Web technologies enable agile color management

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

***Abstract*—With the number of display technologies, cameras, operating systems and software solutions, one of the only technologies that is compatible across this diversity is the web browser. We propose to show that the technologies now present in web browsers allow an indepent management of the color information on large variety of devices. For this purpose we introduce the basic concepts of color management and then we show how to implement them with WebAssembly and WebGL by introducing the concept of WebCMM. A WebCMM adapted for the color management of HTML elements in 2D, 3D but also for virtual environments. Finally, we present how we can implement this WebCMM for a real case of color workflow implemented in a demonstrator web page.**

***Index Terms*—Color management, World Wide Web, Browsers**

I. Introduction

It is a well-known fact that different color imaging devices process color information in different ways. For instance, it is pretty obvious that a printer and a display employ very different processes to produce the desired colors. But also within one class of devices, e.g. displays, there are significant differences, even between displays based on the same technology (e.g. LCD). Therefore, achieving consistent reproduction of colour images in heterogenous systems (such as the web) has been an important research challenge for decades [1], [2]. Thanks mainly to the invention and implementation of the concept of color management [3]–[5], in which every imaging device must be characterized colorimetrically, color consistency may be achieved trough adapted color space transformations. Most of the recent work on display colorimetric calibration concerns the calibration of HDR displays [6], which is not considered in this work.

Mainly based on ICC profiles for general purpose and graphic arts industry defined by the International Color Consortium[1], it has evolved with iccMAX for spectral and appearance management or with ACES for digital cinema[2]. It uses Color Management Modules (CMM) implemented in the operating system (accessible to software running on this operating system) or in specific software (e.g. Photoshop) in order to do the necessary color transformations.

This is limited in the sense that:

- Different CMM handle the same information (ICC profiles for example) in different ways across operating systems.
- Some software has no link to color management (e.g. most of the pdf readers except Adobe Acrobat).
- Some software may have reduced connection to color management (e.g. Google Chrome and Mozilla Firefox).
- The color management is only applied to static calibrated images and not to the full work space.
- It is not designed to work with 3D or virtual reality (VR) rendering.

On the other hand, web browsers are usually compatible across operating systems, and can read several types of files via dedicated applications (incl. pdf). Web browsers can work with 3D (WebGL) and VR (WebVR) rendering, and simplify communication through the stream of video, sharing screen and video-conferencing systems (WebRTC).

We propose here a color management solution based on web technologies, that embed all those features into a single HTML element. This impacts the way we use color management today and create opportunities to extend the control of colors in extended reality (mixed, augmented or virtual) and related visual technologies.

People and applications benefiting from this potential are web developers or hybrid application developers, e.g. use web technologies for color applications and image rendering. It is also useful for web designers in the context of marketing or creation of visual content, e.g. on-line shopping or advertisement, appearance, brand-colors. And it is extremely useful for researchers working on color and vision, e.g. tools for colorimetric image rendering in psychological studies and computer graphics.

Two other contributions of this work are 1-exchange between the color imaging and the web communities. 2-the potential to keep the stimulus under control while using VR headset, compared to only use a rendering engine, such as Unreal Engine [7].

In the following we describe the basic concepts of color management in Section II and in Section III the web technologies that we used in order to make an implementation inside a modern web browser. Section IV describes accurately an experimental demonstrator that is associated to this article, with the parameters, the results it achieves and its performance.

[1]http://color.org/
[2]https://www.oscars.org/science-technology/sci-tech-projects/aces

978-1-7281-5686-6/19/$31.00 ©2019 IEEE
DOI 10.1109/SITIS.2019.00057

II. Color management

A. Color characterization

Device characterization [8], [9] can be defined as the process of how a device reproduces or reacts to colors. The result is typically stored as a file that can be an ICC profile or raw color measurement data (in CxF file format or inside an ICCMax Profile). This type of file does not allow direct color modification, it must be associated with a dedicated system or application's color management module as well as with another device's characterization file to allow color transformation. Knowing the characteristics of the two devices makes possible to transfer the colors from one to the other. This characterization process enables determining the range of colors that can be produced by an output device (such as a monitor or a printer) or discernible by an input device (such as a camera) defined as color gamuts.

B. Connection color spaces

1) Profile Connection Space: The profile connection space (PCS) is used to describe a device in a CIE reference space (with the 1931 standard colorimetric observer). This standard color space is the interface that ensures an unambiguous connection between input and output profiles. If the input and output color transformations are based on the same PCS definition, even if they are created independently, then they can be arbitrarily matched.

