a mapping is solved in a different manner than in the previous approach. As both the nearest neighbor and interpolation of coefficients do not produce satisfactory results, *reradiation matrices* of the neighbor lattice points are used. Although this leads to higher memory requirements, the results are smoother and do not produce disruptive artifacts.

In this section, we have presented multiple techniques capable of spectral uplifting. None of them, however, propose a way in which to constrain the uplifting process to deliver specific spectral shapes, and they also cannot trivially be extended in this direction. This is mainly due to their spectral representations, which are simple and unable to reproduce all possible user-defined spectra. In order to demonstrate the type of shapes the individual techniques are capable of reproducing, we provide comparison of their results upon uplifting a specific RGB value in fig. 2.3.

2.2 Constrained spectral uplifting

Achieving identity of uplifted spectra to the real-world spectra is, obviously, impossible. However, uplifting many RGB-based assets does not require us to be able to uplift the whole RGB gamut, but only the spectra used for the creation of said assets. As it is pretty common for artists in the VFX modeling industry to use specific color atlases when designing textures and materials, the ability to *constrain* the uplifting system with these atlas colors would be extremely useful.

In other words, the user would define specific RGB:spectra mappings which would later be used in order to uplift certain RGB triplets. RGB values that would not have a pre-defined mapping would be uplifted by slightly altering the curves of their already-mapped neighbors, in order to prevent color inconsistencies under different illuminating conditions.

We call this process *constrained spectral uplifting*. The fact that it does not provide as much freedom as other spectral uplifting approaches works for our benefit, as the results are not a subject to high metamerism, which is, after all, the goal of this thesis.

In this thesis, we base the algorithm used to implement constrained spectral uplifting on algorithm 1. We use a similar RGB cube as a structure for saving the mappings, and we also create an uplifting model prior to rendering. We leave the specific details of implementation to chapter 3. In this section, we discuss the theoretical background of the constraining itself. Specifically, we focus on the means of storing the individual spectra in our structure and the problems that arise along with it.

2.2.1 Spectral sampling

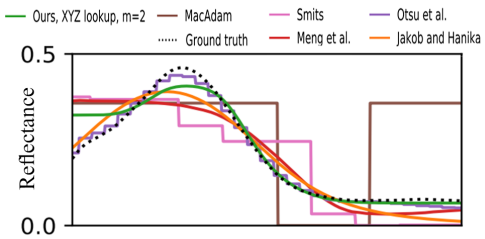
The constraining process starts when the user inserts a set of spectra. Finding an RGB match and creating a mapping is a straightforward task — one must simply convert the spectra to RGB. However, the spectra must then be stored in the structure, which requires its discretization.

A trivial way to store spectral information is via regular sampling, i.e. to store every ith value of the spectrum, starting from roughly 380nm and ending at 780nm (as that is the range for visible light). However, for an acceptable color reproduction, around 30 samples are needed for each spectrum [35]. Storing so many values is extremely memory inefficient, especially when taking into account all the other mappings that must be created later in the system.

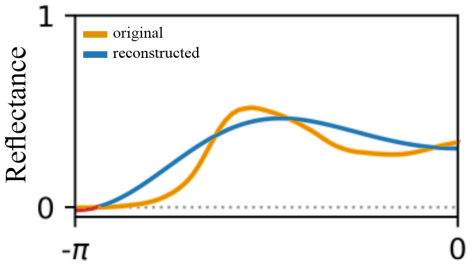
Another issue with such storage arises during the optimization stage of the uplifting process. Trying to modify so many values is infeasible for any optimizer. We must therefore create a more compact representation, which recreates the spectra as accurately as possible.

As already mentioned, utilizing the representations of one of the existing approaches to spectral uplifting (reviewed in section 2.1) is not an option, as all of them are restricted to a specific spectral space and therefore unable to recreate all the possible reflectance curves. We provide an example of this in fig. 2.4a, where the resulting spectra of the uplifting process of multiple methods are compared to the actual measured spectrum of the real-life material.

Therefore, discretization of spectra must be approached in a different way.



(a) Reconstruction with uplifting models as plotted by Peters et al. [34], where *Ours* represents the moment-based technique



(b) Reconstruction with truncated Fourier series [35]

Figure 2.4: Comparison of the accuracy of multiple techniques when attempting to reconstruct a real-life measured spectrum

The simple and smooth shape of the spectra indicate that using a lower-dimensional linear function space, such as the Fourier series, could be the key to their storage. Techniques based on this observation have been studied for the storage of emission spectra [37], and appear promising also for reflectance spectra. This method is studied in an article and subsequent presentation by Peters et al. [35]. As the reflectance spectra are aperiodic, it is reasonable for the Fourier basis to consist of cosine transforms only. Eventually, a truncated Fourier series is used for the reconstruction, which is computed according to the following equation:

$$f = \sum_{i=0}^{m} c_j cos(j\varphi) \tag{2.2}$$

where c_j are the Fourier coefficients eventually stored in the lattice points of the RGB cube.

We show an example of a result obtained by this method in fig. 2.4b. Although the reconstruction is not far off, the resulting spectra do not always have a physical counterpart, as the reconstruction does not obey the $[0, 1]$ range constrain. We can observe this drawback even in fig. 2.4b.

In contrast to a linear function space, spectra can also be represented non-linearly. These representations are, however, incompatible with linear prefiltering of textures [34].

To eliminate the shortcomings but utilize the strengths of both linear and non-linear methods, Peters et al. [34] propose a novel approach to spectral representation. The representation is based on the theory of moments (i.e. it is non-linear), but consists of Fourier coefficients, which implies compatibility with linear filtering. Additionally, it aims for the satisfaction of the $[0, 1]$ range constraint.

Following, we provide a brief overview of both the algorithm for obtaining coefficients and the reconstruction process. For more details, we refer the interested reader to the original article [34].

Obtaining coefficients The first problem with obtaining the coefficients is caused by the shape of the spectra. In contrast to the Fourier basis, they are

aperiodic. Their storage with Fourier coefficients therefore requires their conversion to a periodic signal.

A simple solution would be to linearly map wavelengths to a 2π-periodic signal. Although such an approach can be used (we discuss its performance in section 4.1.1), it causes distortions and strong artifacts at the boundaries. Moreover, Fourier coefficients computed for such signal are complex instead of real, which requires almost twice the memory for storage.

Therefore, an improvement to the mapping, called *mirroring*, is proposed. It first maps the negative values of the signal as in the following equation:

$$\varphi = \pi \frac{\lambda - \lambda_{min}}{\lambda_{max} - \lambda_{min}} - \pi \in [-\pi, 0] \tag{2.3}$$

The mapping for the positive part is then defined as the negative part mirrored, i.e. $g(\varphi) = g(-\varphi)$ for all $\varphi \in [0, \pi]$, which results in smooth transitions at the boundaries. The Fourier coefficients are then computed for only this mirrored signal as follows:

$$c = 2\Re \int_{-\pi}^{0} g(\varphi)\mathbf{c}(\varphi)d\varphi,$$

where $\mathbf{c}(\varphi)$ is the Fourier basis.

The reconstruction is also obtained only for the mirrored part of the signal. Although this might seem wasteful, the signal created by this approach is even and therefore requires only real Fourier coefficients for its representation, which benefits the storage requirements.

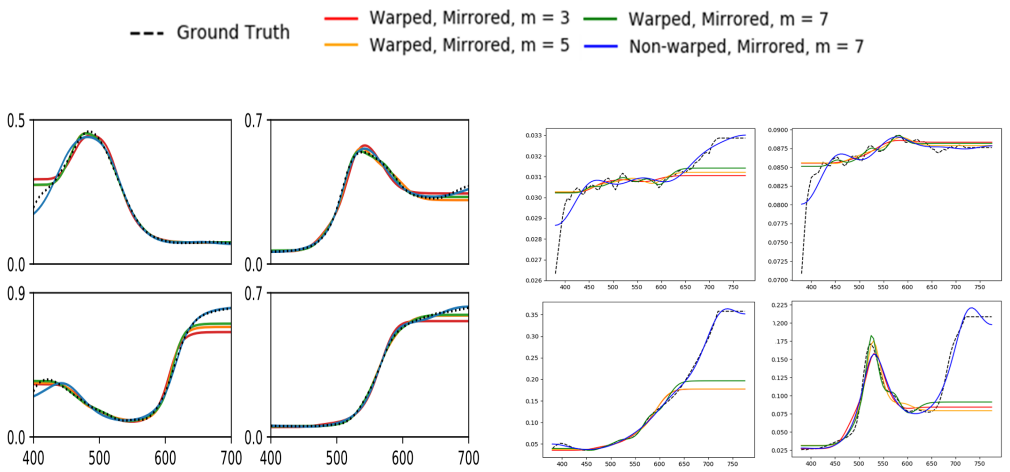
Another proposed improvement to obtaining the coefficients is by focusing accuracy on important regions of the curve in terms of color reconstruction (i.e. around 550nm), also called *warping*. This is achieved by means of a differentiable, bijective function that maps the wavelength range to the $[-\pi, 0]$ range and is used as a weighting function when computing coefficients. Warping of the signal is useful especially when using only a small number of coefficients, as they are unable to capture more complex curves, and it is highly recommended to always warp the signal if using less than 5 moments.

In our implementation of the constrained spectral uplifting, we choose to mirror, but not warp the signal. We explain this decision in section 4.1.1, where we also provide results of experiments that justify our choice.

Note that using m complex Fourier coefficients for storing a spectrum implies that $m + 1$ coefficients are actually saved. The $+1$ factor stands for the zeroth moment c_0, which is real in both the mirrored and the non-mirrored case. Therefore, overall, mirroring requires storing $m + 1$ scalars, while non-mirroring requires $2m + 1$ scalars.

Reconstruction The default Fourier coefficients (without improvements such as mirroring and warping) are stored for a 2π-periodic signal $d(\varphi)$, where $d(\varphi) \geq 0$ is a density for all phases $\varphi \in \mathbb{R}$. Therefore, they satisfy the definition of trigonometric coefficients for the *trigonometric moment problem* [24]. Specifically, the coefficients γ can be expressed as

$$\gamma = \int_{-\pi}^{\pi} d(\varphi)\mathbf{c}(\varphi)d_{\varphi} \in C^{m+1}, \tag{2.4}$$



(a) Examples of reconstruction of smooth spectra as provided by Peters et al. [34]

(b) Reconstruction of spectra of the Macbeth chart as plotted by us

Figure 2.5: Examples of reconstruction with the trigonometric moment method

where $d(\varphi)$ is the finite measure that they represent, and $\mathbf{c}(\varphi)$ is the Fourier basis.

By building upon this observation, the reconstruction of spectra is based on the theory of moments, specifically on Maximum Entropy Spectral Estimate (MESE) [6]. The MESE is capable of reconstructing both smooth and spiky spectra, and has been shown to produce impressive results when used for the reconstruction of emission spectra.

However, the problem with this approach is that it is not bounded, i.e. not suitable for reflectance spectra. Therefore, a novel, *bounded MESE*, has been introduced. It is based on the research by Markoff [26] and, subsequently, Kreĭ [23], who developed a duality between bounded and unbounded moment problems formulated in terms of Herglotz transform. This duality is used for transforming trigonometric moments to *exponential moments* so that the bounded problem represented by the trigonometric moments has a solution if and only if the dual unbounded problem represented by the exponential moments has a solution.

The summary of the reconstruction process is as follows:

1. compute exponential moments from the trigonometric moments
2. evaluate unbounded MESE for the exponential moments
3. compute bounded MESE by applying duality to the unbounded MESE

We present examples of results obtained by the storage and subsequent reconstruction of reflectance spectra with the trigonometric moment method in fig. 2.5, where we also compare the performance of different number of moments and different signal mapping techniques. For smooth spectra, even a small number of moments ($m = 3$) provide a quite accurate reconstruction (see fig. 2.5a). However, more complex spectra with sharper edges and spikes tend to lose detail and their reconstruction is a lot smoother. As seen in fig. 2.5b, especially for the "black" and "neutral35" patches, even 7 moments are not capable of describing the original curve accurately.

Emission spectra perform even worse. To give an example, even a mirrored approach with $m = 31$ is unable to reconstruct the most spiky details [34]. However, as the focus of this thesis is purely on reflectance spectra, we do not encounter a spectrum that would require such high number of moments.

The optimal number of moments for storing a spectral reflectance curve depends on many factors, such as the shape of the curve, memory limitations and the accuracy for which we aim, either in terms of curve similarity or color error under various illuminants. We discuss these factors more thoroughly as we progress with this thesis, and explain our method for determining the sufficient moment count in section 4.1.2.

Due to memory efficiency, smoother gradients, higher precision and multiple other reasons, thoroughly explained as we progress with this thesis, we decide to use variable number of moments per an input reflectance spectrum in our implementation. Therefore, although Peters et al. [34] state that the interpolation of coefficients provides satisfactory results, our uplifting process cannot rely on this method for uplifting non-mapped RGB values.

3. Implementation

We approach the problem of spectral uplifting similarly to Jakob and Hanika [19], where an uplifting model is created prior to rendering. Our implementation therefore consists of two parts — *model creation* and its subsequent *utilization* in a rendering software.

For the first part, we extend an already existing uplifting tool, Borgtool, which is currently used for creating sigmoid-based RGB cubes in a way as described in algorithm 1. We add the possibility for creating trigonometric moment-based cubes (from now on referred to as *trigonometric moment cube*), i.e. for the spectra to be stored with trigonometric moments rather than sigmoid coefficients. We also add an option for constraining such a cube with a user-specified constraint set (e.g. a color atlas).

We then describe the theory behind the integration of this model into a rendering software. In practice, we add its support into ART, which, up until now (version 2.0.3), has used only one built-in sigmoid-based cube for uplifting purposes.

3.1 Uplifting model

As we base most of our implementation on the already existing sigmoid-based approach, we start this section with the detailed description of its model, i.e. the sigmoid-based cube. We then describe our trigonometric moment cube, which can be viewed as its extension.

The sigmoid-based cube structure contains multiple entries in form of evenly spaced lattice points. Following, we name the main parameters of a single cube entry:

- *target RGB* — the coordinates of the point in the RGB cube
- *coefficients* — 3 sigmoid coefficients used to reconstruct a spectrum so it matches the target RGB
- *lattice RGB* — the actual RGB that the reconstructed spectrum evaluates to. Ideally, this should match the target RGB

Along with its entries, the resulting cube structure also stores a few other properties, both *static*, such as the illuminant according to which the RGB cube is uplifted, and *user-adjustable*, such as the cube dimension or the fitting threshold (i.e. the maximum allowed difference between the target and the lattice RGB).

Our trigonometric moment cube includes most of these parameters, and mainly extends the ones that are not suitable for the moment representation. The main difference between the two cubes lies in the distinction of the lattice points — while the sigmoid cube regards all of its points as equal, the trigonometric moment cube distinguishes (by means of a `seeded` boolean parameter per entry) between *seeded points*, i.e. the lattice points that store the user-inputted RGB:spectra mappings; and *regular points*.

