An online tool for displaying and processing spectral reflectance images

Philippe Colantoni[1], Jean-Baptiste Thomas[2], Mathieu Hebert[1], Alain Trémeau[1]
[1]Université de Lyon, Université Jean Monnet,
Laboratoire Hubert Curien, UMR 5516
Email: (philippe.colantoni,mathieu.hebert,alain.tremeau)@univ-st-etienne.fr
[2]NTNU - Norwegian University of Science and Technology
The Norwegian Colour and Visual Computing Laboratory
Email: jean.b.thomas@ntnu.no

***Abstract*—Modern web browsers allow to manipulate different types of multimedia files and can be adapted, with standardized technologies (WebAssembly, WebGL, etc.), to an ever-increasing number of contents. In this article, we describe how we were able to set up the necessary data structures and software techniques to enable web browsers to manipulate and visualize multi- and hyper-spectral images. A demonstrator, based on two images from a SpecimIQ hyperspectral sensor, is also presented as showcase.**

***Index Terms*—Multispectral imaging, Hyperspectral imaging, Image color analysis, World Wide Web, Browsers**

I. Introduction

Web browsers had become, over time, our main visualization tools for a very wide variety of content. It can appear as a dedicated software (Google Chrome, Mozilla Firefox, Apple Safari, etc.) or as a web view integrated into an application. Unfortunately, many types of content are not natively supported specifically for image type contents (only Jpeg, PNG, GIF and WebP formats in *RGB* are supported), even by the newest web browsers. This is the case for spectral reflectance images, mostly due to a lack of standard image format an issue which should be solved soon attended the recent initiatives, e.g. the one reported in [1].

Reflectance, also called reflectivity, is the fraction of light reflected from the surface of a material. It is defined as the ratio of reflected light flux to incident light flux. The reflectance of a surface generally varies according to the wavelength of the incident light. The curve representing the reflectance is a function of wavelength called a *spectral reflectance*.

It is possible, with specific sensors [2]–[4], [6]–[8] or by reconstruction [9], [10], to produce reflectance images where each pixel contains a spectral reflectance. These spectral images are useful to analyze the nature of the materials that compose an object.

There are several techniques to obtain spectral reflectance images. One way is to use dedicated spectral sensors (filter wheels [2], [3], Spectral Filter Arrays (SFA) [6], diffraction gratings [7], [8], liquid crystal tunable filter [4], etc.). It is also possible to perform spectral reconstructions calculated from one or several RGB images [9], [10].

Visualization and processing techniques are available for this kind of images [5], [11]–[13], but they are all available as dedicated software. IIPImage[1] is the only online tool we know that can handle spectral images but, at this time, it does not include any real-time color or spectral reconstructions.

In this article, we propose to show that it is possible to interact with such images directly through web pages viewed in a web browser. We propose an implementation with hyperspectral reflectance images acquired by a SpecimIQ camera. For this purpose:

- We perform a transformation and store the information adapted to usage scenarios, we also implemented a Web platform with accelerated calculations through the Graphics Processing Unit (GPU).
- We perform color (with color management) and spectral reconstructions optimized for implementation on GPUs to perform our calculations in real-time and therefore vary the parameters (virtual illuminant, luminance, wavelengths selection and mapping to *RGB*, etc.) for a better exploration of the spectral image information.
- We use an evolutive processing and visualization model based on custom shaders (dynamically built graphics shaders as in [14]).
- We use existing Web technologies: WebAssembly[2] for the implementation of complex algorithms and data structures, WebGL[3] to access the computing power of the GPU.

The operations that allow to create the different data structures required by our tools are described in Section II, our tools with their functionalities and their usages are described in Section III, the limits of our tools in Section IV, and, in Section V we present our demonstrator.

II. Data structures

A. Spectral reflectance images

The images are then multi- or hyper-spectral images, depending on the dimensionality and the nature of the images. The reflectance estimation is based on a calibration process based on a standard reference. This calibration process allows

[1]https://iipimage.sourceforge.io/
[2]https://webassembly.org/
[3]https://www.khronos.org/webgl/

978-1-7281-5686-6/19/$31.00 ©2019 IEEE
DOI 10.1109/SITIS.2019.00118

to produce a reflectance factor image which can be considered as reflectance image only in the case of Lambertian surfaces (therefore perfectly diffuse, this hypothesis is never fully verified but is assumed in this paper).

