A Multi-illuminant Synthetic Image Test Set

Xiangpeng Hao, Brian Funt; Simon Fraser University, Vancouver, Canada

*Abstract--***A new multi-illuminant synthetic image test set called MIST is described and made publicly available. MIST is intended primarily for evaluating illumination estimation and color constancy methods, but additional data is provided to make it useful for other computer vision applications as well. MIST addresses the problem found in most existing real-image datasets, which is that the groundtruth illumination is only measured at a very limited number of locations, despite the fact that illumination tends to vary significantly in almost all scenes. In contrast, MIST provides for each pixel: (i) the percent surface spectral reflectance, (ii) the spectrum of the incident illumination, (iii) the separate specular and diffuse components of the reflected light, and (iv) the depth (i.e., camera-to-surface distance). The dataset contains 1,000 stereo pairs, each of the 2,000 images being a 30-band multi-spectral image covering the visible spectrum from 400nm to 695nm at a 5nm interval. Standard sRGB versions of the multi-spectral images are also provided. The images are synthesized by extending the Blender Cycles ray-tracing renderer. The rendering is done in a way that ensures the images are not only photorealistic, but physically accurate as well.**

Index Terms—spectral rendering, illumination estimation, color constancy, full-spectrum image dataset.

I. INTRODUCTION

Until relatively recently, the majority of work on colour constancy has been based on the assumption that the scene is lit by a single illuminant. Everyone is aware that this assumption is very often violated, but there are only very limited image datasets providing groundtruth data for multi-illuminant scenes. Furthermore, these data sets are small and mainly consist of images of quite constrained single-object scenes. In response, we have developed and made public a new image test set called MIST that includes 1,000 full-spectrum stereo image pairs of complex scenes. The images are synthetically generated and accompanied by pixelwise groundtruth data that includes, for each pixel, its percent surface spectral reflectance, the spectrum of the incident illumination and its depth (i.e., camera-to-surface distance). The pixel-wise spectrum of the reflected light is also provided in terms of its separate diffuse and specular components. Although the primary objective in constructing MIST is to aid in the evaluation of illumination estimation methods for colour constancy, the inclusion of depth and stereo data is expected to be useful for testing other computer vision and hyper-spectral algorithms as well.

II. RELATED WORK

The majority of research on illumination estimation methods [1][2][3][4][5][6] has focused on determining the chromaticity of the dominant scene illuminant. These single-illumination methods are evaluated on datasets for which a single groundtruth measurement of the dominant scene illumination is measured using a ColorChecker [7] or a grey object placed in the scene. Examples of these datasets are the Gehler-Shi ColorChecker set [8] (recently improved and standardized by Hemrit et al. [7]), the NUS set [9] the SFU Greyball set [10], and the Cube+ dataset of 1,707 images taken with a single Canon EOS 550D camera [11]. However, as Xu and Funt [12] point out, not all the images are, in fact, of single-illuminant scenes. For the Gehler-Shi set, in particular, as many as 258 of the 568 scenes involve multiple illuminants.

Illumination estimation methods designed to handle multi-illuminant scenes include those of Beigpour et al. [13], Funt and Mosny [14], Gijsenij et al. [15], Bianco et al. [16] and Joze and Drew [17]. For testing Gijsenij et al. [15] created a small dataset consisting of 59 multi-illuminant laboratory images obtained using filters and multiple exposures of the same static scene. In addition, they provide an additional 9 multi-illuminant outdoor images, with the illumination measured using a

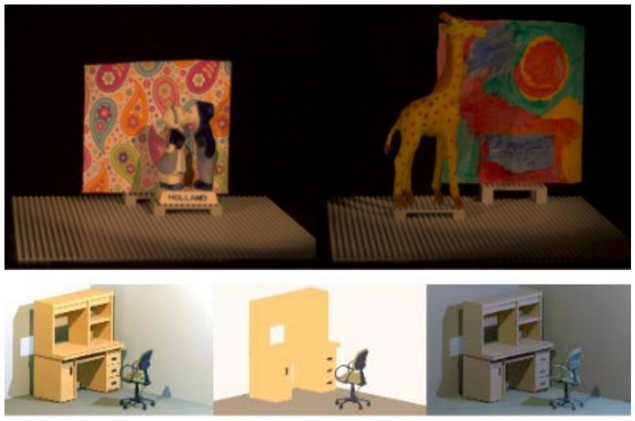
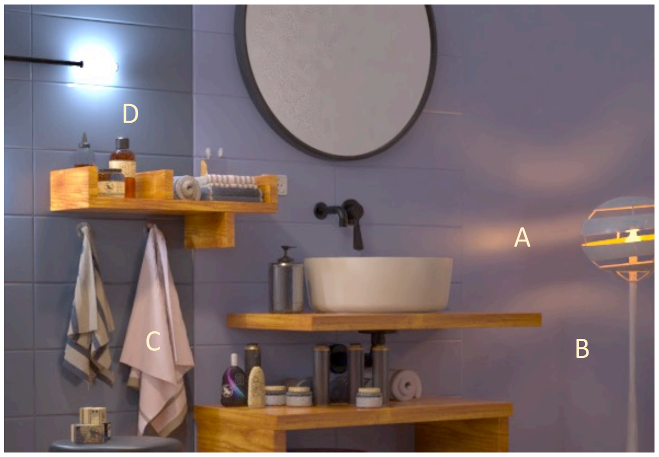


Fig. 1. Example images from the real image (top) and synthetic image (bottom) data sets of Beigpour et al. [18]



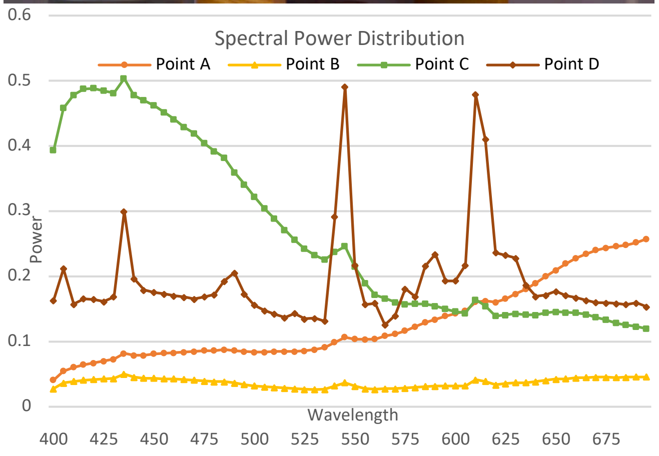


Fig. 2. (top) Example of a full-spectrum image (displayed here in sRGB) in which each pixel is represented by the spectral power distribution of the light reflected from the corresponding surface location. (bottom) The spectral power distributions of the reflected light from the four points indicated in the image.

