Web Browsers Colorimetric Characterization

Philippe Colantoni[1(✉)] and Alain Trémeau[2]

[1] Université de Lyon, Université Jean Monnet, Centre Interdisciplinaire d'Études et de Recherches sur l'Expression Contemporaine, EA 3068, Saint-Etienne, France
philippe.colantoni@univ-st-etienne.fr
[2] Université de Lyon, Université Jean Monnet, Laboratoire Hubert Curien, UMR 5516, Saint-Etienne, France
alain.tremeau@univ-st-etienne.fr

Abstract. The great heterogeneity of mobile display devices currently available on the market makes the implementation of universal color management difficult. To address this problem, we have targeted a common feature of these devices: their ability to run web browsers. In this article we describe a colorimetric characterization tool, uncorrelated to the hardware and operating system used, which is an essential element in the implementation of an universal color management process in web browsers. This tool consists of a software that controls a calorimeter running on a computer located in the same local network as the mobile device running the web browser. It uses colorimetric parameters that allow us to obtain, for the metrics we want to optimize, accurate color transformation models (with maximum errors in ΔE_{1994} and ΔE_{2000} up to 2 times lower than the state of the art).

Keywords: Color management · Colorimetric characterization · Colorimetric calibration · Web browser color calibration

1 Introduction

The great heterogeneity of mobile display devices (smartphones and tablets) currently available on the market makes the implementation of an universal color management process difficult. To address this problem we have targeted a common feature of these devices: their ability to run web browsers. Smartphones and tablets are mobile devices that capture (with one or more cameras) and reproduce (through their displays) RGB images. In both cases the RGB images manipulated are dependent on the input and output devices that are used. The result is a color inconsistency that can be a problem. To face this issue a color management workflow must be used to compensate for any colorimetric differences between input and output images, and to ensure that a color scene is accurately acquired and displayed on a display.

S. Tominaga et al. (Eds.): CCIW 2019, LNCS 11418, pp. 145–161, 2019.
https://doi.org/10.1007/978-3-030-13940-7_12

The implementation of a color management system is therefore necessary. In this paper, we want to complete and improve the work we published in previous papers [12,13] toward an optimal colorimetric characterization that works for any display technology, with the aim to transpose it to mobile devices displays. In particular, we will focus on setting up the two main elements (the colorimetric characterization and calibration) necessary to create a color management module that can be fully executed within a web browser on a mobile device.

The colorimetric characterization and the colorimetric calibration of a mobile device are not trivial for end-users as it exists in the state of the art several solutions for which the efficiency varies from one device to another one, or from one product to another one of the same device model. In [4] the authors reported that the most efficient colorimetric characterization models tested in their study perform well for all tested iPhone displays, but that these models were ranked differently on two iPhones of the same model (iPhone 4), which suggests that the intra-model consistency may be an issue for the colorimetric characterization of smartphone (or tablet) displays. Furthermore, available solutions and free software are in general not user-friendly and not simple to use, as several parameters may impact the accuracy of these tasks. In this paper, we propose an alternative solution to conventional approaches based on a web application. To the best of our knowledge, only [4] addressed this problem in the literature. However, their solution contains less functionalities than the one we propose.

In this article we propose to address the issue of color characterization and calibration in a web browser with a dedicated tool (see Sect. 4). To allow the characterization of a very large number of display system types (desktop, mobile and VR headset displays), we will see in Sect. 5 that our solution includes some optimizations that improve the accuracy of our calibration models. Before that in Sects. 2 and 3 we will introduce the concept of color management and color characterization adapted for this case of study. Although also suitable for the characterization of VR headsets, we will focus, in this article, on mobile devices. However, we will briefly describe the method we use for virtual reality devices in Sect. 6.