Our requirements for the shape of the spectra at seeded points differ from the ones at regular points. While we prefer the regular points to have their spectra as

smooth as possible in order to avoid unexpected artifacts under other illuminants, the coefficients of the seeded points must reconstruct spectra almost identical to the input spectra, which might include sharp edges and spikes.

Therefore, it is sufficient for the regular points to be represented with a smaller number of coefficients, while seeded points might require a lot more. Although a smooth spectrum can be represented with a high number of coefficients, such a representation is memory inefficient, its reconstruction is more time consuming, and, most importantly, it does not work well with the optimizer. Based on our experiments in section 4.1.2, we decide to store the spectra of regular points with 3 coefficients and adjust the number of coefficients of the seeded points depending on the nature of its desired spectral shape.

Besides supporting variable number of coefficients (ranging from 3 to 21), the trigonometric moment cube also supports the possibility of having multiple coefficient representations per lattice point. The sole purpose of this extension is to lower the cube size requirements upon constraining, which we discuss later in section 3.1.2.

In addition to the cube structure, the construction of the trigonometric moment cube is also similar to the one of the sigmoid cube, which follows algorithm 1. Its main distinctions are in the constraining process (which the sigmoid cube lacks) and in the first round of optimizing, or, as we refer to from now on, *fitting*.

Following, we name the individual steps of the process, which we then describe in greater detail.

1. *Initialization*
2. *Cube seeding (optional)*
3. *Fitting of starting points*
4. *Cube fitting*
5. *Cube improvement*
6. *Cube storage*

3.1.1 Initialization

This part of the run is responsible for the following:

- parsing of the parameters
- initialization of the cube and its entries with default values
- loading of the required constraint sets

The initialization of the cube is pretty straightforward, as all of its properties are either user-defined or set to default (note that the default illuminant is always D65). The number of cube entries is directly proportional to the cube's `dimension` parameter, which specifies the number of entries per one axis. This

```
Entry ID:   orange
-------------------------------------------------------------------------------
Description            :  "orange" patch of the Macbeth colour checker
Type                   :  reflectance spectrum
Fluorescence data      :  no
Measurement device     :
Measured by            :
Measurement date       :

Sampling information
--------------------
Type                   :  regular
Start                  :  380.0 nm
Increment              :  5.0 nm
Maximum sample value   :  100.0

ASCII sample data
-----------------
{6.143748,  5.192119,  4.867970,  5.092529,  4.717562,  4.663087,
4.455331,  4.562958,  4.517197,  4.536289,
4.454180,  4.543101,  4.491708, ... }
```

Figure 3.1: A sample entry from the Macbeth Color Checker atlas

renders the total number of entries to $dimension^3$. As the lattice points are positioned evenly, their target RGB values are equivalent to their coordinates in the RGB cube.

The constraint sets are inputted in a form of a simple .txt file, which contains a list of entries in a textual form as shown in fig. 3.1. In order to avoid high memory requirements arising with large input datasets, we do not store the spectral data directly, but take advantage of the trigonometric moments.

We store the spectral curves of the individual constraints with Fourier coefficients as described in section 2.2.1. We mirror but do not warp the signal prior to coefficient computation (see section 4.1.1).

The number of coefficients per constraint variable, ranging from 4 to 21. We explain our method for determining the sufficiency of coefficient representation, and therefore the coefficient count for each constraint, in section 4.1.2.

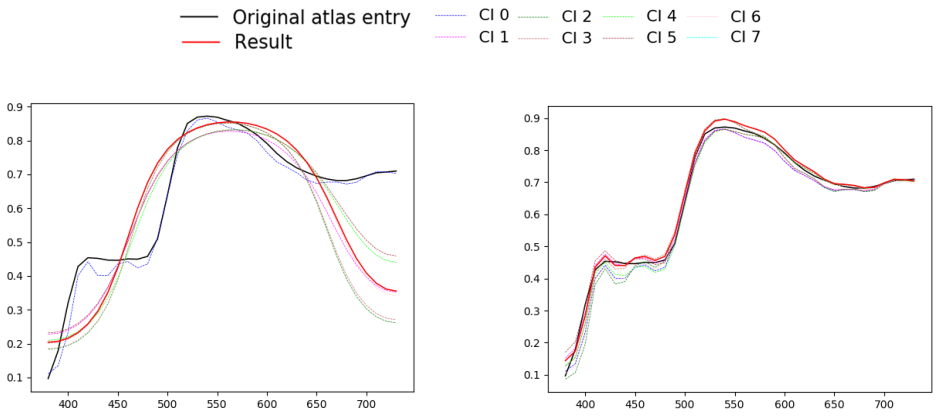
Note that we require the constraints to have identical sampling information. This representation is used internally throughout both the fitting and the uplifting process for e.g. spectral reconstruction for the purposes of color conversions.

3.1.2 Cube seeding

In order to uplift the whole cube as described in algorithm 1, we must first fit one or more *starting points* whose coefficients can then be used as an initial guess (called *prior*) for the fitting of other lattice points. For this purpose, we utilize the user-specified constraint set. The general idea behind this process is to copy the coefficients of constraints to specific lattice points, and then use these coefficients as prior for fitting said lattice points. We refer to this process as *seeding* of the cube, and term the constraints assigned to these lattice points their *seeds*.

The ideal scenario would be if the RGB values of the constraints were to perfectly match the coordinates of the lattice points. However, as the constraints can evaluate to virtually any triplet within the $[0, 1]$ range, it is most likely that they would correspond to points inside the cube's voxels.

An approach that first comes to mind would be to create a complete injective



(a) Only 1 voxel corner seeded

(b) All 8 voxel corners seeded

Figure 3.2: Uplifting results according to the seeding method

mapping (one constraint seeds only one voxel corner) between the constraints and their closest lattice points. However, as we employ all 8 voxel corners during the uplifting of non-mapped RGB values (because we disregard the nearest-neighbor approach due to its inaccurate results, see fig. 2.2), this method is insufficient — seeding only one of the 8 points might cause the spectral curves of the other 7 to be considerable distinct from the original constraint. This is mainly due to our choice of coefficient count for non-mapped entries, which is a lot lower than for the seeded entries (see thorough explanation in section 4.1.2), i.e. the curves of the non-seeded entries cannot be as precise.

We can observe this behavior in fig. 3.2a, where we provide the curves reconstructed at the 8 corners of a voxel and compare their trilinear interpolation to the original constraint. The original constraint significantly differs from the uplift, which may cause color artifacts under different illuminating conditions. However, as seen in fig. 3.2b, propagating the information about the original reflectance of the constraint to all voxel corners improves the result remarkably. We therefore opt for seeding all 8 voxel corners per constraint.

During the seeding process, it may occur that two constraints would fall into neighboring voxels, i.e. that they would share some of the voxel corners. In order to utilize both of these constraints as seeds, we support the possibility of one lattice point having multiple coefficient representations. In addition to coefficients and their count, we also store an entry ID per each representation, so as to later distinguish the reconstructed curves during rendering and decide which to employ (explained in more detail in section 3.2).

If two constraints fall into the same voxel, however, there is no way of determining the interpolation of which the user desires upon uplifting the RGB values inside said voxel. We therefore discard one of the constraints and throw an error informing the user of the collision and suggesting the increase of the cube `dimension` parameter.

We show an example of a cube seeded and subsequently fitted with the Munsell Book of Color used as a constraints set in fig. 3.3. Lattice points marked as black represent the seeded points. Note that a lot of these points store multiple coefficient representations.

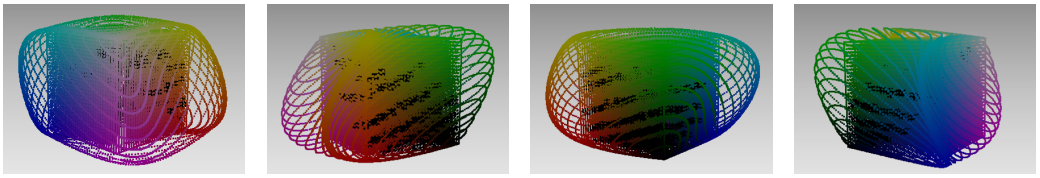


Figure 3.3: A 32-dimensional cube fitted with the Munsell Book of Color. Note that, due to the low dimension of the cube, multiple collisions occurred, i.e. not all constraints have been utilized.

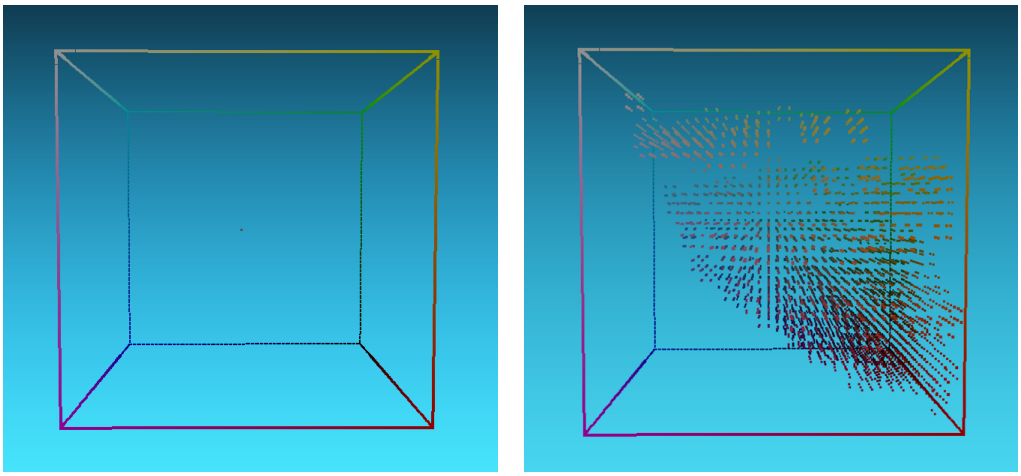


Figure 3.4: Comparison of the position of starting points when fitting from the middle with sigmoids (left) and when seeding with the Munsell Book of Color (right)

Constraining the uplifting process is optional. If no constraint set is inputted, all lattice points are regarded as regular points and the cube is fitted from the middle in the same manner as the sigmoid cube. The resulting uplifting structure provides no advantages over the sigmoid cube, other than having slightly different spectral shapes.

Supporting this option requires us to specify 3 prior coefficients for the center point (i.e. the point corresponding to an RGB of $(0.5, 0.5, 0.5)$). By storing spectral curves that roughly evaluate to such RGB with the trigonometric moments, we observe that the values of the coefficients are approximately $\{0.5, 0, 0\}$. Therefore, we use them as prior.

We provide a comparison between the starting points' placement when seeding from the middle (either with our or the sigmoid method) and when seeding with a constraint set in form of the Munsell Book of Color in fig. 3.4.

3.1.3 Fitting of starting points

By seeding the cube, we have appointed coefficients to some of the lattice points. These coefficients reconstruct a spectrum that evaluates to an RGB value which we denote as the *lattice RGB*. The difference between the lattice and the target RGB is therefore equal to the distance between the lattice point and its assigned

constraint in the cube. It is apparent that the distance may be higher than the defined fitting threshold. In such cases, we must "improve" upon the coefficients so that the resulting color difference is as low as possible.

Our problem of improving the coefficients satisfies the definition of the *Non-linear Least Squares* problem [14]. Non-linear Least Squares is an unconstrained minimization problem in the following form:

$$\underset{x}{\text{minimize}} \; f(x) = \sum_i f_i(x)^2, \tag{3.1}$$

where $x = \{x_0, x_1, x_2, ...\}$ is a parameter block that we are trying to improve (i.e. our coefficients) and f_i are so-called *cost functions*. The definition of cost functions is dependent solely on the current problem. In our case, we primarily require to minimize the difference between the lattice and the target RGB. Our secondary requirement is for the shapes of the curves of the original constraint and the of the seeded point to be similar. This gives rise to multiple choices for cost functions, such as using the difference between curves along with the Delta E error, using one or multiple cost functions for the RGB error etc... After implementing some of them and testing their performance, the results of which we provide in section 4.1.3, we decided on using four cost functions — three for specifying the absolute difference in one of the three axes of the cube, and one for specifying the average distance between curves per wavelength sample. We also include a heuristic which sets the value of the fourth cost function to 0 if the curve distance falls below a certain threshold, and iteratively increases the threshold if the optimizer fails. The reasoning behind this is also explained in section 4.1.3.

To solve an optimization problem defined in the way as described above, we use, similarly to Jakob and Hanika [19] and Jung et al. [20], CERES solver.

CERES solver

As already mentioned in section 2.1, CERES solver is an open-source library for solving large optimization problems such as the Non-linear Least Squares problem. It consists of two parts — a *modeling API*, which provides tools for the construction of optimization problems, allowing us to set parameters such as maximum number of iterations of the optimizer or maximum number of consecutive nonmononotic steps; and a *solver API* that controls the minimization algorithm.

To solve a Non-linear Least Squares problem, the solver requires us to specify only a so-called *residual block*, which is a structure defined by the prior coefficients and the cost functions. During the execution, the solver attempts to minimize the values of the cost functions (or *residuals*) in the residual block. The execution is aborted and the current best parameter block returned when the solver achieves either the specified number of iterations or nonmonotonic steps. For more information on the specifics of CERES solver, we refer the interested reader to its documentation by Agarwal et al. [2].

There is one main downside to using CERES solver. As it was designed to handle very large, sparse problems where every residual term depends on only a few of the input parameters, it is not ideal for solving problems with only one large parameter block, i.e. it might get stuck in local minima and therefore produce

Algorithm 2 Fitting of one coefficient representation of a *point* from seeded points

```
1: threshold ← 0.001
2: while threshold < 1 do
3:     coefsToFit ← either the first 4 coefficients or all of them depending on the coefficient count of point
4:     i ← 0
5:     while optimizer is unsuccessful and i < maxIterations do
6:         heuristically change the first coefficient of coefsToFit
7:         run the optimizer with parameters coefsToFit and threshold set to threshold
8:         i++
9:     if optimizer was successful then
10:        point.coefs ← coefsToFit
11:        break
12:    increase threshold
```

unsatisfactory results. Unfortunately, if a seeded point is represented with a high number of coefficients, our optimization problem falls into this category.

We solve such problematic cases by applying a simple heuristic, which consists of slightly altering the first coefficient (as it has the highest influence on the shape of the curve) and running the optimizer again. However, although such an optimization greatly improves the overall performance of fitting, it remains insufficient for too high a number of coefficients, i.e. the threshold for the fourth residual must be increased to values extremely high and, by then, it loses resemblance to the original shape.

Therefore, we implement another heuristic improvement — if the coefficient count is higher than 14, we let the optimizer optimize only the first 4 coefficients while leaving the others constant. We use the threshold of $c > 14$ as that is roughly the boundary where the fitted curves begin to show undesired artifacts, and we optimize the first 4 coefficients because their number is both low enough for the optimizer to handle without errors, and high enough so we give the fitting process enough degrees of freedom.