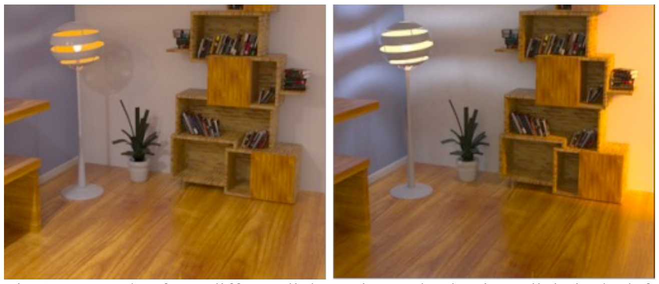


Fig. 3. Example of two different light settings. The dominant light in the left scene is out of sight and the lamp is very dim, while the dominant light in the right scene is a bright lamp along with sunset from a window (not visible). Images in the dataset are full-spectrum but shown here converted to sRGB.

number of strategically placed grey balls that are subsequently removed. In this case, the groundtruth illumination is available only for the grey ball locations and not pixel-wise. Beigpour et al. [18] developed a dataset of multi-illuminant images again in a laboratory setting with groundtruth data specified in terms of the surface colours under a specific white light. They built the dataset by very careful construction of each scene and camera setup. After capturing images of a scene under 20 different lighting conditions, the scene is spray painted grey and photographed again under the same 20 lighting conditions. This provides the necessary information for computing the groundtruth. The scenes are quite limited in their complexity, usually of a single object. The final 600 images in the dataset are of 5 scenes under 20 different illumination conditions, photographed using 6 cameras. In other words, for a given camera there are only 100 images of 5 scenes/objects. Fig. 1 shows two examples from that dataset.

In terms of synthesizing images of multi-illuminant scenes as done here, Beigpour et al. [18] synthesized images via Photon mapping using 6 bands. Their dataset is quite small, containing only 36 low resolution (480x270) images of 9 scenes under 4 different illumination conditions, 2 of which are more or less uniform in spectral power distribution. Hence, only 18 of these are truly multi-illuminant. Fig. 1 (bottom) shows an example of a scene under two different illumination conditions along with the ground truth (middle).

Spectral rendering has been used in applications such as the study of objects of cultural heritage [20]. However, with the exception of special cases such as the rendering of jewelry or prisms, spectral rendering is not widely used in the graphics industry, and existing tools are either incomplete or only focus on a particular purpose. The Mitsuba renderer [21] and Indigo renderer [22] both provide some support for spectral rendering; however, they require the user to specify fully the spectral information for each surface and illuminant, and they do not support complex textures. Textures, however, are very important for rendering photorealistic scenes. All the images in our dataset contain many textures, and so these renderers do not meet our requirements.

III. Background

Our goal is to create a dataset of multi-illuminant images that can be used for researchers interested in illumination estimation, color constancy and automatic white balancing in multi-illuminant scenes. Since it is difficult to measure the illumination incident at every point in a real scene, photorealistic rendering of synthetic scenes is used instead. As a byproduct of the rendering process, we are also able to provide stereo pairs, groundtruth depth data, and separate diffuse and specular components of the reflected light, all of which should be useful in the context of computer vision research related to 'intrinsic' image properties [23].

Blender Cycles [24] is used as the renderer. It is a free, open source, 3D graphics application for creating 3D models, visualizations, and animations. It only renders 3-channel, RGB images but since it is agnostic to the definition of R, G, and B we feed it spectral data and render the same scene multiple times to obtain a full-spectrum result. The typical application of Blender is to model and render 3D computer graphics images and animations using various techniques such as raytracing, radiosity, ambient occlusion and scanline rendering. Blender includes a 3D modeling environment and two original renders: Blender Internal and Blender Cycles. It can also be extended with external renderers such as Mitsuba [21] or Luxrender [25] to provide additional rendering features. Blender Cycles is designed to utilize multi-core CPUs and GPUs for rendering efficiency in both off-line and interactive rendering and is easily extensible via C++ and Python.

Blender Cycles is used here to render ray-traced stereo images of multi-illuminant scenes. The rendered images in the dataset are stored in OpenEXR [26] format. OpenEXR is a high dynamic-range (HDR) image file format developed by Industrial Light & Magic (ILM) for use in computer imaging applications. ILM released OpenEXR as free software, and Blender Cycles [24] has built-in support for it. OpenEXR provides an HDR "HALF" format based on 16-bit floating-point values. HALF format provides 1,024 values per color component per f-stop, over a total of 30 f-stops. In addition to the HALF data type, OpenEXR also supports several lossless compression methods such as PIZ [27] and ZIP [27], which can achieve lossless compression ratios of approximately 2:1. OpenEXR supports images containing an arbitrary number of channels, each with a different data type.

IV. Dataset Overview

The dataset is based on 3 different geometric scene models with each geometry rendered from 50 different viewpoints, and each viewpoint (i.e., camera location and orientation) rendered using six different light settings. Fig. 2 (top) shows an example full-spectrum image from the dataset, where each pixel is represented by the spectral power distribution of the light reflected from the corresponding surface location. The bottom shows the spectral power distributions of reflected light from the four points indicated in the image. Fig. 3 shows two different light settings from the same viewpoint, and Fig. 4 shows the "living room" geometry and four sample viewpoints inside the geometry. In total, the dataset consists of 1,000 full-spectrum stereo image pairs. This combination of images covers a mixture of indoor and outdoor lights.