B. Image resolution and dimensionality

The very large variety of sensors and techniques to obtain spectral reflectance images generate a multitude of image types with various resolutions, numbers of spectral bands, and sensitivities.

A Silios CMS-C multispectral camera based on SFA allows to obtain images of size 1280×1024 pixels with 9 bands, between 430 and 700 nm, and 10-bit coded information (this kind of image requires to use specific demosaicing algorithm [5]). A HySpex VNIR-1800 camera scans lines of 1800 pixels with 186 bands, between 400 and 1000 nm, and 16-bit coded information using diffraction gratings; A SpecimIQ hyperspectral camera, produces with an integrated scanner that uses diffraction grating, a 512×512 pixels image with 202 bands, between 400 and 1000 nm, with 16-bit encoded information.

A HySpex VNIR-1800 camera placed on two translation axes X and Y, scanning 10 cm stripes, will generate for a painting of the size of Leonardo da Vinci's Mona Lisa (77×53 cm) an image of size 13860×9540 pixels with 182 double-byte coded strips which would represent 48.130GB of data. For the same painting, the multispectral camera Silios CMS-C will produce an image of 1.64 MB of data, with a much lower spatio-spectral resolution.

We see that the amount of information that needs to be manipulated to process and/or display such images can therefore be very large, making it difficult to handle them. It is therefore essential to have data structures adapted to their usage.

C. Hierarchical transformation of the information

a) Spatial resolution: We use a Gaussian pyramid that enables us to have a multi-resolution representation of our images. This representation lets us select the level of detail we want to manipulate/display [16]. Each level l of this pyramid has a resolution divided by 2 compared to level $l-1$, level $l=0$ containing the original image. Depending on the usage, it is sometimes interesting to decompose an image into tiles. This representation is particularly interesting when we want to visualize images in very high resolution (VHR) [22] from a server. This representation mode is particularly well suited for dynamic client/server visualization processes where the client only requests the subparts of the image that he/she needs for his/her process while keeping the still visible parts of the image.

b) Spectral dimension: The spectral information that we manipulate with these images is highly correlated, this allows to use different techniques to simplify the interaction with the huge data. We can consider different information hierarchies based on dimensionality reduction techniques [17]. For this article we implemented a global principal component analysis (PCA) [18]. An image with reflectance dimension of m can then be represented by an image of dimension k (with $k \leq m$), with minimum loss.

D. Metadata

In order to facilitate its manipulation, it is possible to add metadata to a complex and voluminous data structure. In our case, we have chosen to integrate metadata that quantify the quality of the spectral and color reconstructions according to the number of dimensions used as a result of the PCA. For this purpose we used the eigenvalues (sorted in descending order) and 2 metrics described in [20]: the ΔE_{00} [19] on color reconstructions and the root mean square error (RMS error) on spectral reconstructions. This allows us to produce a set of 5 indicators:

1) The sum of the k first eigenvalues (k varying from 1 to m).
2) The average ΔE_{00} values between the color reconstruction, with a D65 illuminant, of the pixels in the original image (encoded on m dimensions) and those reconstructed with k dimensions of the PCA, k varying from 1 to m.
3) The 99 percentile of the ΔE_{00} values between the color reconstruction, with an illuminant D65, of all pixels in the original image (encoded in m dimensions) and the pixels reconstructed with k dimensions of the PCA, k varying from 1 to m.
4) The average value of the root mean square error for all pixels in the original image (encoded on m dimensions) with the corresponding reconstructed pixels with k dimensions of the PCA, k varying from 1 to m.
5) The 99 percentile of the root mean square error for all pixels in the original image (encoded on m dimensions) with the corresponding reconstructed pixels with k dimensions of the PCA, k varying from 1 to m.

E. Usage scenarios

The benefits of the metadata are multiple because they allow to have a report on the analysis of the accuracy of the reconstruction according to the number of dimensions chosen. They also allow to define scenarios for the usage of the spectral images stored in this way. For example, we can define a colorimetric scenario based on the 99 percentile of the ΔE_{00} that will indicate the number of dimensions required to have a maximum value of 1.0 or a spectral and colorimetric scenario that defines the dimension necessary to have an average RMS error value below 0.003 and an average ΔE_{00} below 1.0.