The default measurement parameters for the profile connection space and all other color spaces defined in this specification are based on the ISO 13655 standard, "Graphic technology - Spectral measurement and colorimetric computation for graphic arts images". Essentially, it requires to use a D50 standard illuminant and the 1931 CIE standard colorimetric observer (and for graphics arts and photography print viewing environment an illumination level of 500 lux), this PCS can be either the *CIEXYZ* or the *CIELAB*. Inside a PCS a color gamut appears as a 3D shape. All the colors outside this shape are considered as out of gamut (unreproducible or undistinguishable).

2) Target color space: For research purpose, we also define a Target Color Space (TCS). The TCS is a custom variation of the profile connection space. It is based on *CIELAB* (D50 CIE 31 or 64 standard observer) but adapted to a sampling method based on non-Euclidean color differences. Different sampling strategies can be used depending the sampling distance considered [10] (e.g. ΔE_{76}, ΔE_{94}, ΔE_{CMC}, ΔE_{BFD} or ΔE_{00}). By using use such samplings we can redefine the volume properties of the different colors gamuts (see Figure 1) and thus offer new settings for the gamut mapping techniques that we will present in the next section.

C. Gamut mapping

One of the goals of the color management is to let people transform colors from one device to another while keeping some of their characteristics. As mentioned above, each device has its own ability to reproduce or distinguish colors. The transformation from one device to another is therefore not always possible. This is the reason why this process usually involves a gamut mapping. This gamut mapping is, for most of the different techniques available [11], based on 3D geometric transformations allowing to switch from a source gamut to a destination gamut. The application of these techniques is traditionally done in the PCS, in our case we propose to apply them in the TCS that has been chosen. For a same gamut mapping technique, the choice of TCS modifies significantly the shapes of the gamuts (see Figure 1) and so the result that we obtain.

D. Rendering intent

Rendering intents are linked to ICC profile which can contain data associated to each rendering intent provided by the corresponding ICC file. They allow the Color Management Module (see next section) to perform different color transformations for a same device according the usage scenario wanted by the user and, of course the destination device. The 4 available rendering intents (Absolute colorimetric, Relative colorimetric, Perceptual and Saturation) can be seen as predefined transformations which can perform color transformations similar to a gamut mapping.

E. Color management module

The CMM makes possible to perform all the calculations necessary to transform colors: from a device to the TCS or PCS (forward transformation), from the TCS to the device (backward transform) and a gamut mapping.

III. Web implementation

A. Color characterization

Traditional color characterization methods use software tools, which can drive color measurement instruments, executed directly on the computer that is connected to the screen to be characterized. Then the obtained profiles are registered to the operating system or assigned to the software that performs the color management. Unfortunately, the great heterogeneity of systems that can execute a web browser makes impossible to use this kind of method.

In [16], we presented a method portable across any operating system supporting a web browser. This method which integrates an algorithm that allows to dynamically select color patches according to several criteria has been implemented inside a software that controls a colorimeter running on a computer using the same local network. This software is also a web server able to send the selected colors to a mobile device, a computer or a VR headset running the web browser.

B. Color management module

Because the web browser handles exclusively *RGB* data, the forward and backward transformation models correspond to the following:

- $RGB_{src} \rightarrow XYZ_{src} \rightarrow L^*a^*b^* \rightarrow TCS$
- $TCS \rightarrow L^*a^*b^* \rightarrow XYZ_{dst} \rightarrow RGB_{dest}$

The transformations used in our demonstrator are described in [17]. The forward transformation is based on polyharmonic

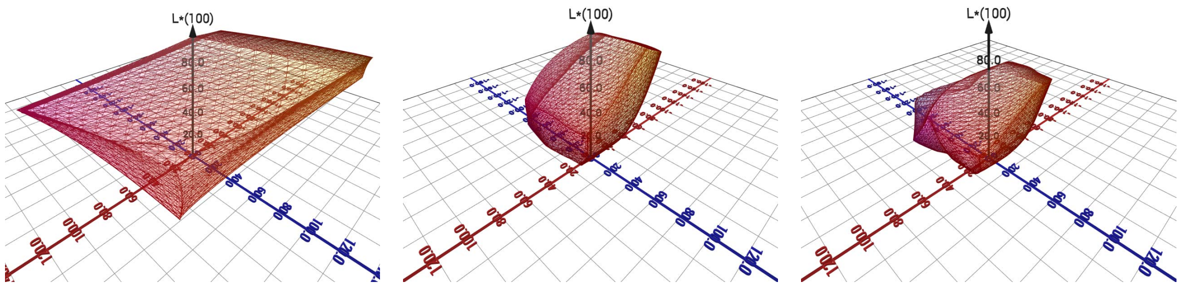


Fig. 1: Color gamut of an Apple iPad 2 visualized in 3 custom TCS (based on ΔE_{76}, ΔE_{94} and ΔE_{00})

splines (a subset of the Radial Basis Functions that can be used for interpolating or approximating arbitrarily distributed data) and the backward transformation (or inverse transformation) is based on a tetrahedral interpolation [15].