Each image in the dataset has a resolution of $960x720$ pixels, with each pixel represented by a spectral power distribution. All the spectra in the dataset are sampled from 400nm to 695nm at 5nm increments, so an image array is $960x720x60$. Apart from the full-spectrum images, the dataset also provides groundtruth data specifying for the surface location corresponding to each pixel: (i) the surface percent surface spectral reflectance for its matte reflectance component; (ii) the surface percent surface spectral reflectance for its specular reflectance component; (iii) the spectral power distribution of the light incident upon it; (iv) its depth/distance to the camera; and, (v) its surface normal.

Images in the dataset are stored in OpenEXR Multilayer format, which allows multiple layers (channels) in a single image. In our case, each image consists of 60 channels with each channel storing the data for one wavelength represented in

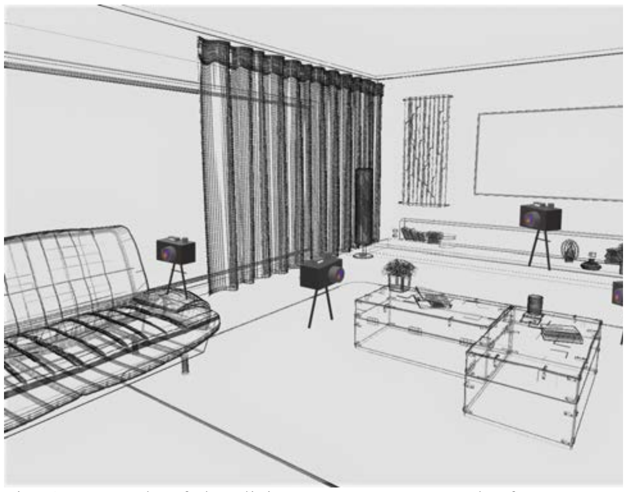


Fig. 4. Example of the "living room" geometry. The four cameras represent four different rendering viewpoints and view directions.

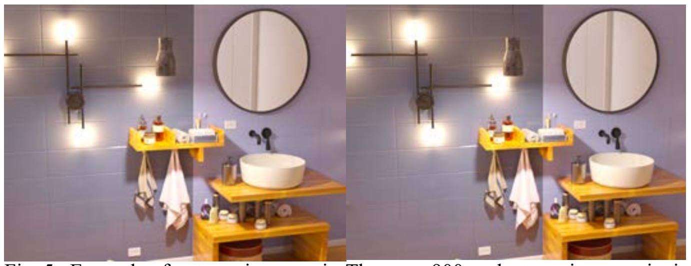


Fig. 5. Example of a stereo image pair. There are 900 such stereo image pairs in the dataset. The left and right images are rendered using same camera parameters (focal length, clipping etc.).

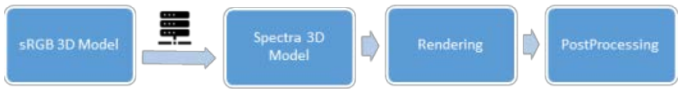


Fig. 6. The process used to generate a full-spectrum image. Begin with an sRGB 3D model, derive a full-spectrum model based on metameric lights and reflectances, use the model to render images wavelength-by-wavelength, and finally combine the renderings into a full-spectrum image.

OpenEXR's 16-bit floating-point format. The dataset is available online at https://www2.cs.sfu.ca/research/groups/Vision/data2/MIST-Dataset/ along with the source code and geometrical models for all the tools mentioned.

V. Methods

Fig. 6 provides an overview of the method for generating a full-spectrum image for the database. We start with an existing sRGB scene model and convert it to a full-spectrum scene model by replacing each sRGB value with a metameric spectrum (details below). Then we use standard Blender to render the full-spectrum scene model 3 wavelengths at a time, and finally combine the 20 separate renderings into a full-spectrum image. Although we could build a full-spectrum scene model from first principles, it is simpler to start with an existing sRGB scene model and modify it. The standard Blender model specifies a color for each surface triangle in the geometric model based on linear sRGB (i.e., the linear color space defined by the Rec. 709 chromaticities and D65 white point).

Similarly, lights are also specified by their color in linear sRGB. Blender approximates the physics of light reflecting off a (Lambertian) surface simply by multiplying the light's sRGB by the surface's sRGB. Although this method usually produces realistic looking images, they do not model the physics of reflection correctly. To model the physics correctly, the full spectra of the lights and the surfaces must be multiplied, not simply sRGB 3-tuples. The next section describes a method for deriving full-spectra for reflectances and lights from the linear sRGB specified in standard Blender 3D models.

A. Replacing sRGB with Metameric Spectra

To convert an sRGB-based computer graphics model into a full-spectrum one, we replace every surface's sRGB by a metameric (i.e., yields the same sRGB under the given light) percent surface spectral reflectance, and similarly every light's sRGB by a metameric spectral power distribution. This requires a large enough database of reflectance spectra that for every sRGB in the model the database will contain a surface reflectance that is approximately the same color under the D65 since sRGB is based on D65. Note that it is not necessary to match the colors exactly because our goal is simply to create a set of interesting scenes to render, not to precisely match any given scene model.

For reflectance spectra, we combine the 218 spectra of the Joensuu natural-color database provided by Parkkinen et al. [29] with the 1,269 spectra of the Munsell color system [28], the 350 spectra of the Krinov data set [30] and the spectra of 120 Dupont paint chips [31]. The Joensuu reflectances are sampled from 400nm to 700nm in 5nm increments, while the Munsell reflectances are sampled from 380nm to 800nm at 1nm increments. We combine them into a set of 1,957 spectra using the range 400nm to 695nm at a step of 5nm. Perhaps surprisingly, we found this number of spectra large enough to match most sRGB colors in the Blender models reasonably well so long as arbitrary uniform scalings of the reflectances are included as matches.