2 Color Management

Color management consists in a controlled transformation between the color representations of different devices. Forward color transform consists to convert device dependent color image data, e.g. RGB values, into device independent color data, e.g. XYZ or $L^*a^*b^*$ values. Device independent color spaces such as $CIEXYZ$ or $CIELAB$ space represent colors in an absolute manner independent from any capturing or display device using the concept of standard observer. The development of a forward color transform (a forward model) requires knowledge of the image acquisition device (or a display device which can also be considered as an input device). This transform can be inverted by an inverse device transform (a backward model) adapted to the display device used that produces display device dependent color data, e.g. again RGB values, that ensures the accurate reproduction of the color defined by the device independent color data.

Device characterization and calibration are closely linked. They are the two key components of a color management system, they enable color images to be communicated and exchanged in a device independent color space. The aim of the device characterization is to model in the most accurate way how input values are processed by a device, it also provides several complementary information about this device such as the gamut of its primaries or the color channels correlation. Characterization is based on measurements of input values (e.g. RGB input values of a display) and output values (e.g. XYZ values measured on the display by a colorimeter or spectrometer). Color characterization enables to estimate, with a strong reliability, the color output of a display device while knowing the input color, and vice versa for an input device. The reliability of the characterization depends of the number of measures used to do this characterization, and of the set of colors (color patches) which are measured. It also allows to define the color gamut of the device.

While the characterization process defines the relationship that links the device dependent color space to the device independent color space, the calibration defines the setting up of this device, i.e. it defines how to match color values provided by this device to another one. The calibration step comes after the characterization step. The aim of the calibration step is to adjust the accuracy of output values produced by a device by comparison with a standard. Color calibration consists in set up a model from the set of color values used for the characterization and from the model resulting from this characterization, to ensure correct color reproduction of these color values.

The color management process is implemented through a Color Management Module (CMM). This CMM have to deal with the fact that the color gamut of different devices vary in range which makes an accurate reproduction impossible. For this reason, it has to implement a gamut mapping algorithm [3]. The gamut clipping is one of the simplest forms of gamut mapping that can be used.

3 Colorimetric Characterization

The colorimetric characterization of a device consists to model accurately how color values are transformed by this device. This can be done only by using a device independent color space such as the $CIEXYZ$ or the $CIELAB$ color space, as device dependent color spaces (such as the RGB color space) produce color values which differ from device to device.

Three main categories of colorimetric characterization methods can be found in the literature [1]. The first one aims to physically model the behavior of a color device. Generally, physical models are based on a number of simplifying conditions such are channel independence, chromaticity constancy, angle view independence [2]. The main disadvantage of physically models is the need to draw, test and validate assumptions which is time consuming and not simple for a wide range of end-users.

The second category corresponds to methods based on numerical models. Several numerical methods exist in the state of the art, such as polynomial regression methods, Radial Basis Functions (RBFs), neural networks [2]. These methods

are based on a training set which is used to estimate the optimal parameters for the numerical functions used. In comparison with the physical methods, numerical methods are able to accurately model devices without assumption of channel independence. However, when precision is needed these methods are expensive in terms of measurement and computation time.

The third category corresponds to methods that are using 3D Look Up Tables (3D LUT). In general 3D LUT are built on a regular lattice of the RGB cube. Nevertheless, more accurate LUT can be built on a regular lattice of the CIELAB color space [5]. The accuracy of this kind of methods depends on the number of measurements done to create the table and on the efficiency of the interpolation method used to estimate the color data that are between measurement points. 3D LUT are also used for the computation of ICC profiles. The main advantage of 3D LUT methods is that they do not need to make any assumption regarding the device used. On the other hand, their main disadvantage is the need of a huge number of measurements.

To the best of our knowledge very few studies have been focused on colorimetric characterization of mobile displays. According to [2] only Piecewise Linear Model Assuming Constant Chromaticity (PLCC), Piecewise Linear Model Assuming Variation in Chromaticity (PLVC), and masking models have been applied on mobile device displays. In their paper [2] the authors proposed a model based on Radial Basis Functions (RBF) and polyharmonic splines which outperforms the other methods. They also tested Artificial Neural Networks (ANN) but the main problem with ANN models is that the accuracy of results depends strongly of the number of neurons in the hidden layer(s). An increase of neurons in the layer(s) in general improves accuracy. However according [2], sometime when the number of neurons in the hidden layer(s) reaches a threshold there is no further improvement.