In our case, these transformations written in C++ are not linked to any external software module. This process is therefore performed by our code independently of any function of the system on which it is performed. For this reason, we were able to generate a particular version of our code entirely compiled in WebAssembly[3]. WebAssembly, or wasm, is a low-level binary programming language for developing applications in web browsers. Since WebAssembly only specifies a low-level language, bytecode is usually produced by compiling a higher-level language. Supported languages include C and C++, compiled with Emscripten[4]. The transformation of our code into wasm allowed us to define interface functions in JavaScript. These functions now allow us to run color management systems within web browsers using standard html pages. We therefore have a color management module dedicated to the Web (a WebCMM).

In order to perform the color transformations (forward and backward) as well as gamut mapping our CMM must generate a set of data structures, for a given color flow, during the initialization stage. To do this, we must provide several parameters:

- The source profile.
- The destination profile.
- The *CIELAB* sampling used for the TCS.
- The color adaptation model (a function that defines how the *CIEXYZ* tri-stimuli should be transformed) used before or after the color transformations.
- The gamut mapping method used.

We have built a JavaScript interface that allows the user to pass these parameters to the initialization functions written in C++. This interface also provides access to the transformation and gamut mapping functions as well as functions to generate 3D Look-Up Table (3D LUT). 3D LUT are required to perform accelerated color transformations with the GPU of the system running the web browser with WebGL[5].

[3]https://webassembly.org/
[4]https://emscripten.org
[5]https://www.khronos.org/webgl/

C. Source images

The web browsers can use many types of images. Simple images coming from files (jpeg, png, etc.) and videos (mp4, webm, etc.) but also more exotic sources through the Web Real-Time Communication (WebRTC)[6] Application Programming Interface (API). WebRTC is a JavaScript API developed by the W3C and the IETF, it is also a canvas for real-time communication. The purpose of WebRTC is to link applications such as video conferencing, screen or window sharing in peer-to-peer mode by freeing itself from the proprietary extension modules that were previously required.

These different types of images are all associated with input and output devices. Input devices for images, videos and WebRTC feeds from a WebCam (all three types of images are obtained using RGB sensors) and output devices for those obtained through screen and window sharing. The images produced by sharing screens or windows are quite special because in such a case the output device (corresponding to the split screen) becomes an input device when transferring the images as a video stream.

D. Transformation computations

We defined the JavaScript interface to perform the necessary calculations needed by our different color transformations. Although very efficient with WebAssembly, they do not allow high-resolution images or videos to be processed in an acceptable time. To overcome this, we implemented a highly efficient computational mechanism based on pre-calculated 3D LUT associated with graphics shaders managed through WebGL.

WebGL is a dynamic 3D API specification that allows OpenGL Embedded System to be used within a web page based on the HTML5 standard, with the help of the JavaScript language. WebGL offers a hardware acceleration for the calculation and 3D rendering with the help of the graphics processor of the computer system on which the web browser is running (computer, smartphone, tablet, etc.).

The *RGB* values sent to the color flow (coming from the input image) are used as input for the computation which is done by a tri-linear interpolation computed by a graphic shader with the 3D LUT.

[6]https://webrtc.org

IV. Experimental demonstration

In this section we describe the WebCMM demonstrator that we have developped. This demonstrator is available from this web page *https://www.couleur.org/articles/SITIS2019-I-WeCA/* (with a mirror site here *https://ipem.univ-st-etienne.fr/articles/SITIS2019-I-WeCA/*).

A. Color workflow

We chose to implement a color management process that transforms *RGB* colors from an input device (a camera or a monitor) to an output device (a monitor). This workflow, which can potentially process all the image types presented in Section III-C, is the first element displayed in our demonstrator (see the *Color Workflow box* in Figure 2).

It performs the following transformations:

- $RGB \to XYZ_{src}$: forward transformation
- $XYZ_{src} \to XYZ'_{src}$: color adaptation
- $XYZ \to Lab$: *CIEXYZ* to *CIELAB* color space conversion
- $Lab \to Lab_{\triangle E}$: *CIELAB* sampling forward transformation
- $Lab_{\triangle E} \to Lab_{\triangle EwithGM}$: gamut mapping
- $Lab_{\triangle EwithGM} \to Lab$: *CIELAB* inverse sampling transformation
- $Lab \to XYZ'_{dst}$: *CIELAB* to *CIEXYZ* color space inverse conversion
- $XYZ'_{dst} \to XYZ_{dst}$: inverse color adaptation
- $XYZ_{dst} \to RGB$: backward transformation

The whole sequence of transformations allows to create a single function, which is the composition of these transformations. This function, which is, for efficiency reasons, entirely realized in C++, will be used to generate the 3D LUT required for the real-time transformations a user would like to implement in 2D, 3D and virtual environments.