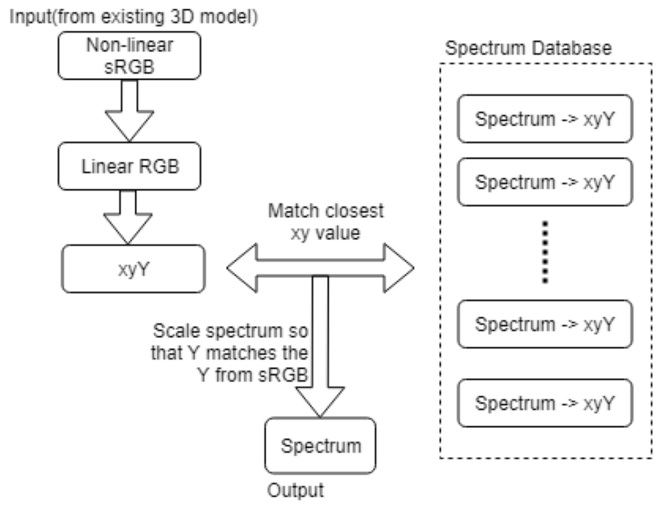


Fig. 7. Converting an sRGB to a metameric spectrum. First non-linear sRGB is transformed to linear RGB, and from RGB to xyY space based on CIE D65 as the illuminant. The xy-chromaticity component of the xyY is then used to find the spectrum in the database having the most similar chromaticity.

For illuminant spectra, we use several standard lights, such as D65, D50, illuminant A, illuminant F and the 318 IES [32] spectra along with spectra based on the D-series formula, and the Planckian locus formula over the range 2,000 to 10,000 Kelvin.

The process of finding a metameric spectrum for a given sRGB is based on matching in CIE xy chromaticity space. Non-linear sRGBs are linearized (Eq. 1) and then the linear sRGBs are converted to CIE XYZs based on the 3x3 matrix from the sRGB standard (Eq. 2). The XYZ of the reflectance spectra are computed based on D65 and the CIE 1931 $\bar{x},\ \bar{y},\ \bar{z}$ 2º-observer color matching functions. All XYZ are then transformed to xyY (Eq. 3) so that matches can be made in terms of xy-chromaticity coordinates. Once a match in chromaticity is found, the spectrum is then scaled so that the Y-values match as well. Fig. 7 summarizes the procedure for converting an sRGB to a metameric spectrum.

1) sRGB to linear RGB

For each sRGB channel, we apply the following transformation to compute the linear RGB values:

$$C_{linear} = \begin{cases} \dfrac{C_{srgb}}{12.92}, & C_{srgb} \le 0.04045 \\ \left(\dfrac{C_{srgb}+a}{1+a}\right)^{2.4}, & C_{srgb} > 0.04045 \end{cases} \tag{1}$$

2) Linear sRGB to xyY

Given a linear sRGB, its XYZ is computed by applying the following matrix [14]:

$$\begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9504 \end{bmatrix} \begin{bmatrix} R_{linear} \\ G_{linear} \\ B_{linear} \end{bmatrix} \tag{2}$$

The xyY value is derived by:

$$x = \frac{X}{X+Y+Z}, y = \frac{Y}{X+Y+Z}, Y = Y \tag{3}$$

3) Reflectance Spectra to xyY

Each reflectance spectrum $S(\lambda)$ in the database is converted to XYZ using the following formulae [15]:

$$X = \int_{\lambda} S(\lambda)I(\lambda)\bar{x}(\lambda)d\lambda$$

$$Y = \int_{\lambda} S(\lambda)I(\lambda)\bar{y}(\lambda)d\lambda$$

$$Z = \int_{\lambda} S(\lambda)I(\lambda)\bar{z}(\lambda)d\lambda \tag{4}$$

To match the sRGB standard, we use D65 as the illuminant $I(\lambda)$.

4) Match xyY values

Once both the reflectance spectrum under D65 and the given sRGB have been converted to xyY space, we simply find the closest one in the database based on Euclidean distance in xy. That most similar spectrum is then scaled by $\frac{Y_{srgb}}{Y_{spectrum}}$.

B. Other Details

1) Processing Textures

Texture mapping plays an essential role in modern photorealistic 3D scene rendering, so RGB texture maps need to be converted to full-spectrum texture maps. This is done simply by applying the sRGB-to-metameric-spectrum method described above to each pixel in the texture map. This texture conversion is only required for color textures. Other types of textures, such as surface normal textures and roughness textures, do not need conversion.

2) Efficiency

Converting an sRGB texture map to a full-spectrum texture map can be computationally expensive when the texture is large if every pixel is treated independently. The following strategies speed up the process significantly.

a. The xyY value of each spectrum in the database is computed once and cached so it need not be recomputed every time.
b. A KD-Tree is used for indexing the xyY values of the spectra, which reduces the time complexity of searching a database of k xyY values to $O(log\ k)$.
c. Results for each pixel are cached in a hash table so that when the same sRGB is encountered the result is available for reuse. This greatly improves the performance when processing textures containing repeated patterns.
d. The process is fully parallelizable since each pixel is independently evaluated.

3) Linear versus non-Linear sRGB

Blender uses linear sRGB internally during its computations but by default assumes the input to be nonlinear sRGB. To prevent Blender doing this conversion, the option "Color Management-Sequencer" is set to "Non-Color". Similarly, every texture node is marked as "Non-Color".

Modern 3D modeling tools have developed quite complicated tricks to render better looking images. Since most such tricks break physical laws, they need to be disabled when rendering physically correct images. To exclude any such tricks, a manual check of the 3D model is required. The most common two tricks are specified by the settings "Color Ramping" and "Random Color." These two features are typically used to add some visual variety, and they can generally be removed without changing the overall rendering result very much.