For a given image, the chosen usage scenario permits to define the number of dimensions k' to be read on the available k. This allows limiting the network bandwidth required and the memory used while maintaining the digital quality of the reconstruction.

F. Data structure

The basic element of our data structure is the *tile* (see Figures 1). It has a fixed size $t \times t$ pixels. A tile contains *m PCA_Image* (see II-B) with values that are either floating numbers in single (4 bytes) or half (2 bytes) precision. The values of a tile are the ones of a spectral band at a position and level of our given pyramid.

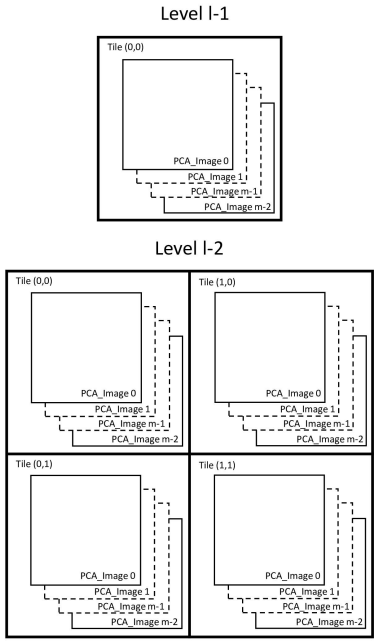

Fig. 1: The last two levels of the data structure (the top of the pyramid)

Standard formats, such as JPEG2000 or TIFF, can be used to store this kind of data structure, but unfortunately it is currently impossible to decode these files in a web browser. We therefore had to choose a simple proprietary format that was easy to decode.

The tiles are stored successively (see Figure 2), for a given image, in one or more files (several files can be used if the size of the structure is so large that it is not desired to handle it as a single large file). All metadata as well as data related to resolution, spectral dimension, PCA eigenvectors, pyramid size, coding of the information used (half or single precision) are stored in the file headers.

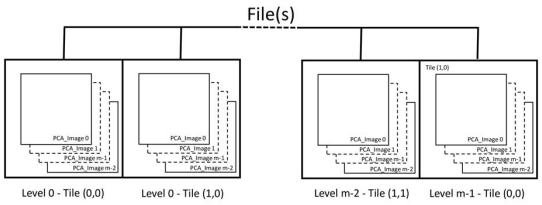

Fig. 2: Data storage in the file(s)

G. Calculations

The PCA and metadata values are calculated from the original image, with m spectral dimensions.

For each of the eigenvalues, sorted in descending order, we project each of the spectra contained in the pixels of the image using the corresponding eigenvector. Each component resulting from these projections generates an image that is used as basis for calculating a Gaussian pyramid that is decomposed into tiles of size $t \times t$ pixels, starting from the top left corner of the corresponding level in the pyramid, before being stored in the destination file(s).

The 5 indicators that constitute the metadata appear as 5 arrays of size m, the spectral dimension of the original image. The values of the indicators that form these arrays are, for a given index, calculated from the same spectral reconstruction. For the i-index of these tables, the reconstruction is calculated from the first i^{th} components of the PCA.

1) Test images: We made 2 image captures with a SpecimIQ which are used as test images for this article (see Figures 3 and 4).

PCA (float)	1	4	8	12	32	64
$sum(\lambda)$	0.8659	0.9964	0.9994	0.9998	0.9999	0.9999
$\overline{X}(\Delta E_{00})$	19.306	4.5692	0.2691	0.1061	0.01882	0.01359
$P_{99\%}\Delta E_{00}$	54.60	16.715	1.2437	0.4597	0.07398	0.05358
$\overline{X}(RMSE)$	0.1228	0.01808	0.00748	0.00472	0.00291	0.00186
$P_{99\%}$ RMSE	0.2518	0.06098	0.02416	0.00992	0.00440	0.00305
PCA (half)	1	4	8	12	32	64
$sum(\lambda)$	0.8659	0.9964	0.9994	0.9998	0.9999	0.9999
$\overline{X}(\Delta E_{00})$	19.306	4.5692	0.2696	0.1072	0.02205	0.01757
$P_{99\%}\Delta E_{00}$	54.609	16.715	1.2437	0.4598	0.08469	0.07138
$\overline{X}(RMSE)$	0.1228	0.01808	0.00748	0.00472	0.00291	0.00186
$P_{99\%}$ RMSE	0.2518	0.06098	0.02416	0.00992	0.00440	0.00305