We do not provide the results of the experiments that lead to our decision of the heuristic due to its trivial nature.

We summarize the fitting process of the starting points, including the utilization of threshold for our fourth cost function, in algorithm 2. If the optimizer is unable to fit a seeded point, we throw a warning and convert it into a regular point.

3.1.4 Cube fitting

As the seeded points are represented with a higher number of coefficients and may even contain multiple coefficient representations, they cannot be directly used as prior guesses for the regular points. First, they must be "converted" into a lower-dimensional representation.

We refer to the conversion process as *coefficient recalculation*. It consists of reconstructing the reflectance spectrum of the seeded point and subsequently

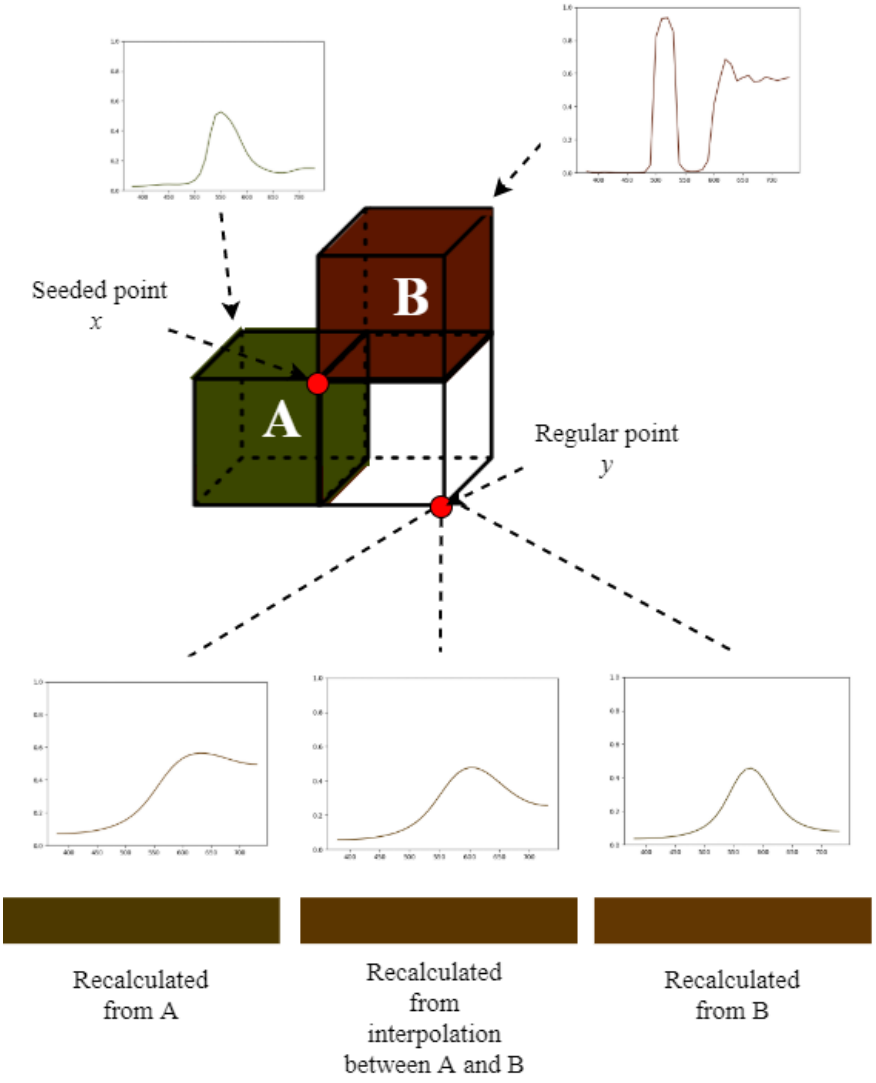
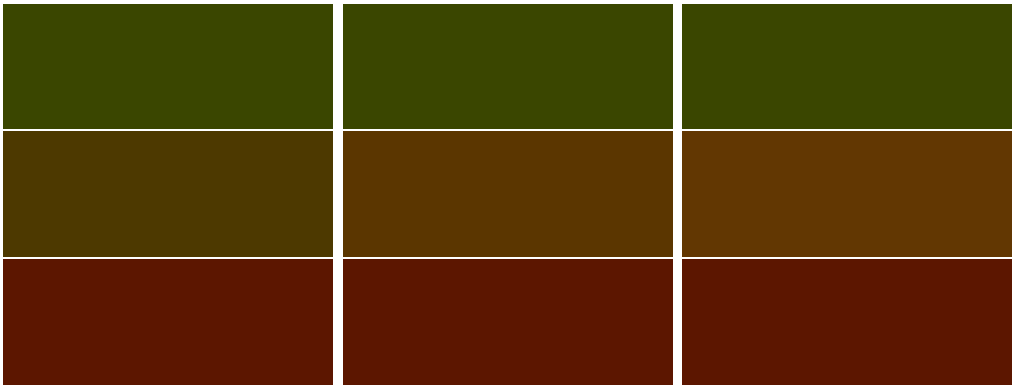


Figure 3.5: An illustrative demonstration of a problem posed by the coefficient recalculation of a seeded point containing multiple metameric spectra (stored in the form of moment representations)

saving it with 3 coefficients. Although this process causes significant loss of spectral information, it preserves the rough outline of the curve. This works to our benefit — it reduces the likelihood of significant color artifacts between the seeded points and regular points while keeping the spectra smooth.

A problem arises if the seeded point that is being recalculated for the purposes of fitting a regular point contains multiple moment representations of distinctly shaped spectral curves (note that, for the purposes of this thesis, we call a set of similarly shaped spectral curves a *metameric family*). We illustrate this situation in fig. 3.5. The recalculated seeded point (denoted x) contains the moment representation of both the constraint A and B. As these have vastly distinct shapes, it is natural that they evaluate to completely different colors under error-prone illuminants (specifically, the colors shown in the image are under FL11). Choosing to recalculate only the representation of A in order to obtain the prior coefficients of the regular point (denoted y) would result in color artifacts between y and the voxel seeded with B. Symmetrically, the same applies to choosing to



(a) Recalculation of the coefficient representation of A only

(b) Recalculation of the interpolation between A and B

(c) Recalculation of the coefficient representation of B only

Figure 3.6: Color gradients resulting from our experiment in fig. 3.5. For each figure, the upper and the lower patch stand for the colors of coefficient representations of A and B respectively, while the middle patch stands for the color achieved from the current recalculation technique.

recalculate only the coefficient representation of B. We can also observe this in fig. 3.6, where we present the comparison of the color gradients created by respectively using the individual recalculation techniques. Note that the spectra (and, subsequently, colors) that the A and B entries evaluate to in both fig. 3.5 and fig. 3.6 are not the original spectra of the constraints, but the spectra that the coefficient representations saved at the seeded point x evaluate to. Also note that the spectra and colors corresponding to the point y are the recalculation results, not the final fitting results.

As it is our intention to keep the color transitions within all voxel pairs smooth, we propose the interpolation of spectra reconstructed from the moment representations.

Interpolation of metamers

In the following, we show that the linear combination of two spectra that are metameric under a given light source results in another metameric spectrum. To our best knowledge, this insight, while not particularly mathematically complex, has not been explicitly stated in graphics literature before.

Let us assume the spectral power distributions of two metamers saved at a lattice point, $P_1(\lambda)$ and $P_2(\lambda)$, that satisfy the conditions

$$\begin{aligned} \int P_1(\lambda)\overline{r}(\lambda)d\lambda &= \int P_2(\lambda)\overline{r}(\lambda)d\lambda \\ \int P_1(\lambda)\overline{g}(\lambda)d\lambda &= \int P_2(\lambda)\overline{g}(\lambda)d\lambda \\ \int P_1(\lambda)\overline{b}(\lambda)d\lambda &= \int P_2(\lambda)\overline{b}(\lambda)d\lambda \end{aligned} \tag{3.2}$$

where $\overline{r}(\lambda)$, $\overline{g}(\lambda)$ and $\overline{b}(\lambda)$ are the RGB color matching functions.

Let us express the R component of the RGB value resulting from the linear

combination of $P_1(\lambda)$ and $P_2(\lambda)$ as follows:

$$R = \int a \cdot P_1(\lambda)\overline{r}(\lambda)d\lambda + b \cdot P_2(\lambda)\overline{r}(\lambda)d\lambda, \quad \text{where } a + b = 1 \tag{3.3}$$

By rewriting this expression and utilizing the equality from eq. (3.2), we get

$$R = a \cdot \int P_1(\lambda)\overline{r}(\lambda)d\lambda + (1 - a) \cdot \int P_1(\lambda)\overline{r}(\lambda)d\lambda,$$

So

$$R = \int P_1(\lambda)\overline{r}(\lambda)d\lambda$$

The same proof can be equivalently applied to the G and B components of the resulting RGB value. Therefore, we conclude that the resulting spectral distribution is also a metamer.

We use this observation in order to achieve smoother color transitions between distinct metameric families by interpolating between metameric spectra stored (in the form of moment representations) at lattice points that contain multiple coefficient representations.

Additionally, we use it to obtain the lattice RGB of such points — i.e. we store the RGB of the interpolated spectrum. Although this information is meaningless for the purposes of further fitting, it gives us an approximation of how well the individual points are fitted.

Other than the coefficient recalculation, our fitting process is carried out in a manner similar to that of the sigmoid fitting (see algorithm 1), where the lattice points are fitted in multiple *fitting rounds*, each round attempting to fit the neighbors of the already fitted points. We provide a more detailed description of the principle behind the fitting algorithm used in our implementation in algorithm 3.

3.1.5 Cube improvement

In extreme cases, such as when using a low fitting threshold or a sparsely-sampled constraint set, the fitting of some points may be unsuccessful. We assign most of these failures to the shortcomings of the ART color conversion library, since its functions are employed in the Borgtool.

Specifically, the conversion of an equal energy reflectance spectrum to RGB under the D65 illuminant does not produce the expected $RGB = (255, 255, 255)$, but rather an RGB of $(254.95, 255.005, 255.0003)$ for a 1nm sample increment, and, even worse $(254.88, 255.07, 254.93)$ for a 10nm increment. To force ART to reproduce an RGB of $(255, 255, 255)$, a spectrum with slightly lower values in the area around 550nm is required. Therefore, although a coefficient set $c = 1, 0, 0$ represents an equal energy spectrum, it does not suffice for the optimizer. Additionally, 3 trigonometric coefficients are not capable of representing the slightly modified spectrum which ART regards as equal energy spectrum without slight error. This becomes even more visible if the amount of samples used for the internal representation of spectra is low, as it gives less freedom to the optimizer. Furthermore, this problem may also arise for target RGB values extremely close to $(255, 255, 255)$ (and not only the lattice point with RGB=$(255, 255, 255)$), which is mainly the case in higher-resolution cubes.

Algorithm 3 Fitting of the cube from starting points

```
1: fittingRound ← 0
2: unfittedPoints ← a list of all points in RGBCube \ startingPoints
3: for all point ∈ unfittedPoints do
4:     point.fittingDistance = MAX_DOUBLE
5: while unfittedPoints is not empty do
6:     currRoundPts ← points from unfittedPoints that have at least one fitted neighbor
7:     for all point ∈ currRoundPts do
8:         for all fittedNeigbor ∈ fitted neighbors of point do
9:             if fittedNeighbor ∈ seededPoint then
10:                point.coefs ← recalculateCoefs(fittedNeighbor.coefs)
11:            else
12:                point.coefs ← fittedNeighbor.coefs
13:            [sDist, sCoefs] ← CERES.Solve(point.coefs, costFunctions)
14:            if sDist ≤ point.fittingDistance then
15:                point.fittingDistance ← sDist
16:                point.coefs ← sCoefs
17:            if cDist ≤ fittingThreshold then
18:                point.treated = true
19:                break
20:        if point.fittingDistance > fittingThreshold or point has tried the coefficients of all of its neighbors then
21:            remove point from unfittedPoints
22:    fittingRound ← fittingRound + 1
```

To minimize the created errors, we add a heuristic-based improvement of the coefficients, which sets their values to ones we assume are closest to the optimum and then proceeds similarly to the coefficient improvement of seeded entries. For a pre-defined amount of times, it slightly changes up the coefficients and runs the optimizer again, terminating if successful. However, neither this, nor any other heuristic-based improvement approach we attempted to implement, were capable of completely eliminating failures. Their only asset was a slightly lowered fitting threshold in some of the cases.

However, we note that these shortcomings are extremely rare and do not visibly lower the accuracy of the uplifting model.

3.1.6 Cube storage

Once the cube is fitted, its contents are written to a binary file. As we want the resulting file to be as small as possible, we save only the information crucial for the purposes of rendering. Following, we provide a list of contents of a cube file:

- *version*
- *moment flag* — a flag signifying that the cube is based on trigonometric moments. We extend the sigmoid cube structure in a similar manner with a sigmoid flag for an easier cube recognition in a rendering software.

- *dimension* — the number of lattice points per axis
- *illuminant* under which the cube was fitted
- *fitting threshold*
- *spectral range* — the sampling information for internal representation of spectra
- for every point, we store:
  - *coefficient representations*, along with their *entry IDs* and their *sizes*
  - *lattice RGB*
  - *fitting distance* — the distance between lattice and target RGB

Note that storing the target RGB of lattice points is unnecessary, as it can computed from the cube's dimension parameter. Although we could similarly compute the lattice RGB from the coefficient representations, we store it in order for the moment cube structure to remain compatible with the sigmoid cube structure.

In addition to storing the cube, we extend Borgtool with the functionality to load such a cube and utilize it for either the purposes of rainbow texture uplifting (specifically, uplifting of the texture in fig. 2.2) or for its 3D visualization.

3.2 Renderer integration

In order to demonstrate the proper utilization and, subsequently, the performance of the trigonometric moment cube, we integrate it into an existing renderer — specifically, ART. As ART already has the support for uplifting with the sigmoid cube, we solely extend both its cube structure and uplifting capabilities in a similar manner as in the Borgtool. For the scene description files, we add an option for specifying the cube file to be used for uplifting — due to our extension in terms of the *flag* parameter, we are capable of recognizing the type of the cube without it being manually specified by the user.

The uplifting process itself must be, as already concluded in this thesis, based on the trilinear interpolation of spectra at the corners of the voxel that the desired RGB falls into. Therefore, we proceed as follows:

Firstly, from the notion of the desired RGB, we obtain the 8 voxel corners along with their distances to the RGB triplet. These will later be used as weights for interpolation. We then examine the sets of entry IDs at the voxel corners and find their intersection S.

If S is not empty, it must contain precisely one ID, and that is the ID of the constraint with which the voxel was originally seeded. To therefore reconstruct this constraint, we use only the coefficient representations corresponding to the common ID for spectral reconstruction of each voxel corner. We then carry out a weighted trilinear interpolation of the reconstructed curves, which results in the final, uplifted spectrum.