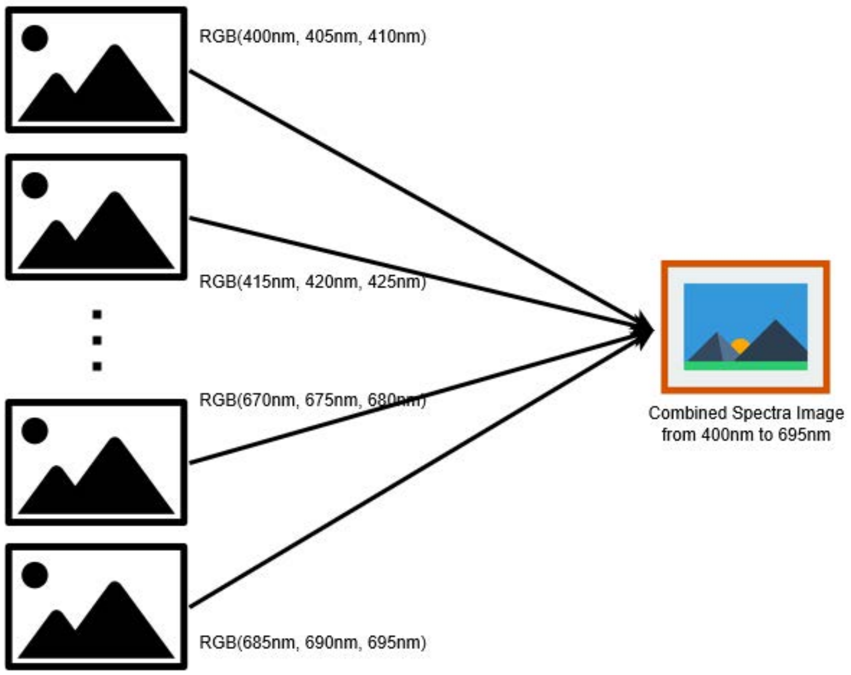


Fig. 8. Spectral rendering involves invoking Blender Cycles 20 times to render 3 wavelengths each time, after which all 60 renderings are combined into a single full-spectrum image.

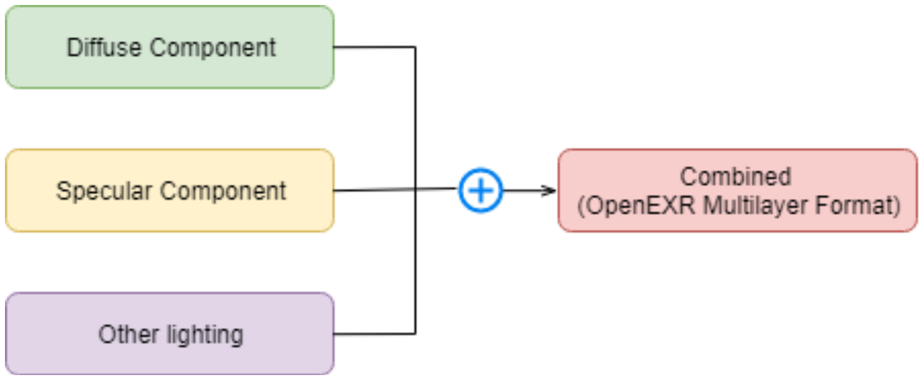


Fig. 9. Blender Cycles render layers. For each rendering, Blender Cycles carries out multiple rendering passes, then adds or multiplies them to generate the final combined image. The “Diffuse Component” corresponds to the reflectance ground truth.

C. Spectral Rendering via Blender Cycles

The strategy for using Blender Cycles for spectral rendering is to store the reflectance data sampled at three wavelength samples into the 3 RGB channels Blender Cycles uses and repeat this for multiple wavelength 3-tuples. Fig. 8 illustrates the spectral rendering process. In total, Blender Cycles is invoked 20 times resulting in renderings for each of 60 individual wavelengths. These renderings are then combined into a single full-spectrum image, with each pixel's spectrum sampled every 5nm from 400nm to 695nm (not the usual 700nm because of the 60 samples).

D. Ground truth and additional data

Blender Cycles renders the diffuse, specular and indirect lighting (i.e., interreflection) image components separately and stores them in OpenEXR Multilayer format, as shown in Fig. 9. For the final rendering, these channels are added together; however, in the database they are retained, along with the final rendering for use by researchers interested in the analysis of specularities and interreflections. In addition, the pixel-by-pixel groundtruth spectral reflectance is included in the database.

In addition to the full-spectrum renderings, the database also includes sRGB versions of the groundtruth reflectance, specular and interreflection components. These are derived from the spectral data by first converting to CIE XYZ based on the CIE 1931 $\overline{x},\ \overline{y},\ \overline{z}$ 2°-observer color matching functions, and then from XYZ to linear sRGB, and finally to non-linear sRGB. Conversion to linear sRGB is based on Eq. 5:

$$\begin{bmatrix} \mathrm{R_{linear}} \\ \mathrm{G_{linear}} \\ \mathrm{B_{linear}} \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} \mathrm{X_{D65}} \\ \mathrm{Y_{D65}} \\ \mathrm{Z_{D65}} \end{bmatrix} \qquad (5)$$

If the converted $R_{linear}, G_{linear}, B_{linear}$ is out of the sRGB [0,1] range it is simply clipped to [0, 1]. Conversion to nonlinear sRGB is via Eq. 6:

$$\gamma(\mu) = \begin{cases} 12.92\mu, & u < 0.0031308 \\ 1.055\mu^{\frac{1}{2.4}} - 0.055, & otherwise \end{cases} \qquad (6)$$

where μ is R, G or B.

The dataset also provides the depth map and the normal map associated with each viewpoint. These data can be similarly extracted from the render passes. In order to ensure that the depth data is comparable across all the images in the dataset, the values in the depth map are not independently scaled. Fig. 10 summaries this additional data.