To be able to run this function, it is necessary, as a preliminary step, to launch an initialization process allowing to set up all the various data structures used internally. The parameters required for this function are, for this demonstrator, to be chosen among many presets (see the *Color Workflow Parameters* box in Figure 2).

B. Workflow parameters

The parameters chosen for this demonstrator allow to address the main usage scenarios presented in our introduction. Web developers or hybrid application developers who wish to produce classic sites or applications for e-commerce, marketing, etc. will prefer to use ICC profiles associated with chromatic adaptation, while a vision science researcher should be able to use profiles based on CxF files for reproducing identical stimuli on several screens as part of the implementation of a psychovisual experiment.

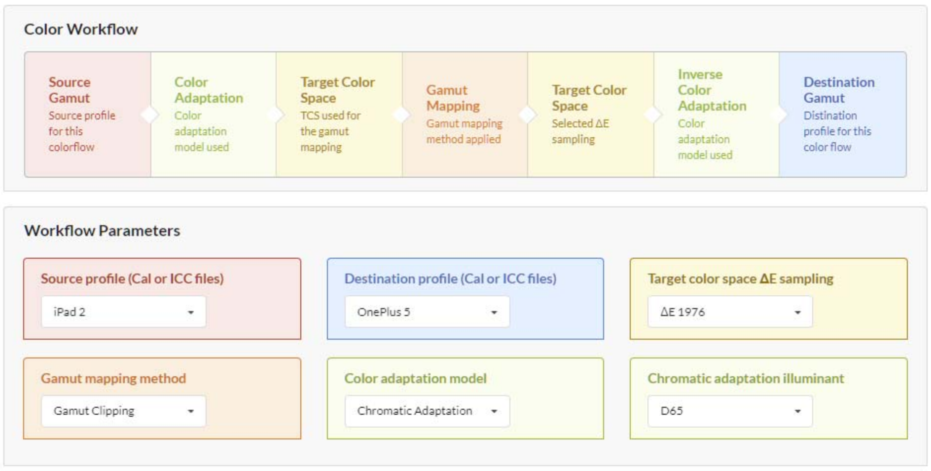
1) Source and destination gamuts: The source profile and the destination gamuts can be ICC or CxF files. This demonstration makes it possible to choose among 6 profiles (see Figure 3:

- 4 CxF profiles generated with the characterization tool presented in [8] corresponding to the following devices: 2 tablets with LCD screens (Apple iPad 2 and Google Pixel C), a smartphone on an OLED screen (OnePlus 5) and a virtual reality headset with an OLED screen (Oculus Rift).
- 2 generic ICC profiles: sRGB and ProPhoto.

2) Color adaptation and CIELAB *conversion:* The color adaptation is a function that defines how the *CIEXYZ* tristimuli should be transformed. Three different color adaptation models can be used in this color workflow:

- The first adaptation (Absolute) consists in using the white point of the destination device. The white point of a display corresponds to color value for which all R, G and B channels are set to their maximum. We use the *CIEXYZ* value of this white point as reference for all the colorimetric transformation to the *CIELAB* color space without any transformation of the *CIEXYZ* obtained with the forward model. With this method we try to reproduce exactly in the destination device the same *CIEXYZ* stimuli coming from the source device.
- The second adaptation (Luminance) is similar to the first one except that we first scale all the inputs *CIEXYZ* Y values in order to have the same *CIEXYZ* maximum value as the output device. With this method we equalize the luminance while we keep the *CIEXYZ* xy chromaticities. It represents a luminance compression or expansion (depending the scale of the luminance in the source and destination gamuts).
- The third one is a chromatic adaptation (Chromatic Adaptation). Several Chromatic Adaptation Transforms (CATs) exist in the state of the art, such as von Kries, Bradford, Sharp, CMCCAT2000 and CAT02 [13], [14]. Chromatic adaptation corresponds to the human visual systems ability to adapt (by adjusting human cone cell spectral sensitivity responses) to changes in illumination, in order to preserve the appearance of object colors. The aim of Chromatic Adaptation Transforms is to predict corresponding colors between pairs of corresponding colors, e.g. a color displayed on a screen observed under one illuminant and another color displayed on another screen observed under a different illuminant that has the same appearance than the first one. The method that we propose corresponds to the Bradford transform, any other CAT could be also used. XYZ values are independently scaled by a linear transformation (a 3x3 diagonal matrix) which compensates chromaticity shifts induced by a change of illuminant (between the input illuminant and the output illuminant). For this color adaptation the user can choose the illuminant that will be used from a predefined list of standard illuminants.