TABLE I: Metadata results for test image 1

PCA (float)	1	4	8	12	32	64
$sum(\lambda)$	0.9195	0.9995	0.9998	0.9999	0.9999	0.9999
$\overline{X}(\Delta E_{00})$	22.951	1.5975	0.34817	0.19544	0.08743	0.05218
$P_{99\%}\Delta E_{00}$	45.696	6.3196	1.7967	0.9407	0.37329	0.2129
$\overline{X}(RMSE)$	0.1006	0.00677	0.00373	0.00332	0.00229	0.00156
$P_{99\%}$ RMSE	0.2400	0.02650	0.00741	0.00553	0.00397	0.00289
PCA (half)	1	4	8	12	32	64
$sum(\lambda)$	0.9195	0.9995	0.9998	0.9999	0.9999	0.9999
$\overline{X}(\Delta E_{00})$	22.951	1.5976	0.3485	0.1961	0.08908	0.05478
$P_{99\%}\Delta E_{00}$	45.701	6.3191	1.7970	0.9395	0.3744	0.2145
$\overline{X}(RMSE)$	0.1006	0.00677	0.00373	0.00332	0.00229	0.00157
$P_{99\%}$ RMSE	0.2400	0.02650	0.00741	0.00553	0.00397	0.00289

TABLE II: Metadata results for test image 2

2) Metadata calculation results: In tables I and II, we can see the values of the calculated metadata (with $k = 1, 4, 8, 12, 32, 64$) for the test images 1 and 2.

The values obtained indicate that it is not necessary to store the information with a single precision (float) encoding because the gain in terms of accuracy, compared to half-float, is negligible for a storage twice large. We also see that the gain is marginal for a k value higher than 32. For the two test images, 9 dimensions are enough to obtain a 99 percentile of the ΔE_{00} lower than 1.0.

H. Conversion tool

We currently have a converter tool that can handle multi-spectral images from the CRISATEL project as well as hyper-spectral images from the SpecimIQ. This tool, implemented in C++, is cross-platform (Mac OS, Linux and Windows) and can be modified to support other multi- or hyper-spectral

(a) Color reconstruction: with a D65 illuminant and a sRGB ICC profile, to be visualized on a sRGB monitor

(b) Spectral reconstruction: 650 nm assigned to the Red channel, 550 nm to the Green and 450 nm to the Blue channel

(c) Spectral reconstruction: 850 nm (IR) assigned to the Red channel, 550 nm to the Green and 450 nm to the Blue channel

Fig. 3: Specim IQ test image 1

(a) Spectral reconstruction: 450 nm assigned to the Red, Green and Blue channels

(b) Spectral reconstruction: 850 nm (IR) assigned to the Red, Green and Blue channels

(c) Color reconstruction: with a D65 illuminant and a sRGB ICC profile

Fig. 4: Specim IQ test image 2

image formats. It is also in charge of the metadata computation and we can, if we want, add new statistics (metadata) in our files (for this it will be necessary to reconvert all the original spectral images). This tool is optimized to handle very high resolution multi- and hyper-spectral images.

III. Web-based tool

In this section, we present the functionalities and implementation methods of our Web visualization tool. This tool is dedicated to process and display the images described in the previous section It has been designed to integrate basic and evolutive functionalities.

A. Basic functionalities

We integrate the following basic functionalities into our tool:

a) Color reconstruction: The purpose of color reconstruction is to define for each pixel of the image the *CIEXYZ* tri-stimulus values generated when illuminated by a virtual light L. Once reconstructed, the colors are displayed on the screen using a color management process. The numerical values for the pixel below the mouse cursor are displayed in a dedicated information frame at the bottom left of the display window. This information box displays the calculated *CIEXYZ* values as well as the corresponding *CIELAB* values (under illuminant L) and the *RGB* values calculated during the color management.

b) Spectral reconstruction: The purpose of spectral reconstruction is to generate by interpolation, for a given wavelength, a reflectance image. We provide the possibility to simultaneously display 3 reflectance images (for 3 given wavelengths) assigned to the R, G and B channels of the displayed image which gives us a false color visualization. The wavelengths associated with the R, G and B channels of the pixel below the mouse cursor are displayed in the information frame (the spectrum will be available soon).