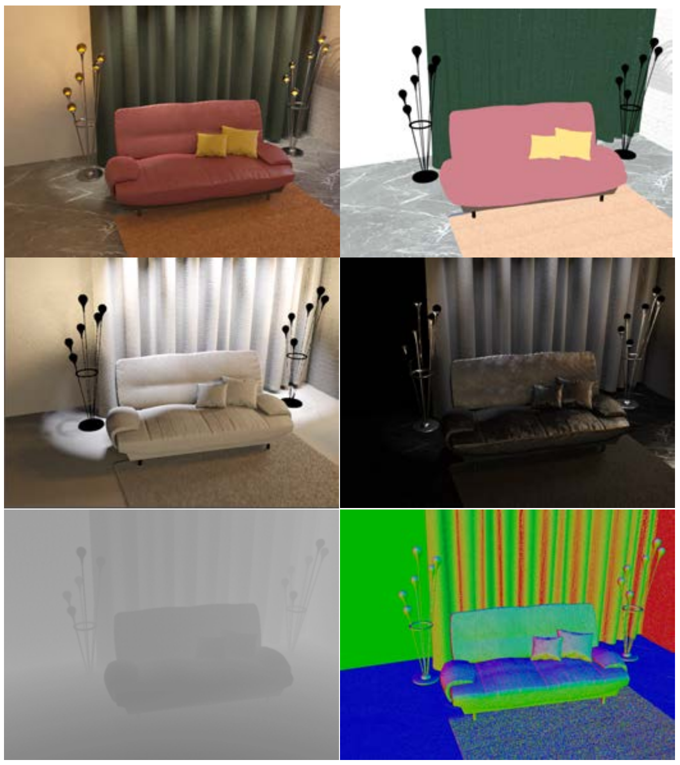


Fig. 10. Illustration of the dataset's data. First row: full-spectrum image (left) and its groundtruth reflectance image (right). Second row: reflected light's diffuse reflection component (left) and specular reflection component (right). Third row: depth map (left) and normal map (right). Images in the first two rows are represented by full spectra in the dataset but displayed in sRGB here.

Light Path Settings

Light Path Type	Max Bounces
Total	16
Diffuse	4
Glossy	10
Transparency	4
Transmission	15
Volume	4

Table 1. Light Path settings in the Blender. Extra ray-tracing resources in terms of the number of bounces are allocated to Glossy and Transmission to improve accuracy. "Total" is the limit on the number of bounces (e.g., Transmission followed by Glossy is still limited to 16).

E. Rendering Environment

Raytraced rendering is computationally intensive, often requiring 10 minutes of CPU time to render a single frame. Creation of 1,000 full-spectrum stereo image pairs requires the rendering of 40,000 individual frames, which could require at least half a year. The rendering is therefore performed on a High-Performance Computing Cluster named Cedar [33] which has a total of 58,416 CPU cores and is available free for academic research in Canada. Using Cedar, the dataset can be rendered in roughly four hours. Rather than interactively calling Blender, a Python script automates the rendering of different scenes for a variety of viewpoints and lighting conditions. The renderer is Blender 2.8 beta.

Table 1 lists the Blender settings used. Blender uses a Monte-Carlo simulation of the rays, and for the dataset at least 800 rays per pixel are used. Blender's Max Bounces parameter is set to 16. More bounces are allocated to the Glossy and Transmission light paths. These two light paths tend to include sharp edges that may be missed during Monte-Carlo sampling, so allocating more resources can significantly improve the accuracy for sharp-edged, shiny objects. Light Clamping is disabled to maintain a physically accurate result. For the same reason, both Reflective Caustics and Refractive Caustics are disabled.

F. Rendering Refraction

Refraction effects are interesting but since they are not that common in standard scenes the MIST database does not contain any. However, they are interesting to consider and can easily be generated using the full-spectrum extension of Blender Cycles described above. Since the index of refraction of a transparent object is a function of wavelength, traditional

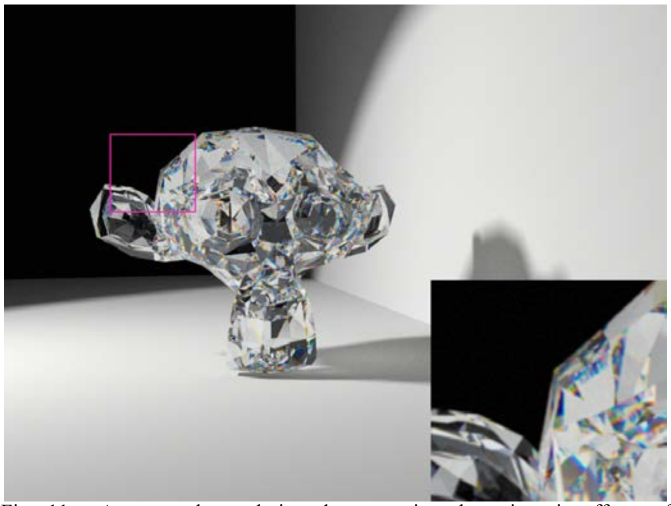


Fig. 11. An example rendering demonstrating the prismatic effects of refraction. The image was converted to sRGB for display. Since all pure spectral colors actually lie outside of the sRGB gamut, the sRGB are clipped to [0, 1].

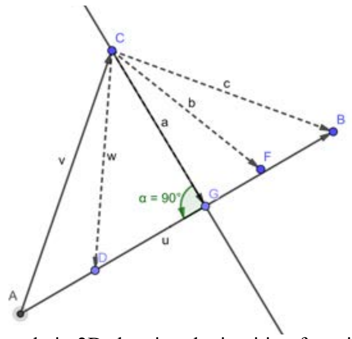


Fig. 12. An example in 2D showing the intuition for minimizing Eq. 10. using Eq. 11. For directions $k\overrightarrow{AB}$ and $\overrightarrow{AC}$, the difference between the two vectors is minimal for scaling by k such that $k\overrightarrow{AB} - \overrightarrow{AC}$ (i.e., $\overrightarrow{CG}$) is perpendicular to $\overrightarrow{AB}$.

RGB renderers do not render the effects of refraction fully. Using the proposed spectral rendering method, however, it is easy to specify the index of refraction for each wavelength (rather than just the three indices Blender normally uses) and thus closely model the true physics of refraction in the rendering. Fig. 11 shows a full-spectrum rendering in which the prismatic effects of the glass create rainbow colors. The process models the physics correctly; however, since pure spectral colors are outside the sRGB gamut the true sRGB need to be clipped to the sRGB gamut when displayed.