The first two methods of color adaptation can only be used if the source and destination profiles are CxF files because only these files contain the necessary colorimetric information. Only chromatic adaptation can be used when the source or destination, or both, are ICC profiles.



Color Workflow

Source Gamut: Source profile for this colorflow

Color Adaptation: Color adaptation model used

Target Color Space: TCS used for the gamut mapping

Gamut Mapping: Gamut mapping method applied

Target Color Space: Selected ΔE sampling

Inverse Color Adaptation: Color adaptation model used

Destination Gamut: Distination profile for this color flow

Workflow Parameters

Source profile (Cal or ICC files): iPad 2

Destination profile (Cal or ICC files): OnePlus 5

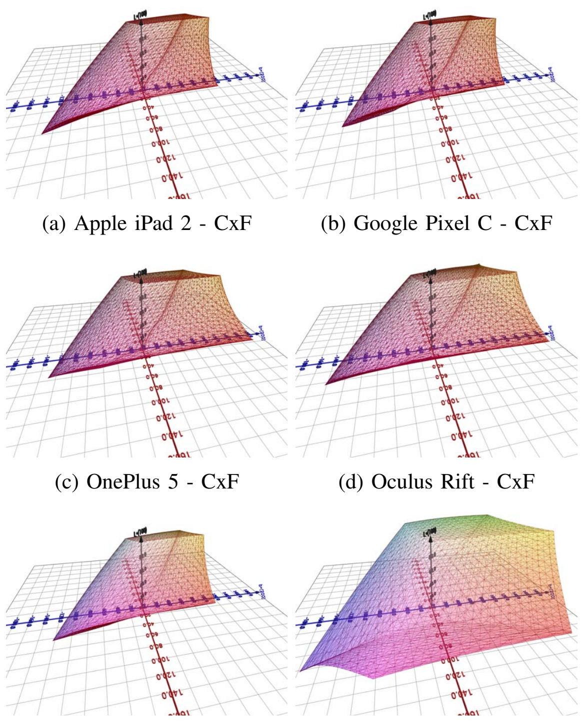
Target color space ΔE sampling: ΔE 1976

Gamut mapping method: Gamut Clipping

Color adaptation model: Chromatic Adaptation

Chromatic adaptation illuminant: D65

Fig. 2: The color flow implemented



(a) Apple iPad 2 - CxF (b) Google Pixel C - CxF

(c) OnePlus 5 - CxF (d) Oculus Rift - CxF

(e) sRGB - ICC (f) ProPhoto - ICC

Fig. 3: The 6 available profiles and the corresponding 3D visualization in *CIELAB*

3) Target color spaces: In this demonstrator we propose 3 TCS based on ΔE_{76}, ΔE_{94} and ΔE_{00} corresponding to a tabulated version of *CIELAB*, which can be found at the following address *https://data.couleur.org/deltaE/* described in [10].

4) Gamut mapping methods: We have chosen to include 3 gamut mapping techniques in our demonstrator, the first two (gamut clipping and gamut compression) are presented in [11] and the third (gamut compression and expansion) is an extension of gamut compression. For these three techniques we use the same convergence point (direction of mapping) which is the isobarycentre of the destination gamut and the ways how gamut compression and expansion are carried out are linear:

- Gamut clipping: change only the colors which are outside the destination gamut, for this we compute the intersection of the ray starting from the isobarycentre of the destination gamut and passing through the source color with the surface of the destination gamut.
- Gamut compression: we compute the intersections of the ray starting from the isobarycentre of the destination gamut through the source color with the surfaces of the source and destination gamuts. If the intersection with the source gamut is on the ray after the intersection with the destination gamut then the source color is transformed linearly into the destination gamut.
- Gamut compression and expansion: this technique makes possible to fully use the destination gamut but can lead to significant color changes. It is very similar to the gamut compression method with additional processing that are performed when the intersection with the source gamut is before the one with the destination gamut. In this case the source color is extended linearly in the destination gamut.

Our gamut mapping tool is very customizable and it allows, by changing the parameters that we can pass to it, to produce a large variety of techniques presented in [11]. For reasons of readability we have chosen to propose only the three techniques presented in the previous paragraph.

C. Data structures and transformations visualization

As mentioned above, all these parameters are used to generate the internal data structures required to transform the source colors in the destination gamut. Our demonstrator allows to display these data structures as well as the result of the transformations on a set of colors resulting from a sampling of the source RGB space.

When a color workflow parameter is modified, all a user has to do is to click on the "Generate" button to update all the internal data structures.