The main advantage of the PLVC model is that it needs only a small number of measured samples. Moreover, it is very simple in terms of implementation. On the other hand, this model does not take into account channel interdependence. PLVC model is able to model channel interdependence for all three primaries but only if we use a polynomial regression of 3rd degree with 19 variables that raises estimation problems especially on gamut boundary. The main difficulty with the RBF model is to choose optimal parameters and an adequate basis function to obtain a good accuracy which is not simple. The best results obtained by [2] were for polyharmonic spline kernel of 4th order.

The investigation done by [2] showed the importance of the size of a training set, their study leads to conclusion that the optimum size for ANN and RBF is around 150 samples, while for Polynomial regression models is around 100 samples. Good average accuracy could be obtained with smaller training sets, but the maximum color difference is then not acceptable. The distribution of color samples in the training set has also an impact on the overall accuracy. The quality of a colorimetric characterization is generally evaluated in terms of colorimetric accuracy but other factors must be analyzed, such as the smoothness of the 3D LUT - based color transform used [8].

In Sect. 3.4 we will develop a new color characterization technique based on the solution proposed in [13]. Our intent is to extend the field of use of this technique in order to improve the accuracy of our calibration models when they are evaluated with perceptually accurate metrics such as the ΔE_{1994} and ΔE_{2000}. For this purpose, we will introduce in the following sub sections 2 related concepts: the forward and backward transformation models (see Sect. 3.2) and the target color space (TCS) (see Sect. 3.3). But before that, we will discuss the opportunity to use ICC profiles for mobile devices.

3.1 Can We Use ICC Profiles?

The International Color Consortium (ICC) provides a solution by defining standards to store calibration data in ICC profiles. ICC profiles describe the color attributes of an input or an output device by defining a mapping model between these devices and a Profile Connection Space (PCS). This PCS can be either the CIEXYZ or the CIELAB ($L^*a^*b^*$). Mappings may be specified using Look Up Tables (LUT), to which interpolation is applied, or through a series of parameters for transformations. ICC profiles can be used by many software enabling color management. Considering that some browsers do not support ICC profiles, one option is to use the default ICC profile of the mobile display used; another option consists to compute the profile of the mobile device using the settings defined by the user.

Associated with the sRGB color space which is the standard color space used by browsers that support color profiles [6], ICC profiles can be a good solution but it will not cover all possible scenarios (operating systems and web browsers).

3.2 The Forward and Backward Transformation Models

The forward and backward forward transformation models correspond to the following parts of a color workflow:

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

As described in [13]: the forward transformation used is based on polyharmonic splines (a subset of the Radial Basis Functions that can be used for interpolating or approximating arbitrarily distributed data); the backward transformation used (or inverse transformation) is based on a tetrahedral interpolation.

3.3 Target Color Space

The **Target Color Space** (TCS) is different from the Profile Connection Space used with ICC profiles. The TCS used in our method is based on CIELAB but we adapted it to any color metrics (i.e. any ΔE_m) that a user would like to optimize. In [5] we proposed a sampling method of the CIELAB color space based on non-Euclidean color differences. Here, we propose to use this sampling

method to improve the accuracy of any color metric (with the aim to minimize the maximum error). Different sampling strategies can be used depending of the color metric used (e.g. ΔE_{1976}, ΔE_{1994}, ΔE_{CMC}, ΔE_{BFD} or ΔE_{2000}) and of the sampling distance considered. This sampling does no impact the accuracy of the forward and backward transforms. The only additional errors that may result from this sampling are related to numerical errors resulting from the geometrical model and from the interpolation method used. On the other hand, this uniform sampling, based on a 3D close packed hexagonal grid, has a strong impact on the geometrical shape of the resulting gamut (see Fig. 1). Not only the shape near the borders of the gamut and the volume of the gamut are concerned but also its centroid.