VI. Best single illuminant estimation

Hao et al. [34] found relatively poor performance when testing a small sample of illuminant estimation methods on the MIST dataset. While some of the methods were multi-illuminant, most provided only a single estimate of the illumination. For the single-illuminant case, it is interesting to consider what the best possible single-illuminant estimate might be in terms of its average error over an entire multi-illuminant image. The dataset provides the groundtruth illumination and reflectance on a pixel-by-pixel basis, which makes it possible to compute the single-illuminant chromaticity that minimizes the minimum average error in the estimated reflectance chromaticity. We call this optimal, single-illuminant illumination estimator the “Oracle” method. Oracle shows what the best any single-illuminant method can possibly do.

Oracle minimizes the squared Euclidean distance error between the *rg*-chromaticities of the corrected test image and the *rg-chromaticities* of the groundtruth image. The groundtruth images are generated by lighting the reflectance images with D65. The squared Euclidean distance error is used even though angular error is more common in the literature. Computing the minimal angular error requires an iterative optimization procedure and is thus slow. By minimizing the squared Euclidean distance error in *rg* color space, the optimal illuminant can be found in constant time using the following method.

The best single illuminant estimate is calculated in chromaticity space, with both the test and the groundtruth images in linear sRGB converted to *rg*-chromaticity space. The squared Euclidean Distance error of two given images is defined by:

$$Err = \sum_{x \in all\ pixels} \sum_{c \in r,g} (\kappa_c \rho_{xc} - \gamma_{xc})^2 \qquad (7)$$

where $\rho\ (\rho_{xr}, \rho_{xg})$ is the test image and $\gamma\ (\gamma_{xr}, \gamma_{xg})$ is the groundtruth chromaticity image. Rewriting Eq. 7 as:

$$\mathrm{Err} = \sum_{c \in r,g} Err_c \qquad (8)$$

where

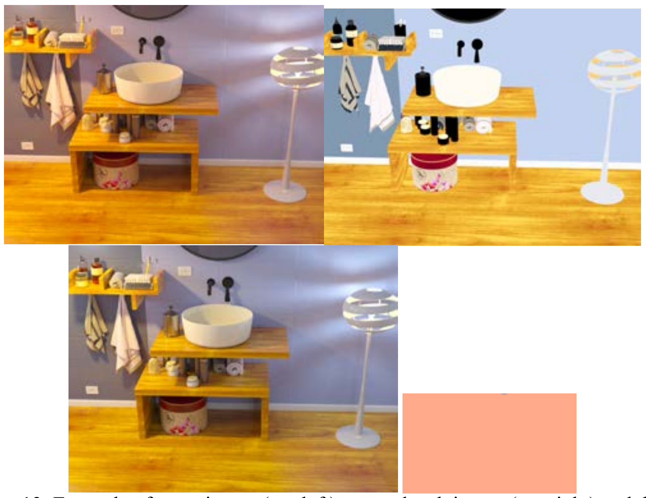


Fig. 13. Example of a test image (top left), groundtruth image (top right) and the corrected image (bottom left). The blue box (bottom right) of the second image shows the best single-illuminant estimate as calculated by the Oracle method.

$$\mathrm{Err_c} = \sum_{x \in all\ pixels} (\kappa_c \rho_{xc} - \gamma_{xc})^2 \tag{9}$$

it is clear that minimizing Err is equivalent to minimizing the Err_c separately. To minimize Err_c, we vectorize ρ_{xc} and γ_{xc} over x (note that κ_c is a scalar) and rewrite Eq. 9 as:

$$\mathrm{Err_c} = \|\kappa_c \overrightarrow{\rho_c} - \overrightarrow{\gamma_c}\|^2 \tag{10}$$

Clearly, $\kappa_c \overrightarrow{\rho_c} - \overrightarrow{\gamma_c}$ is minimal when it is perpendicular to $\overrightarrow{\rho_c}$, in other words, it is minimal when:

$$(\kappa_c \overrightarrow{\rho_c} - \overrightarrow{\gamma_c}) \cdot (\kappa_c \overrightarrow{\rho_c}) = 0 \tag{11}$$

Fig. 12 shows the intuition for this process. Rewriting for κ_c:

$$\kappa_c = \frac{\overrightarrow{\gamma_c} \cdot \overrightarrow{\rho_c}}{\overrightarrow{\rho_c} \cdot \overrightarrow{\rho_c}} \tag{12}$$

The chromaticity (κ_r, κ_g) is the best single-illuminant estimate. Fig. 13 shows a test image (top left), the corresponding groundtruth image (top right), and the corrected image based on the Oracle method (bottom left). The reddish-orange box (bottom right) shows the chromaticity of the best single illuminant. The dataset includes the Oracle-derived illuminant for every image in the dataset.

VI. Conclusions

A new dataset of physically correct, photorealistic, multi-spectral, stereo images with corresponding pixel-wise groundtruth data is described. The dataset is available for download from https://www2.cs.sfu.ca/research/groups/Vision/data2/MIST-Dataset/. This dataset will be useful for the testing of computer vision methods aimed at recovering 'intrinsic' image properties, and especially those directed at estimating the incident illumination in multi-illuminant scenes. Synthesizing images of complex scenes overcomes the crucial problem with real-image datasets, which is that the groundtruth illumination is measured at a very limited number of locations, usually just one. This is inadequate since the illumination tends to vary across almost every real scene. The MIST images are rendered using Blender Cycles extended from 3-channel to 60-channel rendering. To make use of texture mapping, the standard sRGB texture components are extended to full spectra by replacing them with metameric reflectance spectra. Using this technique makes it possible to synthesize images that are both photorealistic and physically accurate.

V. Acknowledgements

The financial support of the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged as is the access to the Compute Canada computer resources.

V. Author Biographies

Xiangpeng Hao completed his B.Sc. at Simon Fraser University in May 2020 and is currently pursuing his Ph.D. at the University of Wisconsin-Madison.