This view (see Figures 4 and 5) is divided into 4 sub-views:

1) Source Gamut: Display the source gamut in *CIELAB* with $\Delta E_{76,\ 94,\ 00}$ sampling.
2) Display Gamut: Display the destination gamut in *CIELAB* with $\Delta E_{76,\ 94,\ 00}$ sampling.
3) RGB Transformation: Transformation visualization of the source *RGB* color space with a sampling $8 \times 8 \times 8$ (512 samples).
4) Target Color Space Transformation: Transformation visualization of the 512 color samples in *CIELAB* with $\Delta E_{76,\ 94,\ 00}$ sampling for the gamut mapping (when a gamut clipping is used, the size of the cubes used to visualize the colors is proportional to the distance of this color from the destination gamut).

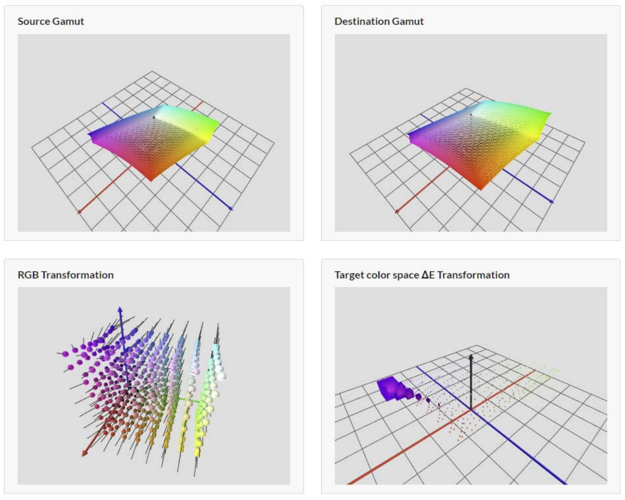


Fig. 4: Result with Source Gamut: iPad2, Destination Gamut: Oculus Rift, Gamut Mapping: clipping, TCS: ΔE_{1976}

We provide sub-views 3 and 4 in the Figure 6 for a comparison, for the same color workflow parameters, and for the three available gamut mapping methods. These two 3D visualizations show the transformed colors in the destination gamut (global transformation for the sub-view 3 and only the transformation during gamut mapping for the sub-view 4) using colored 3D geometric shapes and indicate with segments the transformations that the source color has undergone.

D. 3D LUT generation and test

When the initialization phase is completed, it is possible to generate the corresponding 3D LUT and use it with a test video (a screen capture simulating the use of WebRTC), either in a 3D environment (with the "Test Color Workflow" button) or in virtual reality (with the "Test Color Workflow in VR" button).

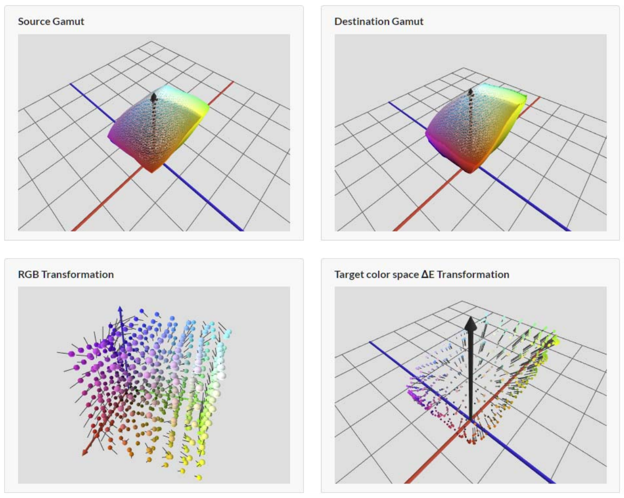


Fig. 5: Result with Source Gamut: iPad2, Destination Gamut: Oculus Rift, Gamut Mapping: compression and expansion, TCS: ΔE_{1994}

A new window (see Figure 7) is then generated in which a 3D environment is displayed containing the test video (the big buck bunny video[7] which is an open content: "The results of the Peach open movie project has been licensed under the Creative Commons Attribution 3.0 license.") shown with these original colors in a rectangle on the left and the video with these modified colors in a rectangle on the right.

We chose to generate a 3D LUT of size $64 \times 64 \times 64$. It is possible to increase the size of the 3D LUT (if this size can fit in the maximum size of the textures that WebGL can manage) but at the cost of a significant increase in computation time.