c) Color management and out-of-gamut color display: Color management consists in a controlled transformation between the color representations of different devices. In

the context of a color reconstruction, the calculated *CIEXYZ* values are transformed into an *RGB* triplet with a complex transformation managed by the color management module (CMM). Since screens can only display a limited number of colors, we have also set up a specific display mechanism for the so-called "out of gamut" colors (not directly displayable on the screen) with a color code that assesses the distance between these colors and the gamut surface of the target screen. The gamut corresponds to the surface of the 3D shape containing all the colors displayable by a screen. When a color is out of the gamut, we bring it back to the gamut surface by using a gamut clipping technique. The out-of-gamut distance is then defined by the Euclidean distance in the *CIELAB* space between the original color and the one on the surface of the gamut.

d) Color mixing of reconstructions: It is possible to select the type of reconstruction that is displayed on the screen but also to combine them dynamically with a mixing ratio (between 0 and 1).

B. Implementation

Our implementation is mainly based on 2 Web technologies. WebAssembly allowed us to transpose our C++ library which manages our spectral data files (and the associated data structures) as well as our color management functions as modules that can be accessed directly through JavaScript. WebAssembly allows us to have execution times that are close to those we can have with native applications. WebGL 1.0 allowed us to access to the computing power of the graphics card used to execute the web browser, via graphic shaders, and to transfer the image data as textures that are also stored in the graphics card's memory for maximum efficiency.

The heterogeneity in the sources of the reflectance spectra that we have to process (cf. subsection II-A) requires us to standardize the sampling of these spectra. Our color and spectral reconstructions are based on tabulated data available with a $1\ nm$ spectral sampling, so we have chosen to use this sampling. To do this, all the reflectance spectra we use for our pre-calculations are resampled using an Akima interpolation [15].

a) Color reconstruction: The computation of the color reconstruction for a given reflectance $R(\lambda)$ and a light $L(\lambda)$ is performed in the color space $CIEXYZ$ 1931 or 1964 with the following formula:

$$\begin{cases} X = \sum_{\lambda=400}^{\lambda=760} \overline{x}(\lambda)R(\lambda)L(\lambda) \\ Y = \sum_{\lambda=400}^{\lambda=760} \overline{y}(\lambda)R(\lambda)L(\lambda) \\ Z = \sum_{\lambda=400}^{\lambda=760} \overline{z}(\lambda)R(\lambda)L(\lambda) \end{cases}$$

where $\overline{x}$, $\overline{y}$ and $\overline{z}$ are the colorimetric functions of the *CIE* 1931 or 1964 reference observer. In the case of a representation of spectral data after the *PCA* this color reconstruction is simplified because we only have to pre-calculate with the previous formula the *CIEXYZ* values for each of the eigenvectors used. The *CIEXYZ* values for a given pixel is then, if we use k dimension for reconstruction:

$$\sum_{d=0}^{d=k-1} PCA_d(i,j) * XYZ_{eigenvector_d}$$

The transformation of the tri-stimuli so obtained into *RGB* values dedicated for the screen is then done through a color management process that requires to transform these *CIEXYZ* values into *CIELAB* values that will be used as input for interpolation process made with the 3D Look Up Table (LUT) defined bellow.

b) Spectral reconstruction: To compute, at a given wavelength, a reflectance image we use the tabulated values coming from the interpolated eigenvectors of the PCA. For a given wavelength, the reflectance value of a pixel is computed with the following formula:

$$\sum_{d=0}^{d=k-1} PCA_d(i,j) * eigenvector_d[\lambda]$$

c) Color management and out-of-gamut color display: Color management is performed using a pre-calculated 3D LUT [23] with a function from our WebCMM (our JavaScript color management module) based on a calibration technique described in [24]. The implementation of this technique was made possible by the creation of a color characterization tool dedicated to Web browsers that we have described in [25]. The computation of the out-of-gamut distance is also performed with this 3D LUT which contains *RGBA* color vectors where the component A is the out-of-gamut distance. The *CIELAB* values that has been calculated in the color reconstruction are used as input for the computation, by a tri-linear interpolation inside the 3D LUT, of the *RGB* color and the A value.