If, on the other hand, S is empty, we perform the interpolation of the spectra at each of the 8 voxel corners. The resulting spectra are then passed as an input

to the voxel's trilinear interpolation. The reason behind this is the same as for the coefficient recalculation (see section 3.1.5), and that is smooth color transitions between various metameric families under different illuminants.

4. Results

In this chapter, we analyze the accuracy of our method for constrained spectral uplifting. We start by focusing on the effects of various implementation details on the results and justifying our decisions regarding their selection. Afterward, we present the actual results achieved in the form of rendered images, and analyze them in terms of their colorimetric properties.

We conclude this chapter by shortly overviewing the performance of our method by providing measurements of both memory utilization and execution time, and proposing improvements for future work.

All assets used in our experiments, such as spectral power distributions of colors atlases and illuminants, are provided by ART [31].

4.1 Implementation choices

The structure of the trigonometric moment cube resulting from the implemented Borgtool's extension greatly depends on the choice of techniques and parameter values used during the implementation. To be specific, the core elements affecting the accuracy include:

- the signal mapping technique used for spectral representation with trigonometric moments
- the number of trigonometric moments used to store spectra
- the utilization of the optimizer, specifically the definition of its residual blocks

This sections overviews our decision-making process regarding these issues, and presents results achieved with other methods to support our claim.

4.1.1 Signal mapping techniques

The first thing we analyze and decide on is the technique used for mapping wavelengths to a signal for the storage and subsequent reconstruction of moments. As already mentioned in section 2.2.1, we have the choice of both mirroring and warping the signal, which overall creates four options — using only mirroring, using only warping, using both or using neither, i.e. utilizing the original signal.

Note that our requirements for the resulting spectral shapes differ depending on the type of the lattice point. For the seeded points, we aim for the highest possible precision in terms of curve reconstruction, so as to lose as little information about the original constraint as possible. Regular points, however, do not have a prior constraint that their spectra must approximate. Therefore, in order to prevent color artifacts upon interpolation and to ensure smooth color transitions between metameric families, we mainly aim for their smoothness.

In addition to the reconstructed shape, we must also take the behavior of the optimizer (i.e. how well it improves its prior coefficients) into account when choosing the signal mapping technique.

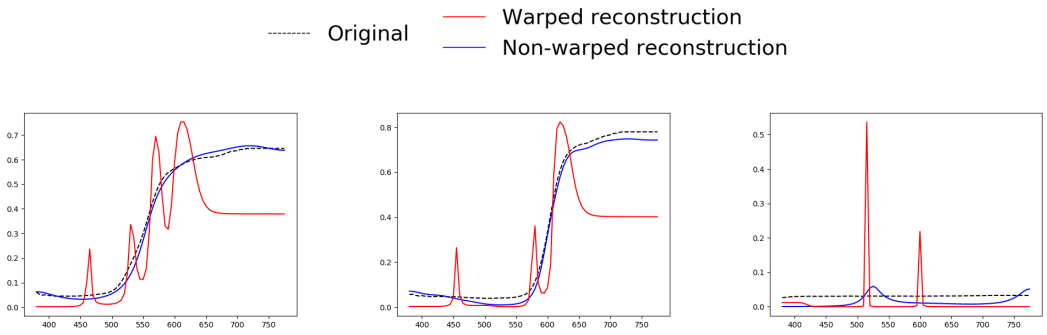
Coefficients	Methods							
	M&W		M&nW		nM&W		nM&nW	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
1	23.88	130.27	23.95	130.62	23.86	130.38	23.95	130.62
2	13.09	97.76	16.92	107.23	—	—	—	—
3	1.39	21.71	10.18	74.62	4.08	51.23	8.48	67.12
4	0.74	7.43	6.3	60.36	—	—	—	—
5	0.49	5.46	2.58	26.36	1.31	20.87	2.56	20.14
6	0.35	3.95	1.1	6.83	—	—	—	—
7	0.31	3.19	0.73	6.12	0.71	8.18	0.89	6.36
8	0.28	2.85	0.71	5.67	—	—	—	—
9	0.27	2.42	0.61	3.94	0.61	5.16	0.46	3.62
10	0.21	2.41	0.43	3.78	—	—	—	—
11	0.21	2.41	0.26	2.62	0.47	4.11	0.37	3.2
12	0.17	2.4	0.2	2.26	—	—	—	—
13	0.17	2.39	0.2	2.32	0.32	3.07	0.27	2.73
14	0.16	2.36	0.2	2.29	—	—	—	—
15	0.16	2.32	0.2	1.93	0.29	2.28	0.24	2.26
16	0.15	2.26	0.18	1.2	—	—	—	—
17	0.15	2.23	0.17	1.17	0.29	1.89	0.23	1.52
18	0.15	2.21	0.15	1.12	—	—	—	—
19	0.15	2.19	0.14	1.12	0.27	1.8	0.19	1.1
20	0.15	2.16	0.14	1.08	—	—	—	—

Table 4.1: The average and maximum Delta E error originating from round-trips under various illuminants. M represents mirroring, W warping, and the symbol n stands for their negation.

We start by focusing on the accuracy of the reconstruction of seeded points. We run an experiment across multiple color atlases (specifically the Pantone Color System, Munsell Book of Color and the Macbeth Color Checker SG) and multiple CIE illuminants in which we compare the average and maximum round-trip errors of spectra under illuminants for all four techniques. Due to its continuous nature, we use the Delta E as the error measure.

In appendix B.1, we provide all results obtained from these experiments. Note that using m moments requires storing $m + 1$ values in case mirroring is used (i.e. the moments are real) and $2m + 1$ values otherwise (i.e. the moments are complex). As, from the implementation point of view, we are interested in the number of $coefficients$ needed for storage (i.e. the number of *double* values per lattice point) rather than the number of moments, we surmise the contents of appendix B.1 in table 4.1, where we present the obtained errors according to the number of coefficients.

By observing table 4.1, we conclude that it is beneficial to mirror the signal. Although the non-mirroring technique performs slightly better for $c \leq 9$, the input atlases often contain more complex spectra, where $c \leq 9$ is insufficient. Additionally, the optimizer behaves similarly for both techniques, and therefore



(a) “orange” patch (b) “red” patch (c) “black” patch

Figure 4.1: Failure of warping when fitting seeded points, shown on patches of the Macbeth Color Chart

does not need to be accounted for.

Table 4.1 also suggests that warping should provide better results. We put this theory to practice and test the performance of the optimizer on the chosen technique. Specifically, our test includes storing atlas entries with a “sufficient” number of coefficients (sufficiency of a representation is determined by the round-trip Delta E error under FL11 illuminant as in section 4.1.2) and using these coefficients as prior for the fitting of closest lattice points. The original and fitted curves are then compared.

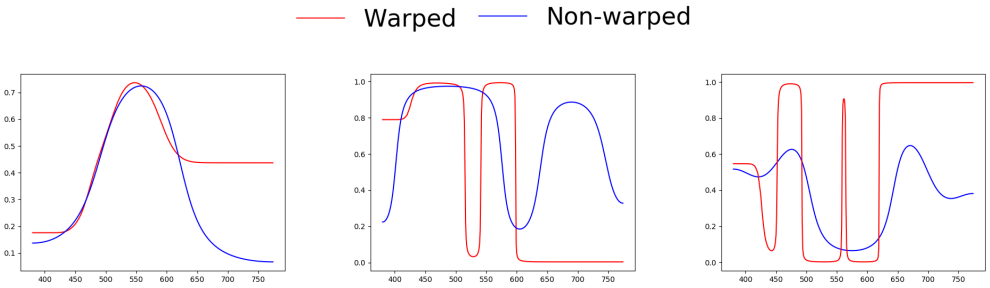
Unfortunately, the results are unsatisfactory. Occasionally, the optimizer fails to recreate the original curve and rather ends up outputting a spiky spectrum susceptible to metameric artifacts. We show some of these results in fig. 4.1.

Therefore, we resort to not warping the signal. Although the optimizer must process more coefficients than if the signal was warped (e.g. if seeding with the Munsell Book of Color with the sufficiency conditions as in section 4.1.2, 16 coefficients are on average required for storing spectra, as opposed to 12 if the signal is warped), the artifacts arising when warping the signal are completely eliminated (see fig. 4.1). Additionally, the average difference between the shape of the original and fitted spectra is substantially lower.

For the regular points, we present a similar comparison of these two methods in fig. 4.2, where we seed cubes of different sizes with distinct atlases and analyze the spectra at specific points. Not warping the signal is superior to warping even in this case, as it creates smoother spectra with less sharp edges.

As we eventually decide on using only 3 coefficients for regular points (see section 4.1.2), and for that purpose, even warping works reasonably well, we do not necessarily need to account for regular points when choosing whether to warp the signal. However, regardless, all the presented evidence points to the superiority of non-warping.

To justify the failures of warping, we analyze its behavior during round-trips in comparison to that of not warping. In fig. 4.3, we present a few such examples on different curves with different number of coefficients. We can clearly observe the behavior for which warping was designated — while the slight waves in the middle of the curve (at around 550nm) are reconstructed almost perfectly, the edges of the curve (i.e. the part of the curve up to 450nm and from 650nm) are flat and attain constant values.



(a) $c = 3, cd = 64$, $RGB = (222.6, 230.7, 230.7)$, initial atlas = Macbeth Color Chart

(b) $c = 5, cd = 32$, $RGB = (82.26, 172.74, 255)$, initial atlas = Page 14 from Munsell Book of Color

(c) $c = 7, cd = 16$, $RGB = (102, 17, 153)$, fitted from middle

Figure 4.2: Comparison of warping and non-warping when used for fitting regular points. Note that the figures are illustrative, as they were created with an older version of the cube and therefore may not exactly correspond to the current results.

Therefore, if the signal is warped, the optimizer has very little influence on the edges of the curve. For that reason, it opts for optimizing only its middle, which might result in amplifying the already created sinusoidal-like shapes.

By not warping the signal, on the other hand, the optimizer focuses on the spectra as a whole and is therefore less prone to spiky artifacts.

Therefore, we conclude that mirroring, but not warping the signal is the optimal approach for our purposes.

4.1.2 Number of moments

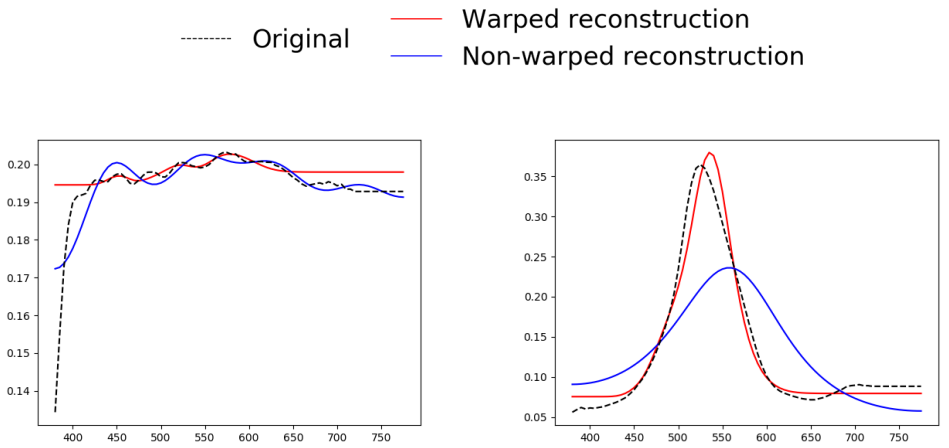
Another important parameter to determine is the number of moments (or coefficients) with which to store spectra at both seeded points and regular points.

We start by analyzing the seeded points. Specifically, we focus on the number of coefficients with which to store constraints in the second step of our implementation (see section 3.1.2), as those are the coefficients that are then used as prior for seeded points. Naturally, we wish for the reconstruction to be as precise as possible, however, we want to prevent unnecessarily high coefficient counts both to avoid high memory utilization and to simplify the run of the optimizer.

By observing fig. 4.4, we see that the Delta E error obtained by round-trips under the CIE illuminants stabilizes at $m = 20$, i.e. $c = 21$. As the curve precision does not worsen by adding more coefficients to the representation, storing all constraints with 21 coefficients might seem to be the optimal solution. Such approach is, however, wasteful for smooth, simple spectra.

Therefore, we decide to store each spectrum with only the necessary amount of coefficients. We obtain this number for each constraint iteratively — starting from $c = 4$, we check whether the coefficient count is sufficient, and, if not, we increase it and move on to the next iteration, repeating this process up to $m = 21$.

We determine the sufficiency of a coefficient representation by picking one of the most error-prone illuminants and determining the round-trip error under said illuminant. If the error falls below a certain, pre-defined threshold, we declare the representation sufficient.



(a) "neutral 5" patch, $c = 9$

(b) "green" patch, $c = 3$

(c) "blue flower" patch, $c = 16$

(d) "foliage" patch, $c = 12$

Figure 4.3: Comparison between the warped and non-warped reconstructed signal shown on multiple patches of the Macbeth Color Chart. In all cases, the signal is mirrored.

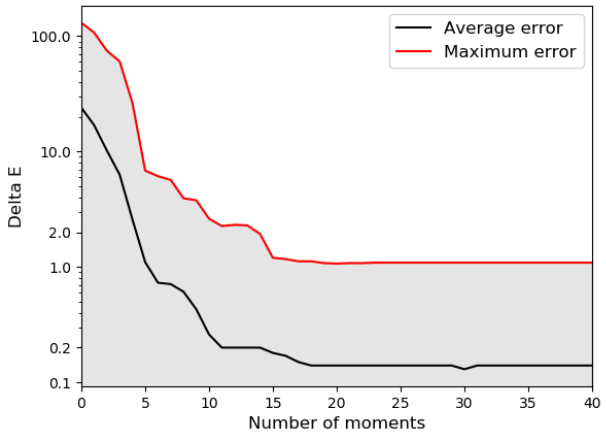


Figure 4.4: The average and maximum Delta E error achieved by round-trips, surmised from table B.1

To determine which illuminant to use, we performed an experiment in which we computed the average number of coefficients required to achieve a specific

Illuminant	Average error	Illuminant	Average error	Illuminant	Average error
A	14.8	F	12.91	F7	12.55
B	5.19	F2	13.01	F8	12.01
C	16.11	F3	12.93	F9	12.09
D50	15.56	F4	12.98	F10	16.24
D65	15.64	F5	12.95	F11	16.46
D75	15.76	F6	13.08	F12	16.30

Table 4.2: The average number of coefficients needed to achieve a round-trip error of $\Delta E^*_{ab} = 0.1$ for different CIE illuminants

error for a set of spectra. We then compared these values for all illuminants from the list of CIE illuminants (excluding the E and D73 illuminants, as they are not a part of the ART database). By the assumption that these results give us a rough approximation of the average error, we concluded that the illuminant with the highest coefficient count is the most error-prone.

We present the results of our experiment in table 4.2. We use an error of $\Delta E^*_{ab} = 0.1$ due to the coefficient count variability under it, but any other would give similar results in terms of the order of the illuminants' coefficient counts.