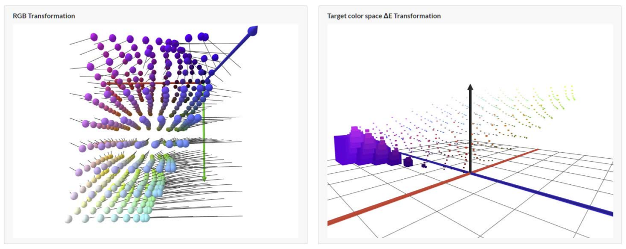
E. Performances

In this section we will show the computation times obtained on different web browsers (Google Chrome, Mozilla Firefox and Apple Safari) with several systems (a OnePlus 7 Pro smartphone, an iPad 2, a desktop computer with an Intel Core i7 6700K CPU and a Geforce RTX 2080 graphics card). These calculation times have been obtained for the following workflows:

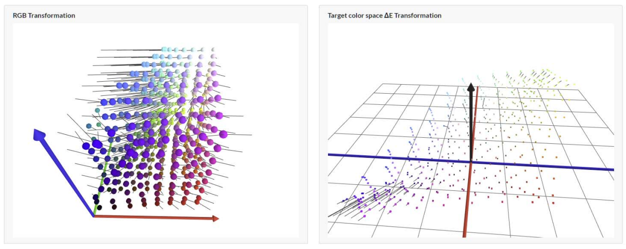
1) Source gamut: iPad2, Destination gamut: OnePlus 5; TCS: ΔE_{76}, Gamut mapping: Clipping, Color adaptation model: Absolute
2) Source gamut: iPad2, Destination gamut: Pixel C; TCS: ΔE_{94}, Gamut mapping: Compress, Color adaptation model: Luminance
3) Source gamut: sRGB, Destination gamut: ProPhoto; TCS: ΔE_{00}, Gamut mapping: Compress and expend, Color adaptation model: Chromatic adaptation

Table I presents the computation time for the data structure generation (initialization process) on several configurations. Table II presents the computation time for the 3D LUT generated on the same configurations.

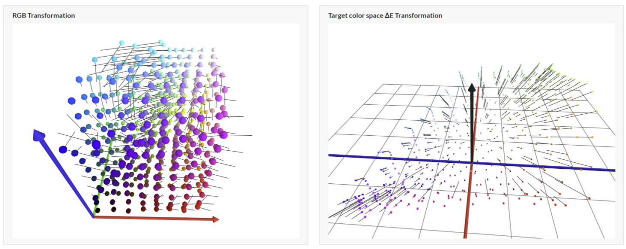
[7]https://peach.blender.org/download/



(a) Gamut clipping



(b) Gamut compression



(c) Gamut compression and expansion

Fig. 6: Results the three gamut mapping methods implemented in our demonstrator with the visualization of the corresponding color transformations (with an iPad2 device as source gamut and an OnePlus 5 device as destination gamut)

TABLE I: Initialization process time

Workflow	OnePlus7Pro Firefox	iPad 2 Safari	Desktop Firefox	Desktop Chrome
1	2.68s	2.64s	1.07s	0.89s
2	12.56s	10.00s	3.60s	3.10s
3	11.19s	9.23s	3.26s	2.82s

The initialization and the 3D LUT generation processes can take between 15 and 17 seconds on smartphone and tablet which is not acceptable. On a desktop computer the computation time is more appropriate (between 2.5 and 5 seconds). We plan to soon accelerate these processes by parallelizing some of the computations with the help of Web Workers.

The tri-linear interpolation computed for each frame of the video by a graphic shader with the 3D LUT is very fast, even on smartphones or tablets (less than 1ms on an HD frame).

V. Conclusion and Perspectives

In this article we demonstrated the possibility to implement a complete color management for html elements dedicated to 2D and 3D (via WebGL) and virtual environment (with WebVR) within modern web browsers, with the full implementation of a WebCMM. Even if these browsers already have

TABLE II: 3D LUT generation time

Workflow	OnePlus7Pro Firefox	iPad 2 Safari	Desktop Firefox	Desktop Chrome
1	3.82s	3.50s	1.15s	1.11s
2	4.56s	4.26s	1.67s	1.58s
3	3.67s	3.74s	1.49s	1.40s

a relatively complete image oriented color management, our proposal opens up new and particularly interesting possibilities for some WebRTC-based applications such as video conferencing and screen sharing (html elements, browser tabs or complete screens) but also the implementation of sophisticated soft proofing techniques dedicated to images and videos.

To promote this technology and distribute our WebCMM, we plan to implement new HTML tags to simplify its use. These tags will initially be associated with file formats that are natively supported by web browsers (*RGB* images and videos) for soft proofing and unconventional files (images and spectral videos). We will also be very vigilant about the security issues related to this technology.

Finally, we intend to integrate the management of the *CMYK* format into our WebCMM and thus be able to manage image files dedicated to printers. This new format will allow us to extend the capabilities for soft proofing techniques with our color management system.