The spectral image is associated to a web page through a JavaScript initialization function with parameters that define the resolution level l used (the level inside the pyramid stored in the file) as well as the dimension k of the PCA used (on the m dimensions available). This dimension k can be defined as a constant or computed by a function that we provide. This function will determine the value of k according to the usage scenario wanted by the user and the metadata embedded in the file. This function downloads all the necessary data from a server into the browser's memory, which then transfers them inside 4-dimensional textures (*RGBA* encoded) either in single precision (float) or in half precision (half float) format (depending on the encoding of the information inside the file). The use of *RGBA* textures allows us to package our k dimensions inside $(k-1)/4+1$ textures. The limited number of texture units available in a graphical shader requires us to use this packing method (see IV-A).

All these calculations, as well as the color mixing, are integrated into a single graphical shader, a fragment shader, that is in charge of the source spectral image processing process. The source code of this shader is dynamically generated during the spectral image initialization in order to be adapted to the used k dimensions of the PCA (and to overcome some WebGL 1.0 limitations). The shader is then compiled by WebGL to be directly executable by the GPU of the graphics card. This shader, which is executed on all pixels of the source spectral image, generates an RGB image as a new WebGL texture (which have the same size as the source spectral image at the resolution level l), which is then displayed in our web

page (with the possibility to zoom in) through the JavaScript Three.js framework[4].

C. Custom shader

In the previous subsection, all of the processing/visualization is performed in a single graphical shader with a dynamically created source code. It is therefore possible to transform this shader's code in order to make it perform other processing/visualizations. It is the concept of custom shader [14] which we implement on the basis of a new function which will replace the one which implements the basic treatment/visualization functions of our tool. Custom shaders make it possible to implement treatment/visualization processes adapted to very specific uses of spectral images (medical, heritage culture, etc.)

IV. Analysis

A. Limitations

For WebGL 1.0:

- Maximum texture size which is generally 16384×16384 and therefore, the images that we can process / view.
- Number of textures usable in a shader which is generally 16, this limits the dimension k that we can use up to 64 (16×4 with our packing method).
- Number of instructions that a shader can execute which may limit the capabilities of the custom shaders that will be produced.

For WebAssembly:

- Compatibility issues with some Web browsers, mainly on smartphones and tablets.

B. Performance

For an image of the SpecimIQ camera with $k = 20$, visualized in Firefox under Windows 10 on a PC with an Intel Core i7-6700K processor and a GeForce RTX 2800 graphics card, we have the following execution times:

- WebCMM initialization (does not depend on the resolution of the source image): 815 ms
- Reading the image: 67 ms
- Image processing: 0.025 ms

With the same image displayed in Firefox under MacOS 10.14.5 on an Intel Core i7-4980HQ processor with an integrated Intel Iris Pro graphics card we have:

- WebCMM initialization (does not depend on the resolution of the source image): 950 ms
- Reading the image: 72 ms
- Image processing: 0.16 ms

V. Demonstrator

A demonstrator is available at the following address: https://www.couleur.org/articles/SITIS2019-WAI/ (with a mirror site here https://ipem.univ-st-etienne.fr/articles/SITIS2019-WAI/).

[4]https://threejs.org

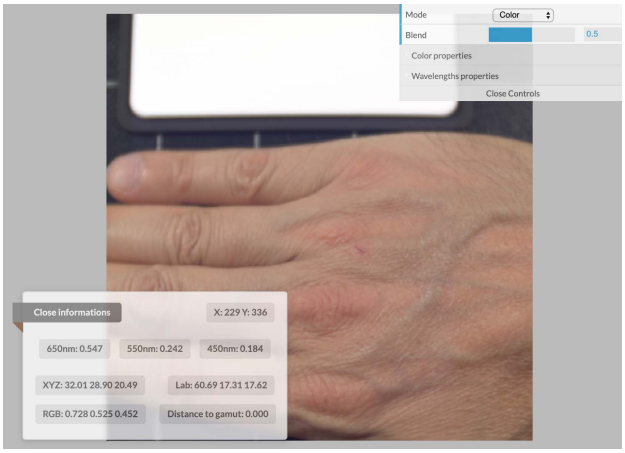

Fig. 5: Spectral image view window full interface

It allows:

- the visualization of the two test images presented in the section II-G1 with several initial configurations where we can change:
  - The number of dimensions used.
  - The testing of color management settings.
- to test 2 custom shaders dedicated:
  - For color segmentation.
  - For highlighting spectral properties in the infrared of an image during its color reconstruction.