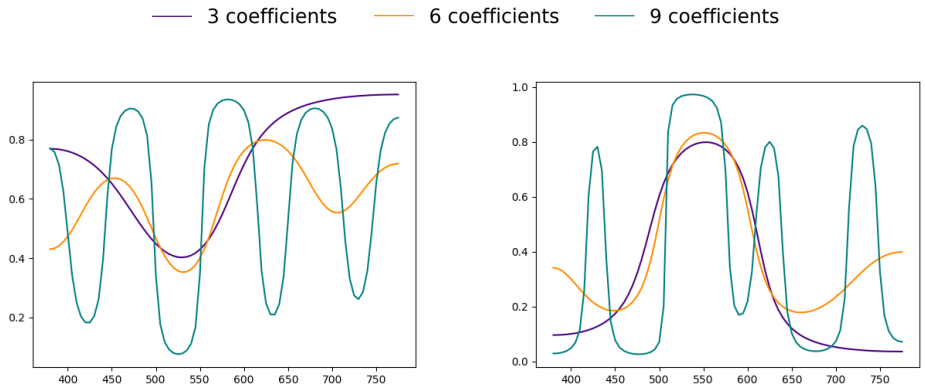
Since the FL11 illuminant requires the highest number of coefficients, we use it for our purposes, and move on to examining the optimal threshold.

Surprisingly, utilizing the Delta E error in our iterative process of acquiring coefficients did not provide as satisfactory results as expected — the entries represented with high number of coefficients ($c \geq 20$) were often unnecessarily accurate, while the entries represented with low number of coefficients ($c \leq 10$) often lacked precision. Therefore, we decided to compute the error as an absolute difference over all three RGB components, which outperformed both the Delta E error and even the Euclidean error in the RGB space (which was prone to similar, but less noticeable, behavior than the Delta E error). We set the threshold to 0.1 for an RGB range of $(0, 255)$. For most of the constraints, such precision is hardly necessary, and could be decreased if we were to focus on memory utilization. However, as the goal of this thesis is to properly assess the accuracy of the uplifting process, we do not concern ourselves with the slight memory overhead.

Although we attribute the failure of Delta E to its non-linearity, we have not yet obtained evidence as to support this claim.

Our approach for determining the sufficiency of coefficients is definitely not flawless. Firstly, neither the threshold, nor the computation of the error have been properly examined. Although our choice of the threshold is purposefully low so as to guarantee proper reconstruction, this might be resulting in an unnecessary memory overhead. Secondly, since we examine only 18 CIE illuminants, there is a high chance that other, more error-prone, exist.

Even the assumption that the average coefficient count determines the worst-performing illuminant falls short. As we have come across multiple atlas entries that attain the highest color error under an illuminant other than FL11, checking whether the error satisfies the threshold for all illuminants might be a more effective approach.



(a) $RGB = (222.1, 106.9, 164.5)$, fitting round $= 8/17$

(b) $RGB = (98.7, 205.6, 41.1)$, fitting round $= 9/17$

Figure 4.5: The effects of multiple choices for the regular points' coefficient count on the shape of their spectra

Furthermore, even if we were to perfect our method, there exist a lot more, vastly different approaches that might be used. However, their exploration and analysis is not within the scope of this thesis, and, as our method produces reasonable results, we leave it for future work.

Finding the sufficient coefficient count for regular points is a lot more straightforward than for the seeded points. We remind that the only requirement is for the spectra to be smooth, so as to avoid interpolation-caused artifacts. This property is especially important if two constraints from different metameric families seed voxels close to each other. Propagating their original shapes during the cube fitting process would, under a different illuminant, create noticeable color artifacts, which is undesired.

Figure 4.2 already suggests that using less coefficients might benefit our smoothness requirement. We put this theory to test by comparing spectra of regular points fitted with a different number of coefficients in cubes of otherwise identical parameters. We present some of the results in fig. 4.5.

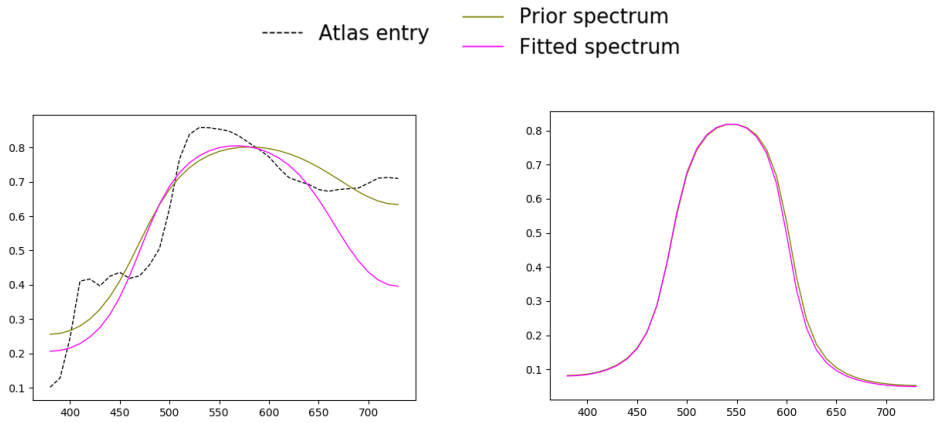
Our assumptions have proven to be correct. By using less coefficients, we limit the degrees of freedom of the optimizer, therefore forcing it to create smooth, simple shapes. Specifically, we decide on using 3 coefficients — using more is unnecessary, but 2 are unable to recreate some of the shapes and occasionally end up reconstructing constant, straight lines.

Additional benefit of using only 3 coefficients per regular point (as opposed to using a higher number) is the lower run-time of cube fitting, as well as less optimizer failures and therefore no need for invocation of heuristic improvements.

4.1.3 Cost functions

In addition to the moment storage technique, another aspect greatly affecting the outcome of the fitting are the cost functions of the optimizer.

For the fitting of the sigmoids, Borgtool uses three cost functions, or *residuals*, each of them specifying the absolute color difference in one axis of the RGB cube. Such an approach outperforms both the Euclidean color distance and even the



(a) Fitting in the second round, i.e. the prior coefficients are the result of "re-calculation" of the fitted coefficients of a seeded point

(b) Fitting in round 8/20, where the prior coefficients are that of a regular point

Figure 4.6: Fitting of regular points with 3 RGB cost functions

Delta E difference — the higher the number of meaningful residuals, the more information can the optimizer deduce about the coefficients' behavior, which, in turn, results in faster and more precise convergence.

As this approach performs rather decently in terms of both time complexity and the obtained results for the sigmoids, we try it out for the purposes of our optimization as well.

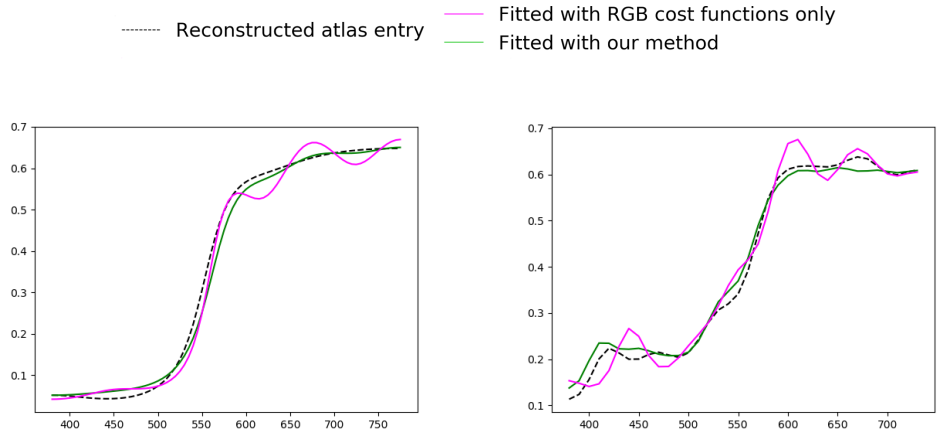
In case of fitting of the regular points (see section 3.1.4), the obtained results are satisfactory, both for the fitting in the second round (i.e. after the "coefficient recalculation") and in the latter rounds (see fig. 4.6). The resulting curves are smooth, they evaluate to the desired RGB values, and the execution time is rather fast. Therefore, we utilize this approach for the regular points.

However, for fitting the seeded points, this method is unsatisfactory. Although it terminates as successful (as the RGB of the resulting curve is within the fitting threshold of the target RGB), the reflectance curve takes on a sinusoidal-like shape with a rather high amplitude, therefore losing resemblance to the original curve. This is due to definition of coefficients, which are, in their nature, Fourier coefficients, and are therefore prone to exhibiting this type of behavior. We show an example of this in fig. 4.7 on the magenta plot.

Note that, in order to keep track of the optimizer's ability to mimic its input, we compare the fitted spectra not with the original spectra of the constraints, but with the spectra that is reconstructed from the original's coefficients.

We therefore conclude that we must incorporate the requirement of curve shape similarity into our cost functions. Following, we review the approaches we attempted along with their outcomes.

The first idea was to implement an approach similar to that for determining the number of coefficients with which to store constraints (see section 4.1.2), i.e. to utilize the color error under a fluorescent illuminant (specifically, FL11). We defined three additional cost functions, each specifying the difference between the original and the reconstructed curve's RGB under the FL11 illuminant in one of the axes. If their values fall below a specific threshold, we terminate the fitting



(a) "orange" patch of the MCC
$c = 8$, $d = 11.84$

(b) 5YR 7/6 patch of the MBC
$c = 14$, $d = 5.85$

(c) N 7.25 patch of the MBC
$c = 20$, $d = 12.67$

(d) "dark skin" patch of the MCC
$c = 12$, $d = 11.55$

Figure 4.7: Comparison between the RGB cost functions and our cost functions for fitting seeded points. c represents the coefficient count, while d is the Euclidean distance between the target and the constraint RGB.

process as successful, if not, we increase the threshold and try again.

Although this approach was successful on average (the average threshold was around $t = 0.025$), occasionally, the threshold often needed to be increased to values so high that it became obsolete (i.e. $t = 1$). Using only one error, either the Euclidean distance or the Delta E error, caused similar issues.

Therefore, we decided to focus on the actual distance between the two curves. Our first attempt consisted of defining one residual per wavelength sample, which specified the absolute distance (as the least square error proved to perform worse) between the two spectra at said wavelength. We examined the behavior of the optimizer both for around 360 residuals (i.e. 1nm increment between samples) and 36 residuals (10nm increment, as defined in most color atlases) when used alongside the 3 already defined RGB residuals. For $c \leq 9$, both of these options performed reasonably well, and for $c > 9$ coefficients, the sufficient threshold was, on average, around $t = 0.0096$, for which the curves were fitted quite accurately.

However, although this method definitely outperformed the previous one, it did not completely eliminate the sinusoidal-like artifacts. During our tests, we came across thresholds as high as $t = 0.23$, which definitely needed improvement.

As we suspected that the failures of the optimizer with lower threshold values were caused by the abundance of cost functions, we summed up their values and saved them into a single residual, which, when divided by the number of spectral samples, represented the average absolute error per sample.

As the importance of curve samples in terms of proper color reconstruction is mainly placed on their middle (at around 550nm), we attempted to add a heuristic-based weighting factor in an effort to focus on minimizing the distance between the two curves in that place. However, we did not succeed in improving our results, and we therefore dropped the experiment and examined the optimizer's behavior without weighting the values.

The proposed method substantially outperformed the previous ones. The threshold error ended up being only about $t = 0.0045$, and, as of yet, no entry requiring $d > 0.03$ has been encountered. Therefore, we concluded the experiments by defining 4 residuals, 3 of them specifying the RGB difference, and 1 specifying the average distance per sample.

We present some of the results achieved with our cost functions in fig. 4.7, where we compare them to fitting with RGB cost functions only. In fig. 4.7c and fig. 4.7d, we specifically focus on the most problematic spectra with the highest distance d between the seeded point and the constraint.

Although our approach substantially reduces the appearance of sinosidual-like shapes, it does not diminish it completely. We can observe it especially in fig. 4.7c. In many cases such as this, however, the deficiency arises mainly due to the imperfections of the moment-based representation, which is, on its own, incapable of reconstructing constant lines due to the nature of the coefficients.

Another drawback of our approach is the substantially greater time complexity, especially if improvement heuristics need to be applied. By examining the scope of the optimizer and utilizing its options further, or maybe even resorting to a different method of optimization, we might be able to improve upon both the time complexity and the resulting spectral shape.

However, as the runtime is not the focus of this thesis and as the resulting shapes are satisfactory for the purposes of accurate uplifting, we leave these improvements for future work.

4.2 Colorimetric properties

In the following, we evaluate the accuracy of our technique when used for uplifting constraints and compare its results to the sigmoid-based uplift as defined by Jakob and Hanika [19]. We then assess its performance when uplifting the RGB gamut as a whole.