Brian Funt is Professor in the School of Computing Science at Simon Fraser University. His research is focused on color imaging, color constancy, color lighting, and metamerism.

References

[1] B. Funt, C. Vlad, and K. Barnard, “Learning Color Constancy,” *Proc. IS&T/SID Fourth Color Imaging Conference: Color Science, Systems and Applications,* pp. 58-60, Scottsdale, Arizona, Nov. 1996.

[2] D. Cheng, B. Price, S. Cohen, and M. S. Brown, “Effective Learning-Based Illuminant Estimation Using Simple Features,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1000–1008.

[3] G. D. Finlayson, “Corrected-Moment Illuminant Estimation,” *IEEE Int. Conf. Comput. Vis.*, 2013.

[4] G. D. Finlayson, S. D. Hordley, and P. M. Hubel, “Color by correlation: A simple, unifying framework for color constancy,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1209–1221, 2001.

[5] J. T. Barron and Y. T. Tsai, “Fast Fourier color constancy,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 6950–6958.

[6] J. van de Weijer, T. Gevers, and A. Gijsenij, “Edge-based color constancy,” *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2207–2214, Sep. 2007.

[7] G. Hemrit, G.D. Finlayson, A. Gijsenij, P. Gehler, S. Bianco, B. Funt, M. Drew, and L.Shi, “Rehabilitating the ColorChecker Dataset for Illuminant Estimation,” in Proc. *IS&T* Twenty-Sixth *Color and Imaging Conference*, 2018, vol. 2018, no. 1, pp. 350–353.

[8] L. Shi and B. Funt, “Re-processed Version of the Gehler Color Constancy Dataset of 568 Images,” *http://www.cs.sfu.ca/~colour/data/*.

[9] D. Cheng, D. K. Prasad, and M. S. Brown, “Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution,” in *Journal of the Optical Society of America A*, 2014, vol. 31, no. 5, p. 1049.

[10] F. Ciurea and B. Funt, “A large image database for color constancy research,” *Proc. IS&T. SID Eleventh Colour Imaging Conf.*, pp. 160–164, 2003.

[11] N. Banić and S. Lončarić, “Unsupervised Learning for Color Constancy,” in *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2018, pp. 181–188.

[12] B. Funt and L. Xu, “How multi-illuminant scenes affect automatic colour balancing,” in *Proc. AIC 2015 International Colour Association Conference*, 2015.

[13] S. Beigpour, C. Riess, J. Van De Weijer, and E. Angelopoulou, “Multi-illuminant estimation with conditional random fields,” *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 83–96, 2014.

[14] Funt, B., and Mosny, M., "Removing Outliers in Illumination Estimation,"*Proc. CIC’20 Twentieth IS&T Color Imaging Conference*, Los Angeles, Nov. 2012.

[15] A. Gijsenij, R. Lu, and T. Gevers, “Color constancy for multiple light sources,” *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 697–707, 2012.

[16] S. Bianco, C. Cusano, and R. Schettini, “Single and Multiple Illuminant Estimation Using Convolutional Neural Networks,” *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4347–4362, 2017.

[17] H. R. V. Joze and M. S. Drew, “Exemplar-based color constancy and multiple illumination,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 860–873, 2014.

[18] S. Beigpour, M. Ha, S. Kunz, A. Kolb, and V. Blanz, “Multi-view Multi-illuminant Intrinsic Dataset,” in *British Machine Vision Conference*, 2016, pp. 10.1-10.13.

[19] Beigpour, S., Serra, M., van de Weijer, J., Benavente, R., Vanrell, M., Penacchio, O. and Samaras, D. “Intrinsic image evaluation on synthetic complex scenes,” in *IEEE International Conference on Image Processing*, 2013, pp. 285–289.

[20] P. Callet, “Spectral Simulation for Cultural Heritage,” Colour Group (Great Britain) www.colour.org.uk, 2013.

[21] W. Jakob, “Mitsuba renderer,” *Http://www.Mitsuba-Renderer.Org*, 2010.

[22] “Indigo Renderer.” [Online]. Available: https://www.indigorenderer.com.

[23] H. Barrow, J. Tenenbaum, A. Hanson, and E. Riseman, “Recovering intrinsic scene characteristics from images,” *Comput. Vis. Syst.*, pp. 3–26, 1978.

[24] “Blender,” 2019. [Online]. Available: https://www.blender.org/.

[25] “LuxCoreRender.” [Online]. Available: https://luxcorerender.org.

[26] R. Bogart, F. Kainz, and D. Hess, “OpenEXR image file format,” *ACM SIGGRAPH, Sketches & Applications*. 2003.

[27] F. Kainz and R. Bogart, “Technical introduction to OpenEXR,” *Ind. Light magic*, pp. 1–15, 2009.

[28] D. Nickerson, “History of the Munsell Color System and its Scientific Application,” *J. Opt. Soc. Am.*, vol. 30, no. 12, p. 575, 1940.

[29] Parkkinen, J., Jaaskelainen, T. and Kuittinen, M.: "Spectral representation of color images," IEEE 9th International Conference on Pattern Recognition, Rome, Italy, 14-17 November, 1988, Vol. 2, pp. 933-935.

[30] E. L. Krinov, “Spectral Reflectance Properties of Natural Formations,” *Natl. Res. Counc. Canada*, 1947.

[31] M. J. Vrhel, R. Gershon, and L. S. Iwan, “Measurement and Analysis of Object Reflectance Spectra,” *Color Res. Appl.*, vol. 19, no. 1, pp. 4–9, 1994.

[32] Illuminating Engineering Society of North America, *IES Method for Evaluating Light Source Color Rendition*. 2015.

[33] “Cedar” [Online]. Available: https://docs.computecanada.ca/wiki/Cedar.

[34] Xiangpeng, H., Funt, B. and Jiang, H. "Evaluating Colour Constancy on the new MIST dataset of Multi-Illuminant Scenes," Proc. IS&T *CIC 27 Twenty-Seventh Color Imaging Conference*, Paris, Oct. 2019.