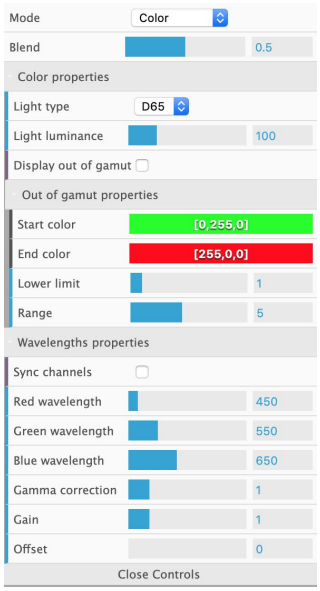

Fig. 6: Control panel

Each of the links presented on the home page of our demonstrator opens a new tab that displays the corresponding image with a classic or custom shader-based view. All spectral image view windows (see Figure 5) have a control panel that

allows changing many display settings in real-time (see Figure 6).

Only this demonstrator is currently available. The C++ code of our conversion tool as well as the one used for our Web tool is not available, for the moment, in open source. In the coming months, we will evaluate the possibility of making them available.

VI. Conclusion and perspectives

In this article, we have presented an innovative web tool for interacting and visualizing spectral reflectance images transformed in a dedicated data format optimized for this purpose. This data format, which includes metadata describing the quality of the color and spectral reconstructions of our images, allows us to precisely define, based on usage scenarios, the dimensionality of the information we have to use and thus limit the quantity of data we have to download.

The use of advanced Web technologies such as WebAssembly and WebGL have enabled to achieve performance levels close to those we can achieve with native applications. The integration around a JavaScript framework that offers basic functionalities (color and spectral reconstructions, color management, visualization of out of gamut, blending) also allows to propose a simplified process for the implementation of new processing techniques and visualizations based on custom shader concept. A demonstrator has been placed online to present the result of this work through two images obtained with a SpecimIQ hyperspectral camera.

The framework can be extended to support new kinds of spectral reflectance images. In particular, we will focus on video streams from spectral cameras equipped with spectral Bayer filters (such as the Silios CMC-C). Currently built to allow interaction with only one image per web page we also want it to be able to use several images simultaneously within the same web page. This last point is essential for its transformation into a more complete tool for visualizing these images in a 3D environment accessible via augmented reality devices (smartphones and tablets), mixed reality devices (Magic Leap One, Hololens 2) and virtual reality headsets thought WebVR and WebXR (when this web standard will be available) this will call to multi-angle measurements and relighting.

Acknowledgments

This work was supported by the *Fondation de l'Université Jean Monnet de Saint-Étienne*.

References

[1] Multispectral Image Formats, CIE 223:2017, ISBN: 978-3-902842-10-7, CIE Division 8, http://www.cie.co.at/publications/multispectral-image-formats

[2] J. Brauers, N. Schulte and T. Aach, "Multispectral Filter-Wheel Cameras: Geometric Distortion Model and Compensation Algorithms," in IEEE Transactions on Image Processing, vol. 17, no. 12, pp. 2368-2380, Dec. 2008.

[3] A. Ribés, H. Brettel, F. J. M. Schmitt, H. Liang, J. Cupitt, D. Saunders, Color and Multispectral Imaging with the CRISATEL Multispectral System, PICS, 2003.

[4] J.Y. Hardeberg , F. Schmitt, H. Brettel, Multispectral color image capture using a liquid crystal tunable filter. Optical Engineering 41(10), 25322548, 2002.

[5] J.B.Thomas, Multispectral imaging for computer vision. Signal and Image Processing. Universit de Bourgogne, Franche-Comt, 2018.

[6] P.J. Lapray, X. Wang, J.B. Thomas, P. Gouton, Multispectral Filter Arrays: Recent Advances and Practical Implementation, Sensors, 2014, vol. 14, 11, pp 21626.