4.2.1 Constraint uplifting accuracy

We tested the accuracy of our proposed constrained uplifting approach on the Munsell Book of Color (MBOC), which we utilized as a constraint set for a 32^3-sized coefficient cube for the sRGB color space. Of the 1598 entries in the MBOC, 1396 are in the sRGB gamut: therefore, our experiments only included those. If a larger RGB space (such as e.g. Adobe RGB) were used as input space, the full number of atlas entries could be used: working with sRGB was not due

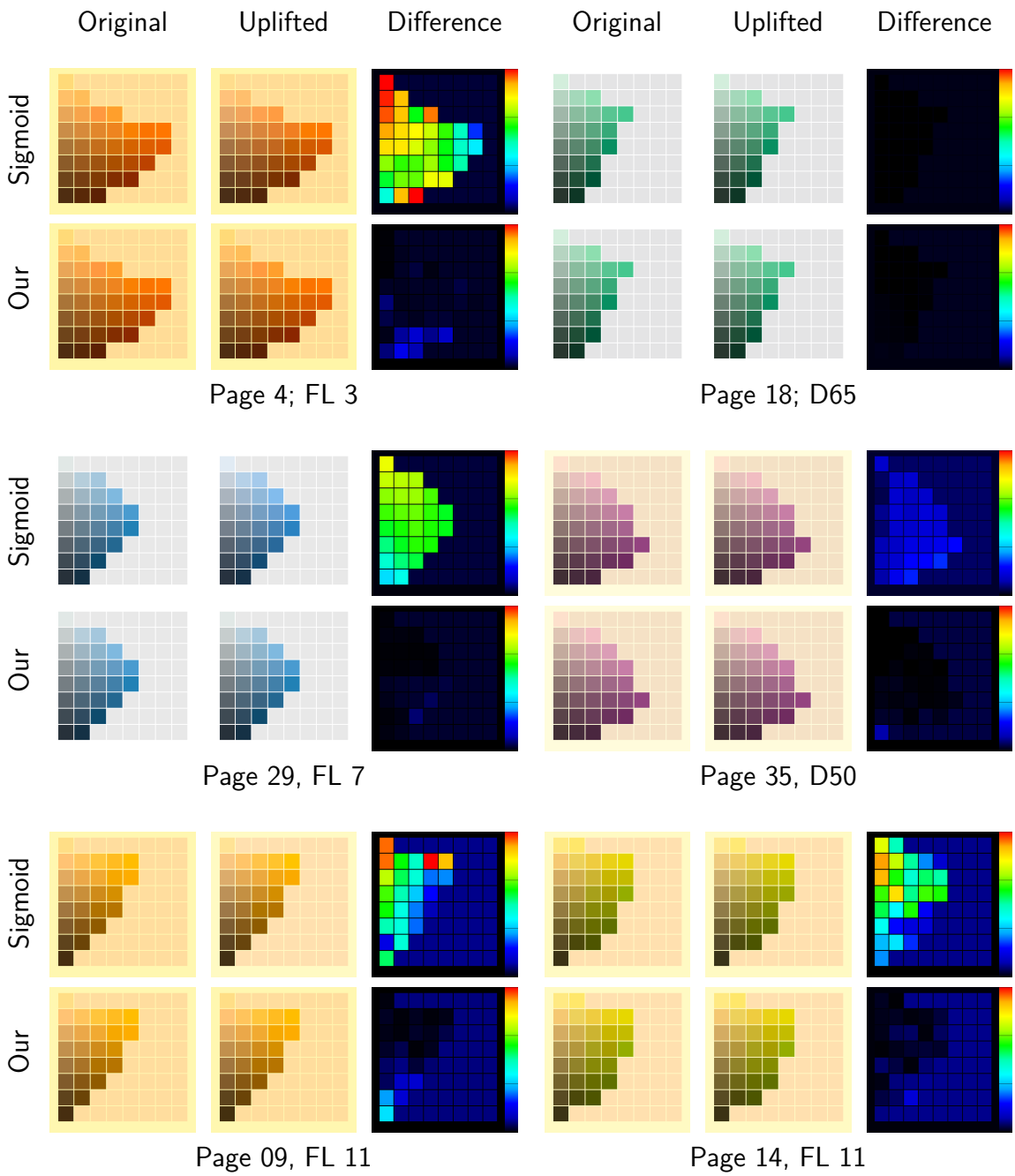


Figure 4.8: Comparison of our uplifting model with the sigmoid-based technique [19] for pages of the Munsell Book of Color under different illuminants. Maximum Delta E in the difference images is 3. Note that patches that fall outside sRGB have been omitted.

to any restrictions in our proposed technique, but only done to stay within the standard RGB space of graphics. Additionally, note that 82 of the atlas entries are not utilized as constraints due to collisions occurring inside voxels, which are unavoidable in a cube with such a low dimension.

We compared the difference between the uplifted spectra to the original ones under a spiky, error-prone illuminant (specifically, FL11) via the standard CIE Delta E color difference metric — under D65, the error was negligible throughout.

The average round-trip error was just a Delta E of 0.21. As the maximum

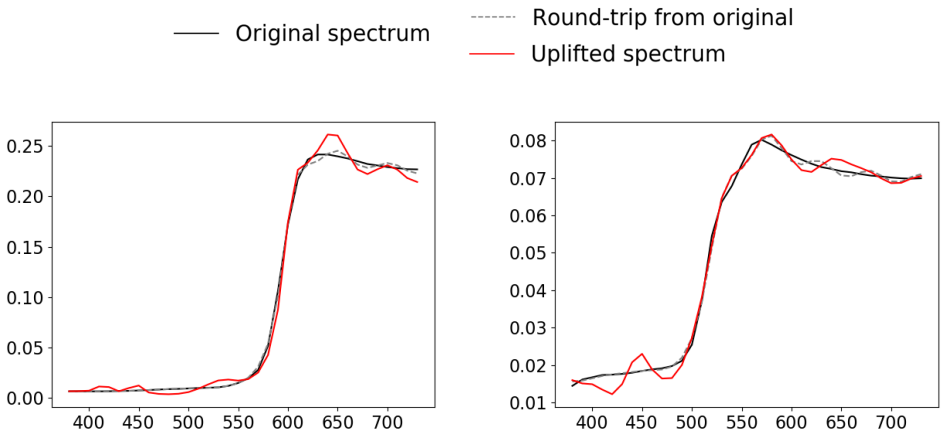


Figure 4.9: Problematic cases of constrained uplifting: reflectance spectra of darker colors. In the areas where the round-trip plot is barely visible, it mimics the original spectrum. Note that the demonstrated range on the y-axis is not $[0, 1]$.

error perceivable by a standard observer is $\Delta E < 1$ [30], this value is negligible and can be regarded as highly satisfactory. Of the 1314 entries, only 22 were found to return a $\Delta E > 1$: all of these had RGB values extremely close to $(0, 0, 0)$.

An example of this behavior in the dark region of the RGB cube can be observed in fig. 4.8. The reason for it is the inadequate fitting of seeded points by the optimizer. This behavior was already present in fig. 4.7, where we concluded that it is due to both the moment representation's inability to flawlessly reconstruct constant curves, and the optimizer's tendency to amplify this deficiency. As the spectral shapes of the dark colors resemble almost constant lines close to zero, it makes sense that the optimizer would primarily fail there.

In fig. 4.9 and fig. 4.10, we provide comparisons of curves of the original spectra, the seeded spectra (labeled *round-trip*, i.e. it represents the direct reconstruction from the moments used to store the original spectra) and the uplifted spectra (i.e. the result of querying the final coefficient cube, including interpolation within the voxel the RGB value lies in). Ideally, all three curves should be identical: a difference between "original" and "round-trip" points to the deficiency of the moment-based spectral reconstruction, while a difference between "round-trip" and "uplift" indicates a drawback in the optimizer. We can observe that, while for darker colors (see fig. 4.9), the errors are mainly due to deficiencies of the optimizer, slight curve deviations for brighter colors (see fig. 4.10) are usually due to imperfections of the moment representation.

However, neither of the mentioned deficiencies severely degrade the accuracy of the uplifting system. On the contrary — in fig. 4.8, we can clearly observe our contribution. While the images uplifted with the sigmoid-based cube demonstrate rather significant color errors, our uplifting system performs modestly and is, except for slight, barely visible discrepancies, almost identical to the original spectral render.

Unfortunately, we have encountered issues when testing the accuracy of our system in a conventional spectral renderer (specifically, ART [31]). Due to minor color deviations that arise during spectral rendering, the RGB values directly

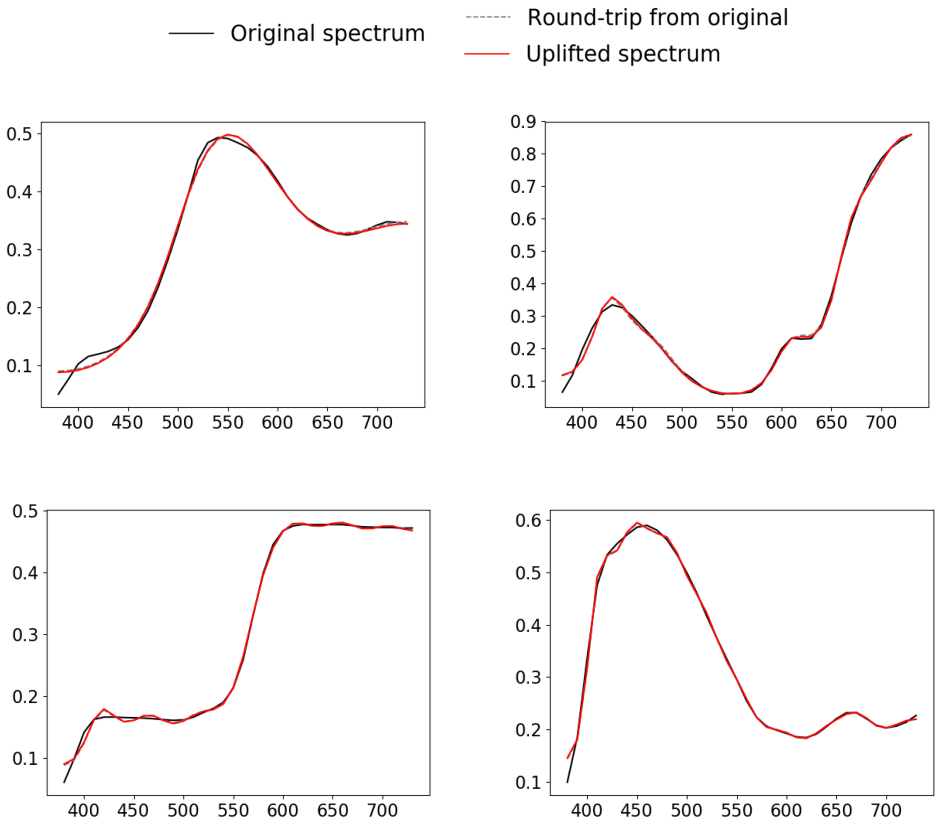


Figure 4.10: Accuracy of constrained uplifting demonstrated on examples of input spectra that correspond to saturated RGB colors. In the areas where the round-trip plot is barely visible, it mimics the uplifted spectrum.

evaluated from the input spectra are not generally equal to the RGB values outputted in the rendered texture. Although the differences are negligible in terms of human color perception, they are high enough to cause the two values to fall into different cube voxels (specifically, to cause the texture RGB to fall into a non-seeded voxel), which in turn results in the insufficient uplift of the texture RGB.

For this reason, in order to demonstrate the correctness of our model on an example scene in fig. 4.11, we do not use the spectrally rendered textures but rather our own, manually-created, which perfectly match the desired RGB values. As the method was not tested in other renderers, we attribute the need for such a process to the inaccuracy of ART.

Due to the inaccuracies during rendering, we do not provide difference images in the actual scene in fig. 4.11. However, our contribution is still visible. In the brighter patches of the middle page (050GY), we can observe how our method preserves the breaking down of the gradients, which are smooth for the sigmoid uplift. For the rightmost page (100Y), both our and the original spectral render display a green tint for their darker patches, which is eliminated in the sigmoid uplift.

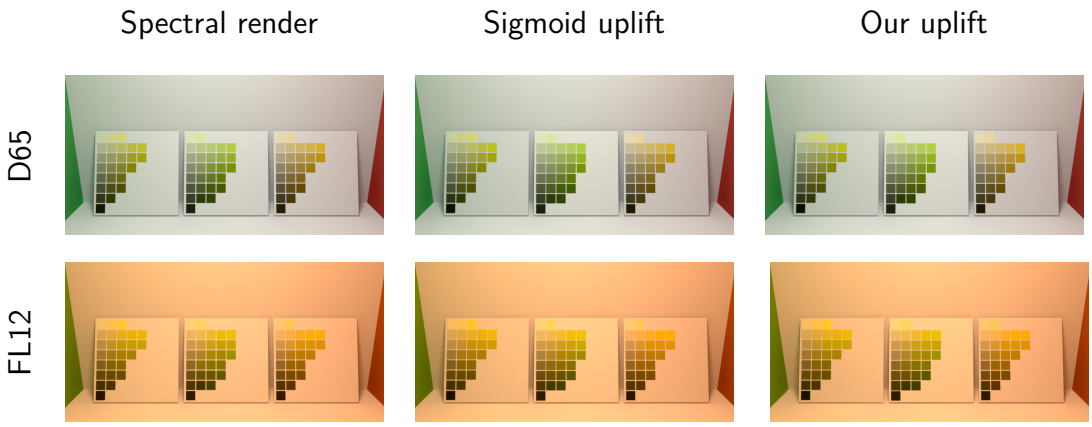


Figure 4.11: Comparison of the sigmoid-based uplift to the constrained uplift on a scene rendered in ART [31]. Pages in scene are (from the left): 100Y, 050GY, 050Y.

4.2.2 Uplift consistency across RGB Space

In order to assess how our technique uplifts the entire RGB gamut (and not just the regions around the seeds), we created multiple coefficient cubes that were seeded with different color atlases. This included a non-constrained cube, i.e. a cube fitted from the middle point at RGB$(0.5, 0.5, 0.5)$ in the same manner as the sigmoid-based approach by Jakob and Hanika [19]).

We first compared their performance in terms of color reconstruction upon uplifting a gradient texture. We specifically selected a gradient with saturated colors in the red-yellow-green region, as that is where differences are most perceivable. The results are shown in fig. 4.12, again illuminated by CIE FL11 (under D65, there are, as per the fitting process, practically zero differences). While the distinctions between individual uplifts under FL11 are barely perceivable by the human eye, the difference images demonstrate that there are some variations – mainly around the locations of seed points, which is precisely what is intended by constraining the uplift process in these locations. None of the gradient textures exhibit any visible discontinuities, though, which indicates that our interpolation approach works properly in the presence of multiple metameric families of reflectance spectra.

In fig. 4.13, we demonstrate that our approach can properly uplift large regions of the RGB gamut simultaneously, without showing artifacts under varying illuminating conditions. We provide multiple uplifts of a rainbow texture covering most of the RGB gamut, constrained with different constraint sets under various illuminants. Although a rather large subset of voxels has been seeded, some of them with complex spectra (especially for the Pantone color atlas), none of these renderings exhibit significant artifacts. Note that all uplifts were performed with 32^3-sized cubes, i.e. some of the constraints may not have been utilized due to collisions during seeding.

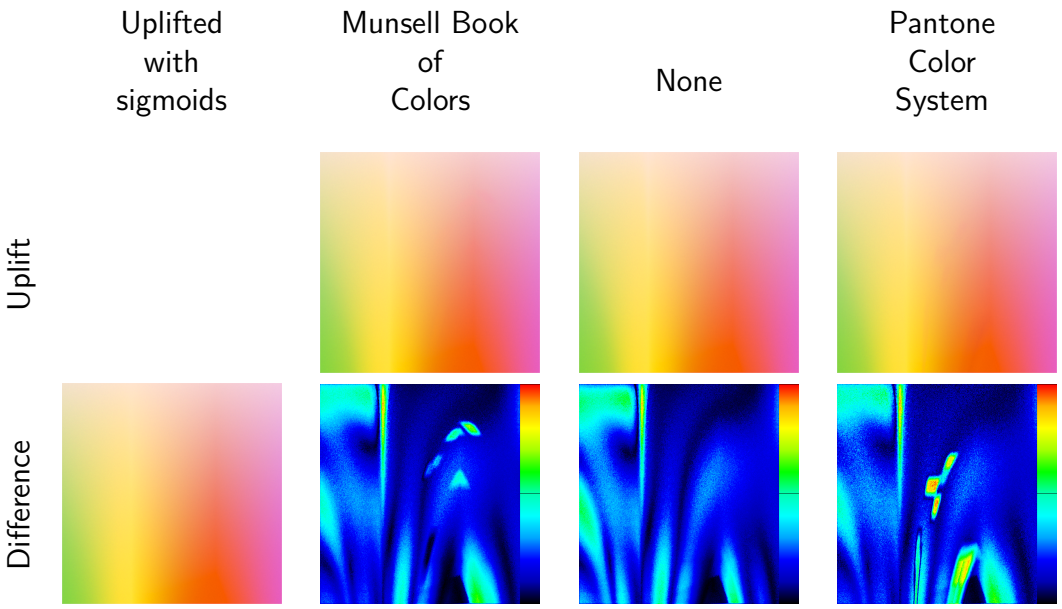


Figure 4.12: A region of the RGB gamut uplifted with different sets of initial spectra, illuminated by FL11. The difference images are relative to maximum $\Delta E = 2$.