References

[1] R.W.G. Hunt, The Reproduction of Colour, 5th Ed., Fountain Press, 1995
[2] R.W.G. Hunt, How to Shop on the Web Without Seeing Red, Proceedings of IS&T/SID Eight Color Imaging Conference, p. 2-7, 2000
[3] P. Green, Color Management: Understanding and using ICC Profiles,Wiley, 2010
[4] E.J Giorgianni, T.E. Madden, Digital color management: encoding solutions, Addison-Wesley, 1998
[5] J. Y. Hardeberg, Color management: principles and solutions, NORsignalets, vol. 3, pp. 612, 1999
[6] F. Dufaux, P. Le Callet, R. Mantiuk, M. Mrak, High Dynamic Range Video 1st Edition, ISBN: 9780081004128, pp.237-272, 2016
[7] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, Y. Wang. 2017. UnrealCV: Virtual Worlds for Computer Vision. In Proceedings of the 25th ACM international conference on Multimedia (MM '17). ACM, New York, NY, USA, 1221-1224
[8] J.-B. Thomas, J. Y. Hardeberg, I. Foucherot, P. Gouton, The PLVC display color characterization model revisited, Color Research And Application 33 (6), 2008. 449460. doi:10.1002/col.20447.
[9] J.-B. Thomas, Colorimetric characterization of displays and multi-display systems, PhD thesis, Université de Bourgogne, 2009.
[10] P. Colantoni, J.B. Thomas, A. Trémeau, Sampling CIELAB color space with perceptual metrics, International Journal of Imaging and Robotics, 16 (3), pp.1-22, 2016.
[11] J. Morovic, Color Gamut Mapping. John Wiley and Sons Ltd. 2008.
[12] J. Morovic, M.R. Luo, The fundamentals of gamut mapping: A survey. Journal of Imaging Science and Technology vol. 45, no. 3, pp. 283290, 2001.
[13] CIE Publ. 160: 2004. A Review of Chromatic Adaptation Transforms. 2004. Vienna: CIE Central Bureau; 2004.
[14] S. Bianco, R. Schettini, Two New von Kries Based Chromatic Adaptation Transforms Found by Numerical Optimization, Color research and application, pp 184-192, Volume 35, Number 3, June 2010
[15] L. M. Kasson,S. I. Nin, W. Plouffe, J. L. Hafner, Performing color space conversions with three-dimensional linear interpolation, Journal of Electronic Imaging , Vol. 4(3), pp. 226250, 1995.
[16] P. Colantoni, A. Trèmeau,Web Browsers Colorimetric Characterization, CCIW'2019, Chiba, February 2019, LNCS Vol. 11418, pp 145-161.
[17] 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.

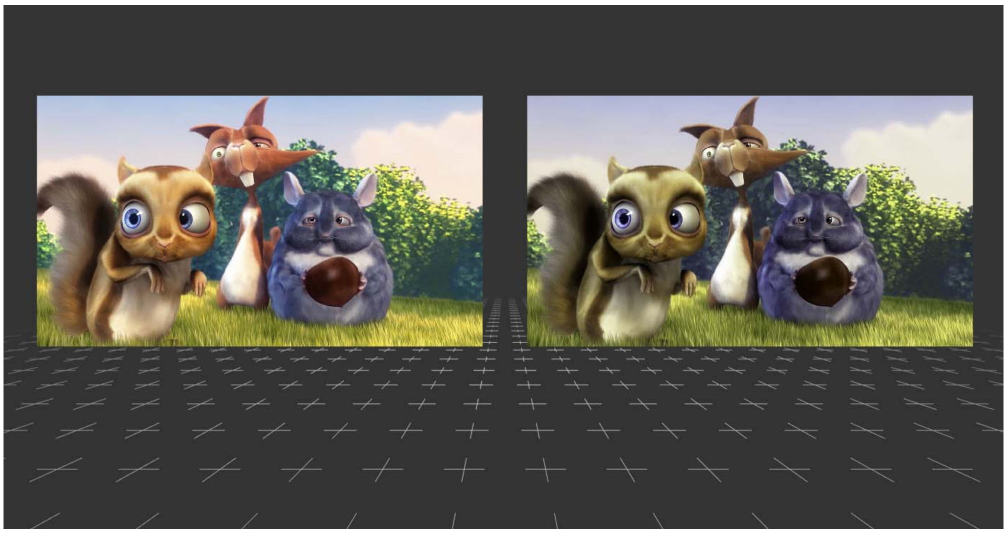


Fig. 7: Visualization of the result in a 3D environment: original colors on the left (sRGB ICC Profile) and modified colors on the right (ProPhoto ICC Profile)