[7] C. Cucci, J. K. Delaney, M. Picollo, "Reflectance Hyperspectral Imaging for Investigation of Works of Art: Old Master Paintings and Illuminated Manuscripts", Acc. Chem. Res., vol. 49, pp. 2070-2079, 2016.

[8] J. Behmann, K. Acebron, D. Emin, S. Bennertz, S. Matsubara, S. Thomas, D. Bohnenkamp, M. T. Kuska, J. Jussila, H. Salo, A.K. Mahlein, U. Rascher, Specim IQ: Evaluation of a New, Miniaturized Handheld Hyperspectral Camera and Its Application for Plant Phenotyping and Disease Detection. Sensors 18(2): 441 (2018)

[9] N. Akhtar, M. Ajmal, Hyperspectral recovery from RGB images using Gaussian Processes, IEEE transactions on pattern analysis and machine intelligence, 2018.

[10] R. M.H. Nguyen, D. K. Prasad, M. S. Brown, Training-Based Spectral Reconstruction from a Single RGB Image, Lecture Notes in Computer Science ; 8695 ; 186-201 Computer Vision, ECCV, European Conference on Computer Vision, 13, 2014.

[11] N. Hashimoto, Y. Murakami, P. A. Bautista, M. Yamaguchi, T. Obi, N. Ohyama, K. Uto, Y. Kosugi, Multispectral image enhancement for effective visualization, Opt. Express 19, 9315-9329, 2011.

[12] G. Poldera, G. W.A.M. van der Heijdena, Proceedings of the SPIE, Volume 4553, p. 132-137, 2001.

[13] M. Vilaseca, J. Pujol, M. Arjona, F.M. Martinez-Verdu, Color visualisation system for near-infrared multispectral images. Journal of Imaging Science and Technology pp. 246-255(3), 2005

[14] F. RoBler, R. P. Botchen, T. Ertl, Dynamic Shader Generation for Flexible Multi-Volume Visualization, 2008 IEEE Pacific Visualization Symposium, Kyoto, 2008, pp. 17-24.

[15] H. Akima, A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures, Journal of the ACM, vol. 17, no. 4, pp. 589-602, 1970.

[16] B.K. Choudhary, N. KumarSinha, P. Shanker, Pyramid Method in Image Processing, Journal of Information Systems and Communication ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, 2012, pp.- 269-273.

[17] C.O. Sánchez Sorzano, J. Vargas, A.D. Pascual-Montano, A survey of dimensionality reduction techniques, ArXiv, abs/1403.2877, 2014.

[18] A. Agarwal, T. El-Ghazawi, H. El-Askary, J. Le-Moigne, Efficient Hierarchical-PCA Dimension Reduction for Hyperspectral Imagery, 2007 IEEE International Symposium on Signal Processing and Information Technology, Giza, 2007, pp. 353-356.

[19] W. Mokrzycki, M. Tatol, Color difference delta EA survey. Mach Graph Vis 20:383-411, 2011.

[20] F. H. Imai, M. R. Rosen, R. S. Berns, Comparative Study of Metrics for Spectral Match Quality, Proceedings of the First European Conference on Colour in Graphics, Imaging and Vision, (Poitiers, France), 492-496 (2002).

[21] F. H. Imai, R. S. Berns, D. Tzeng, A comparative analysis of spectral reflectance estimated in various spaces using a trichromatic camera system, Journal of Imaging Science and Technology. 44 280-287 (2000).

[22] K. Martinez, J. Cupitt, S. Perrya, High resolution colorimetric image browsing on the Web, Computer Networks and ISDN Systems, Volume 30, Issues 17, April 1998, Pages 399-405

[23] P. Colantoni, J.B. Thomas, A Color Management Process for Real Time Color Reconstruction of Multispectral Images, In Arnt-Borre Salberg, Jon Yngve Hardeberg, and Robert Jenssen, editors, Lecture Notes in Computer Science, volume 5575 of 16th Scandinavian Conference, SCIA, 2009.

[24] P. Colantoni, J.B. Thomas, J.Y. Hardeberg, I. Foucherot, P. Gouton. High-end colorimetric display characterization using an adaptive training set. The Journal of the Society for Information Display, 19:520-530, 2011.

[25] P. Colantoni, A. Trèmeau,Web Browsers Colorimetric Characterization, CCIW'2019, Chiba, February 2019, LNCS Vol. 11418, pp 145-161.