4.3 Performance

In this section, we evaluate the performance of our method in terms of both memory and execution time, and propose possible future work for their improvement.

4.3.1 Memory usage

The memory necessary for storing our cube depends on its resolution (i.e. the number of lattice points), which is, in turn, dependent on both the size of our constraint set and on the position of its spectra in the RGB cube. The Macbeth Color Checker (MCC), which contains only 24 entries that are spaced quite far apart from each other in the RGB space (with one of them falling outside of sRGB), requires as little as a 13^3-sized cube. Due to the close proximity of some seeds, the 1396 sRGB entries of the Munsell Book of Color would require as much as 340 lattice points per axis for all seeds to fall into a unique voxel. Additionally, due to the rigid nature of an evenly spaced voxel grid, using a higher cube dimension does not necessarily imply more points that can be successfully seeded. Due to voxel edges being in different positions for different cube dimensions, increasing cube size might even have an adverse effect — for example, while a cube of size 90^3 is sufficient for the RAL Design atlas, in a 300^3-sized cube, 1 point remains unfitted due to a voxel collision.

To store the coefficients of cube entries, we require 3 floating point values for all non-seeded points, and, on average, 16 floating point values per constraint. For the 340^3-sized cube required for the proper coverage of the Munsell Book of Color, this would yield a size of over 450.35MB. Although using less coefficients for storing constraints is possible, it would not noticeably improve the size of the cube — even if we were to use 3 coefficients for all coefficient representations within the cube, the overall size would still be over 449.8MB. That is a negligible

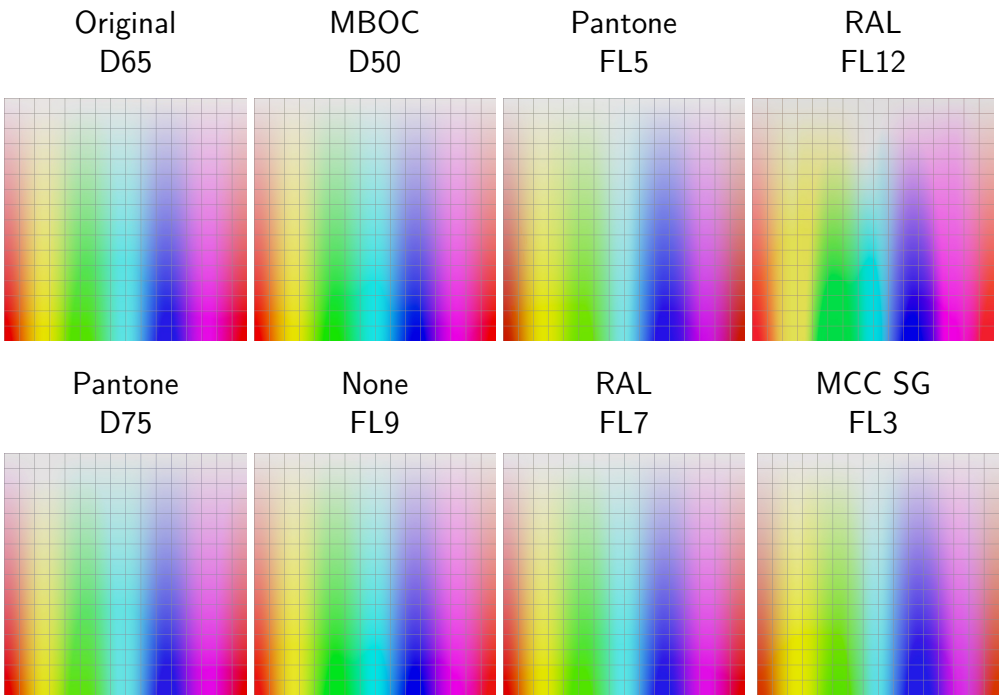


Figure 4.13: Constrained spectral uplifting of a colorful texture for various constraint sets under different illuminants

improvement, as the overall size remains excessive — after all, a seeding of the whole Munsell Book of Color requires only 1396 voxels, which sums up to a maximum of $8 \cdot 1396$ lattice points. Additionally, while most of the regions of the cube are barely utilized, there exist some that have all of their voxels fitted, which might result in a lack of smooth color transitions within these regions.

We therefore conclude that, for the purposes of constrained spectral uplifting for large sets of user-supplied target spectra (like e.g. entire color atlases), a coefficient cube with evenly spaced lattice points is distinctly sub-optimal, in terms of both the memory requirements and its resulting colorimetric properties. For future work, we suggest utilizing a dynamic structure capable of splitting the RGB space into variably-sized voxels according to the number of constraints, such as a kD-tree or an octree.

4.3.2 Execution time

Since our uplifting model is created prior to the rendering process, we separate the evaluation of the execution time of the cube fitting process from the evaluation of the rendering speed when utilizing our cube for uplifting purposes.

We test the execution time of cube fitting on multiple sets of constraints in forms of color atlases, and present the results in table 4.3, where we distinguish the *seeding time* (i.e. the time spent on seeding and fitting of the seeded points) and the *fitting time* (i.e. the time spent on fitting the rest of the cube, i.e. the regular points). All the experiments are performed on an Intel Core i7-8750H CPU (12 logical cores), and the size of each cube is 32^3. Note that such cube resolution may be insufficient for the utilization of all constraints in a given atlas — we therefore provide the *seed count*, which represents the overall number of moment representations stored at seeded points.

Color atlas	Seed count	Seeding time	Fitting time	Overall
Munsell Book of Color	10512	7h 32m 18s	6m 59s	7h 39m 17s
RAL Design Atlas	1336	43m 11s	9m 15s	52m 26s
Macbeth Color Chart SG	576	27m 14s	8m 8s	35m 22s
Macbeth Color Chart	184	9m 29s	14m 17s	23m 46s
None	0	0s	45m 6s	45m 6s
Sigmoid-based approach	0	0s	7m 16s	7m 16s

Table 4.3: Fitting time of a 32^3-sized coefficient cube for multiple color atlases

Due to the higher coefficient count and the strict requirements placed upon the shapes of the reconstructed spectral curves, the fitting of the seeded points takes a lot longer than the fitting of the latter (on average, a seeded point point takes 2.4 seconds to fit, in comparison to the 0.03 for regular points). However, as the cube fitting process is multi-threaded, it benefits from multiple seeds evenly positioned across the RGB cube. This is particularly obvious when comparing the performance of the fitting of the Macbeth Color Charts and the fitting without constraints.

None of these use-cases outperform the sigmoid-based cube in terms of fitting time. While our technique needs to use various complex mathematical operations, such as the application of Levinson's algorithm, Herglotz transform and multiple other conversion processes [34], not to mention the interpolation of metamers for lattice points with multiple representations, the sigmoid-based approach evaluates the spectral curve with as little as six floating point operations for any given wavelength [19].

This drawback of our technique also carries over to using our method during rendering. In order to properly evaluate the execution time, we perform two tests — firstly, we compare the performance of the sigmoid-based cube with the performance of our non-constrained cube (in order to avoid the overhead of spectral reconstruction from higher-dimensional coefficient representations), and secondly, we provide measures of the execution time for the constrained uplift for the renders in fig. 4.8. All experiments are performed on an Intel Xeon CPU E5-2680 v3 (48 logical cores), with 32^3-sized cubes.

For the latter experiments, the sigmoid-based approach performed, on average, 2.1 times better than our constrained cubes. The performance overhead arising from constraining the uplifting process was expected. When used in a spectral renderer – specifically, ART [31] — on a closely viewed texture of one of the pages of the Munsell Book of Color (that is, when pretty much all the pixels in the image correspond to the constraints), rendering times slow down by about a factor of 4 when compared to the sigmoid uplift.

For non-constrained uplifting, our method provides no benefits compared to the sigmoid-based approach, except perhaps that it creates slightly more varied spectral shapes than the plain sigmoid technique. Even when not constrained, uplifting of the colorful textures shown in fig. 4.13 is 2.3 times slower with our cube than with the sigmoid-based technique.

That having been said, we wish to point out that so far, our focus was placed on the correctness and accuracy of the constrained uplift, i.e. our implementation

of both the model creation and its utilization in a renderer does not include any real optimizations yet. In the future, these could be applied to the fitting process (by further exploiting the possibilities of the CERES solver, or, possibly, another optimization technique) and to the actual uplifting, which currently does not cache any intermediate values (such as the exponential moments) and therefore requires them to be unnecessarily re-computed during each uplift. We estimate that such optimizations could improve the performance up to a factor of two, both during fitting and during rendering.

4.4 Future work

In this chapter, we have already mentioned multiple deficiencies of our system. Following, we surmise the most important ones and outline their possible solutions:

- *Cube structure*

  As mentioned in section 4.3.1, evenly spacing lattice points in the RGB cube is not the optimal approach in terms of memory utilization, especially if the constraint set has a high number of spectra evaluating to RGB values in close vicinity. In order to improve both the memory requirements and the resulting colorimetric properties, we suggest utilizing a dynamic structure capable of splitting the RGB space into variably-sized voxels according to the number of constraints, such as a kD-tree or an octree.

- *Execution time*

  In order to improve the execution time of the uplifting process in a renderer, multiple optimizations could be added, such as storing intermediate values (e.g. exponential moments during spectral reconstruction), or perfecting the process of determining whether the voxel has been seeded prior to trilinear interpolation (see section 3.2). Some of these optimization could also be included in the cube fitting process.

- *Supported color gamuts*

  Currently, the only supported gamut is the sRGB color gamut. This renders some of the constraints in the standard color atlases, such as the Munsell Book of Color or even the Macbeth Color Chart, unusable. In the future, we propose adding the possibility to choose a color gamut within which we want to uplift.

- *Sufficient moment count*

  As the conditions for the sufficient moment count have not been properly examined and, for the purposes of this thesis, most often than not, higher count than required is used, the memory requirements could be further improved by selecting more suitable conditions. As a starting point, we suggest increasing the sufficiency threshold, but utilizing multiple distinct illuminants.

- *Fitting of the seeded points*

  Currently, due to the optimizer's tendency to get stuck in local minima when presented with a high number of parameters, the fitting of the seeded points requires a lot of heuristic-based optimizations (see section 3.1.3). This, in turn, takes a lot of time and may not always produce the desired results. This could be solved by either further exploiting the possibilities of the CERES solver, or, possibly, utilizing another optimization technique.

- *Unfitted points around* $RGB \approx (255, 255, 255)$

  As mentioned in section 3.1.5, lattice points evaluating roughly to $RGB \approx (255, 255, 255)$ may, under specific conditions, fail the fitting process. As this is due to the inaccuracies of ART, we propose utilizing a different library for color conversion purposes.

- *Deficiency in the cube's dark region*

  As the color errors occurring in the dark region of the cube are mainly due to the deficiency of the optimizer (see section 4.2.1), it is likely that they could be eliminated by either further exploiting the possibilities of the optimizer or by choosing a different optimization technique. Another option is to propose a special-case handling for curves evaluating to such low RGB values. This could be done in a similar manner than for the points with $RGB \approx (255, 255, 255)$ (see section 3.1.5).

Conclusion

In this thesis, we presented the first method capable of constraining the spectral uplifting process with an arbitrary set of target spectra. By utilizing a trigonometric moment-based approach for spectral representation, the RGB values of the target spectra are accurately uplifted to their original spectral shapes, while the rest of the RGB gamut uplifts to smooth spectra. This results in smooth transitions between the various metameric families that originate from the constraining process.

Our model shows a slight weakness when uplifting very dark colors, which we attribute both to the inability of the moment-based spectral representations to represent constant spectra, and to the deficiency of the optimization process used during the creation of our uplifting model. However, even including these minor drawbacks, the results in terms of color accuracy are noteworthy, as the uplifted curves describe the original ones with negligible differences.

Neither the memory, nor the execution time of either the creation or the utilization of our model are optimal: the new and so far unique capability to perform targeted uplifts comes at the cost of some overhead that is not present in e.g. the unconstrained sigmoid uplift technique of Jakob and Hanika [19].

In the future, we will primarily focus on utilizing a more suitable and memory efficient structure for storing the constraints, such as a kD-tree or an octree. Secondly, we will improve the execution time by optimizing the moment-based spectral reconstruction process, and, furthermore, we will focus on improving other minor deficiencies, reviewed in section 4.4.

Bibliography

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. 2012.

[3] Anikethan Bekal, Ajit M Hebbale, and MS Srinath. Review on material processing through microwave energy. In *IOP Conference Series: Materials Science and Engineering*, 2018.

[4] Laurent Belcour and Pascal Barla. A practical extension to microfacet theory for the modeling of varying iridescence. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[5] Arthur D Broadbent. A critical review of the development of the cie1931 rgb color-matching functions. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 29(4):267–272, 2004.

[6] John Parker Burg. Maximum entropy spectral analysis. *Astronomy and Astrophysics Supplement*, 15:383, 1974.

[7] Kyungah Choi, Jeongmin Lee, and Hyeon-Jeong Suk. Context-based presets for lighting setup in residential space. *Applied Ergonomics*, 52:222–231, 01 2016. doi: 10.1016/j.apergo.2015.07.023.

[8] Asim Kumar Roy Choudhury. *Principles of colour and appearance measurement: Object appearance, colour perception and instrumental measurement*. Elsevier, 2014.

[9] CIE. Commission internationale de l'éclairage, 1913. URL `http://cie.co.at/`.

[10] D Drosdoff and A Widom. Snell's law from an elementary particle viewpoint. *American journal of physics*, 73(10):973–975, 2005.

[11] Hugh S Fairman, Michael H Brill, and Henry Hemmendinger. How the cie 1931 color-matching functions were derived from wright-guild data. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 22(1):11–23, 1997.

[12] Lori Gardi. Planck's constant and the nature of light, 05 2018.

[13] TM Goodman. International standards for colour. In *Colour Design*, pages 177–218. Elsevier, 2012.

[14] Igor Griva, S Nash, and A Sofer. Nonlinear least squares data fitting. *Linear and Nonlinear Optimization*, pages 743–758, 2009.

[15] George G Guilbault. *Practical fluorescence*. CRC Press, 2020.

[16] Martin Habekost. Which color differencing equation should be used. *International Circular of Graphic Education and Research*, 6:20–33, 2013.

[17] CIE HunterLab. L* a* b* color scale. *Applications note, Virginia, USA*, 1996.

[18] Noor A Ibraheem, Mokhtar M Hasan, Rafiqul Z Khan, and Pramod K Mishra. Understanding color models: a review. *ARPN Journal of science and technology*, 2(3):265–275, 2012.

[19] Wenzel Jakob and Johannes Hanika. A low-dimensional function space for efficient spectral upsampling. In *Computer Graphics Forum*, volume 38, pages 147–155. Wiley Online Library, 2019.

[20] Alisa Jung, Alexander Wilkie, Johannes Hanika, Wenzel Jakob, and Carsten Dachsbacher. Wide gamut spectral upsampling with fluorescence. In *Computer Graphics Forum*, volume 38, pages 87–96. Wiley Online Library, 2019.

[21] Douglas A Kerr. The cie xyz and xyy color spaces. *Colorimetry*, 1(1):1–16, 2010.

[22] David H Krantz. Color measurement and color theory: Ii. opponent-colors theory. *Journal of Mathematical Psychology*, 12(3):304–327, 1975.

[23] Kreĭ. *The Markov moment problem and extremal problems*.

[24] Henry J Landau. Maximum entropy and the moment problem. *Bulletin of the American Mathematical Society*, 16(1):47–77, 1987.

[25] David L MacAdam. The theory of the maximum visual efficiency of colored materials. *JOSA*, 25(8):249–252, 1935.

[26] André Markoff. Nouvelles applications des fractions continues. *Mathematische Annalen*, 47(4):579–597, 1896.

[27] Manuel Melgosa. Testing cielab-based color-difference formulas. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 25(1):49–55, 2000.

[28] Johannes Meng, Florian Simon, Johannes Hanika, and Carsten Dachsbacher. Physically meaningful rendering using tristimulus colours. In *Computer Graphics Forum*, volume 34, pages 31–40. Wiley Online Library, 2015.

[29] Michal Mojzík. Fluorescence computations in a hero wavelength renderer. 2018.

[30] WS Mokrzycki and M Tatol. Colour difference delta e-a survey. *Mach. Graph. Vis*, 20(4):383–411, 2011.

[31] Computer Graphics Group of Charles University in Prague. Art, . URL `https://cgg.mff.cuni.cz/ART/gallery/`.

[32] Computer Graphics Group of Charles University in Prague. Art sigmoids, . URL `https://cgg.mff.cuni.cz/ART/archivers/art_2_0_3.html`.

[33] Hisanari Otsu, Masafumi Yamamoto, and Toshiya Hachisuka. Reproducing spectral reflectances from tristimulus colours. In *Computer Graphics Forum*, volume 37, pages 370–381. Wiley Online Library, 2018.

[34] Christoph Peters, Sebastian Merzbach, Johannes Hanika, and Carsten Dachsbacher. Using moments to represent bounded signals for spectral rendering. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[35] Christoph Peters, Sebastian Merzbach, Johannes Hanika, and Carsten Dachsbacher. Spectral rendering with the bounded mese and srgb data. In *Workshop on Material Appearance Modeling*, volume 2019, pages 07–09, 2019.

[36] Dale Purves, G Augustine, D Fitzpatrick, L Katz, A LaMantia, J McNamara, and S Williams. Neuroscience 2nd edition. sunderland (ma) sinauer associates, 2001.

[37] Javier Romero, E. Valero, Javier Hernández-Andrés, and Juan Nieves. Color-signal filtering in the fourier-frequency domain. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 20:1714–24, 10 2003. doi: 10.1364/JOSAA.20.001714.

[38] Iman Sadeghi and HW Jensen. A physically based anisotropic iridescence model for rendering morpho butterflies photo-realistically. *Proc. of Iridescence: More Than Meets the Eye (Tempe, Arizona, 2008)*, page 38, 2008.

[39] Gaurav Sharma and Carlos Eduardo Rodríguez-Pardo. The dark side of cielab. In *Color Imaging XVII: Displaying, Processing, Hardcopy, and Applications*, volume 8292, page 82920D. International Society for Optics and Photonics, 2012.

[40] Gaurav Sharma, Wencheng Wu, Edul N Dalal, and Mehmet U Celik. Mathematical discontinuities in ciede2000 color difference computations. In *Color and Imaging Conference*, volume 2004, pages 334–339. Society for Imaging Science and Technology, 2004.

[41] Brian Smits. An rgb-to-spectrum conversion for reflectances. *Journal of Graphics Tools*, 4(4):11–22, 1999.

[42] Andrew Stockman and Lindsay T Sharpe. Cone spectral sensitivities and color matching. *Color vision: From genes to perception*, pages 53–88, 1999.

[43] Yinlong Sun. Rendering biological iridescences with rgb-based renderers. *ACM Transactions on Graphics (TOG)*, 25(1):100–129, 2006.

[44] Yinlong Sun, F David Fracchia, and Mark S Drew. Rendering light dispersion with a composite spectral model. *Diamond*, 2(37.17):0–044, 2000.

[45] Arthur Robert Weeks, Carlos E Felix, and Harley R Myler. Edge detection of color images using the hsl color space. In *Nonlinear Image Processing VI*, volume 2424, pages 291–301. International Society for Optics and Photonics, 1995.

[46] Guillaume Loubet Sébastien Speierer Benoît Ruiz Delio Vicini Wenzel Jakob, Merlin Nimier-David and Tizian Zeltner. Mitsuba2. URL `https://mitsuba2.readthedocs.io/en/latest/src/getting_started/variants.html`.

[47] John Werner. Human colour vision: 1. colour mixture and retino-geniculate processing. 10 2001. doi: 10.1142/9789812811899_0003.

[48] Sebastian Werner, Zdravko Velinov, Wenzel Jakob, and Matthias B Hullin. Scratch iridescence: Wave-optical rendering of diffractive surface structure. *ACM Transactions on Graphics (TOG)*, 36(6):1–14, 2017.

[49] N Whetzel. Measuring color using hunter l, a, b versus cie 1976 l* a* b*. *Application notes). Retrieved from Hunterlab website: https://support. hunterlab. com/hc/enus/articles/204137825-Measuring-Color-using-Hunter-Lab-versus-CIE-1976-Lab-AN-1005b*, 2016.

[50] Alexander Wilkie, Robert F Tobler, and Werner Purgathofer. Raytracing of dispersion effects in transparent materials. 2000.

[51] Alexander Wilkie, Robert F Tobler, and Werner Purgathofer. Combined rendering of polarization and fluorescence effects. In *Rendering Techniques 2001*, pages 197–204. Springer, 2001.

A. Software user guide

In this appendix, we provide a user guide for compiling and running both Borgtool (for the purposes of creating the cube) and ART (for the purposes of its utilization).

Both softwares have been tested on Ubuntu 20.04 with CMake 3.16.3 and g++ 8.4.0. As we do not provide any binaries, the user must compile the projects first.

A.1 Borgtool

To build Borgtool, execute the following steps:

1. Unzip the provided attachment and open the *Borgtool* folder
2. Make sure you have installed ART and Ceres (see CMakeLists.txt for more details about necessary dependencies)
3. Build the project using CMake

It is possible to run the program with several options. Following, we provide the list of the ones supported with the trigonometric moment-based method:

```
Usage: borgtool [-options]
where the options include:
    (-cc | --createCube) <filename>
        fit new RGB coefficient cube
    -mo | --momentOpt
        use moments for spectral representation
    (-cd | --cubeDimension) <numpoints>
        # of cube lattice points in one dimension
    (-ft | --fittingThreshold) <x>
        threshold in fraction 1/x of voxel size
    -pi | --progressImages
        generate fitting progress images
    -ofr | --onlyFirstRound
        do not attempt to improve the cube
    (-tex | --texture) <filename>
        create EXR test texture for cube testing
    (-ctx | --cubeTexture) <filename>
        create EXR test texture with cube data
    (-tv | --textureV) <0..1>
        HSV V value for the two preceding textures
    (-sfd | --showFittingDelta) <0..1>
        show delta between lattice and target RGB
    (-uc | --useCorner) <index>
        use coefficents from voxel corner #
    (-ply | --generatePLY) <filename>
        create PLY geometry for UCC file
```

```
-srgb | --sRGB
    use sRGB (default)
(-a | --atlas) <atlasID>
    seed the cube with atlas with ID #
```

Examples of usage:

- `borgtool -mo -cc cube_mcob -cd 64 -a 0`

  This command creates a new trigonometric moment-based RGB cube of size 64^3 seeded with an atlas with $ID = 0$, which is the Munsell Book of Color.

- `borgtool -mo -cc cube_middle`

  This command creates a new trigonometric moment-based RGB cube of size 32^3. As no atlas is specified, the cube grows from its center.

- `borgtool -mo -rc cube_mcc -ctx test_texture -tv 0.5`

  This command loads a cube with the name `cube_mcc` and utilizes it for uplifting a rainbow texture (see fig. 4.13). The brightness of the texture is set to 0.5, and the resulting uplift is stored in a file named `test_texture`.

A.2 ART

To build ART, execute the following steps:

1. Unzip the provided attachment and open the *ART* folder
2. Build the project using the instructions from the ART website at `https://cgg.mff.cuni.cz/ART/download/`

In order to utilize the cubes created by Borgtool, execute the following steps:

1. Copy the cubes you wish to utilize to `ART/ART_Resources/SpectralUplift`
2. Add the following code snippet:

   ```
   SET_UPLIFT_CUBE(
       "my_cube_name.ucc"
   ),
   ```

   to the action sequence of the `.arm` file whose image map you wish to uplift. If this option is omitted, the sigmoid cube is utilized.

3. run the `artist` command with the desired parameters

A.2.1 Example scenes

In the `Resources` folder found in the attachment of this thesis, we provide multiple example scenes along with textures and cubes, some of which have been used to render images in this thesis. Following, we provide instructions on how to replicate these renders, and overview the contents of the attached folder.

Before running the example scenes, do the following:

1. Copy all the `.ucc` files from the provided `Resources/cubes` folder to `ART/ART_Resources/SpectralUplift`
2. Copy all the `.tif` and `.tiff` files from the provided `Resources/textures` folder to `ART/ART_Resources`

Contents of the `Resources` folder include:

- Resources for replicating fig. 4.11:
  - `scenes/Three_Pages_Original.arm` file for replicating the original spectral render. Rendering it does not require any additional resources.
  - `scenes/Three_Pages_Texture.arm` file for replicating the constrained and the sigmoid-based uplift. Additional resources required are as follows:
    - uplift cube: `cubes/three_pages.ucc`
    - image maps:
      `cubes/three_pages_page10.tiff`
      `cubes/three_pages_page12.tiff`
      `cubes/three_pages_page14.tiff`
- Resources for replicating fig. 4.12:
  - scene file: `scenes/Gradient_Texture.arm`
  - image map: `textures/gradient.tif`
  - cubes:
    `cubes/mboc_cd32.ucc`
    `cubes/none_cd32.ucc`
    `cubes/pantone_cd32.ucc`

  By default, the cube specified for uplifting in the `Gradient_Texture.arm` scene is `pantone_cd32`. In order to render with a different cube, change the uplift cube in the scene description.
- Three additional scenes uplifting three pages of the Munsell Book of Color:
  - `scenes/Page10_Texture.arm`
  - `scenes/Page12_Texture.arm`
  - `scenes/Page14_Texture.arm`

  For each of the scenes, a corresponding image map and an uplift cube can be found in `Resources/textures` and `Resources/cubes` respectively.

Note: It might happen that some of the provided scenes do not render the desired result. This is due to a known bug in the ART library. In case it occurs, we recommend building ART in the `Debug` mode without any optimizations.

B. Attachments

B.1 Delta E error caused by moment sampling

Moments	Methods							
	M&W		M&nonW		nMW		nMnW	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
0	23.88	130.27	23.95	130.62	23.86	130.38	23.95	130.62
1	13.09	97.76	16.92	107.23	4.08	51.23	8.48	67.12
2	1.39	21.71	10.18	74.62	1.31	20.87	2.56	20.14
3	0.74	7.43	6.3	60.36	0.71	8.18	0.89	6.36
4	0.49	5.46	2.58	26.36	0.61	5.16	0.46	3.62
5	0.35	3.95	1.1	6.83	0.47	4.11	0.37	3.2
6	0.31	3.19	0.73	6.12	0.32	3.07	0.27	2.73
7	0.28	2.85	0.71	5.67	0.29	2.28	0.24	2.26
8	0.27	2.42	0.61	3.94	0.29	1.89	0.23	1.52
9	0.21	2.41	0.43	3.78	0.27	1.8	0.19	1.1
10	0.21	2.41	0.26	2.62	0.28	1.81	0.18	1.28
11	0.17	2.4	0.2	2.26	0.29	1.81	0.16	1.07
12	0.17	2.39	0.2	2.32	0.27	1.64	0.15	1.09
13	0.16	2.36	0.2	2.29	0.25	1.58	0.15	1.09
14	0.16	2.32	0.2	1.93	0.24	1.56	0.16	1.09
15	0.15	2.26	0.18	1.2	0.2	2.66	0.16	1.09
16	0.15	2.23	0.17	1.17	0.21	4.3	0.15	1.08
17	0.15	2.21	0.15	1.12	0.25	4.27	0.14	1.08
18	0.15	2.19	0.14	1.12	0.29	3.3	0.14	1.08
19	0.15	2.16	0.14	1.08	0.44	4.04	0.14	1.08
20	0.15	2.12	0.14	1.07	0.57	4.24	0.15	1.07
21	0.15	2.06	0.14	1.08	0.65	3.42	0.15	1.08
22	0.15	2.01	0.14	1.08	0.37	2.53	0.14	1.09
23	0.15	1.98	0.14	1.09	0.36	2.16	0.14	1.1
24	0.15	1.96	0.14	1.09	0.22	1.44	0.14	1.12
25	0.16	1.95	0.14	1.09	0.22	1.17	0.15	1.14
26	0.16	1.93	0.14	1.09	0.19	1.04	0.16	1.19
27	0.16	1.9	0.14	1.09	0.24	1.11	0.16	1.24
28	0.16	1.87	0.14	1.09	0.19	0.84	0.15	1.29
29	0.17	1.82	0.14	1.09	0.14	0.97	0.13	0.94
30	0.17	1.78	0.13	1.09	0.15	0.99	0.16	1.13
31	0.18	1.94	0.14	1.09	0.16	1.04	0.21	1.59
32	0.19	1.89	0.14	1.09	0.15	0.9	0.29	1.62
33	0.21	1.76	0.14	1.09	0.13	0.95	0.27	1.62
34	0.21	1.86	0.14	1.09	0.13	1.26	0.2	1.46
35	0.19	1.79	0.14	1.09	0.16	1.84	0.23	1.17
36	0.2	1.82	0.14	1.09	0.16	1.71	0.22	1.19
37	0.16	1.79	0.14	1.09	0.19	1.82	0.15	1.08

38	0.11	1.77	0.14	1.09	0.23	2.23	0.1	0.72
39	0.12	1.67	0.14	1.09	0.34	2.28	0.07	0.73
40	0.13	1.68	0.14	1.09	0.27	1.54	0.06	0.73