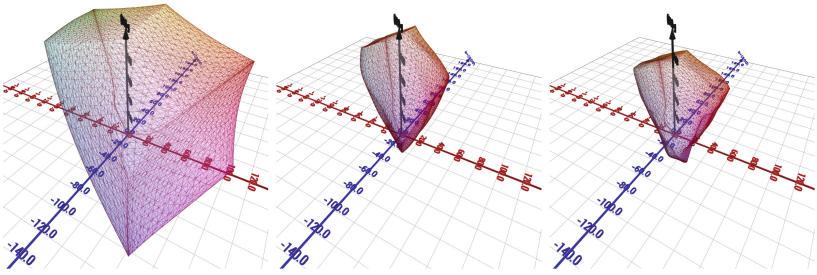

Fig. 1. Gamut of a Google Nexus 9 with 3 TCS based on ΔE_{1976}, ΔE_{1994} and ΔE_{2000}

In this new paper we will focus on 3 TCS based on Δ_{1976}, Δ_{1994} and Δ_{2000} corresponding to a tabulated version of $CIELAB$, which can be found at the following url address [10] (see [5] and Fig. 2).

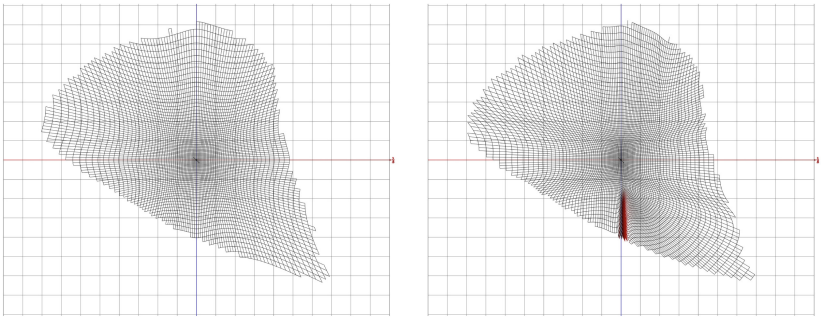

Fig. 2. $CIELAB$ Sampling corresponding to Δ_{1994} and Δ_{2000}

3.4 Characterization Process

Considering that we want to characterize a display with N colors, the characterization method that we propose is based on the following steps:

1. Create a dataset of $27 + 24 = 51$ samples, built on a $3 \times 3 \times 3$ lattice of the RGB cube (i.e. 27 samples) complemented by 4 samples for each face (i.e. 24 samples) of the RGB cube (inter points of the 3×3 RGB values), as learning set (which will be used for the initialization of the interpolation process).
2. Measure the initial dataset and compute (as [13]) the forward and backward models (with predefined non optimal parameters).
3. Create a second dataset of N-51 samples from a uniform sampling of the target color space based on a 3D close packed hexagonal grid (see Fig. 3). The color values defined depend of the color metric used and of the color gamut of the display (see Sect. 3.3). This dataset will be used as optimization dataset.
4. The N-51 color values of the new dataset are ordered by decreasing Lightness (L*) value (as in [13]). This order defines the order of color values that will be measured during the characterization process. In order to be displayed on the screen, next measured, the N-51 L*a*b* color values are sequentially converted (and refined) in RGB values using the backward transform.
5. We refine the accuracy of the estimation model by adding to the current learning dataset the 1st color value on the top of the current optimization dataset. The model and the optimization dataset are recomputed (the forward and backward models) and refined at each iteration of the process value (as [13]), i.e. each time a new color value is taken into account (i.e. added to the learning set and consequently removed from the optimization set).
6. Create a test dataset of 64 samples, built on a $4 \times 4 \times 4$ lattice of the RGB cube, as validation set. The color values of this dataset are by definition (due to the over sampling) different from those of the 1st learning set and of the 1st optimization set. Analyze the accuracy of the estimation model from this dataset.

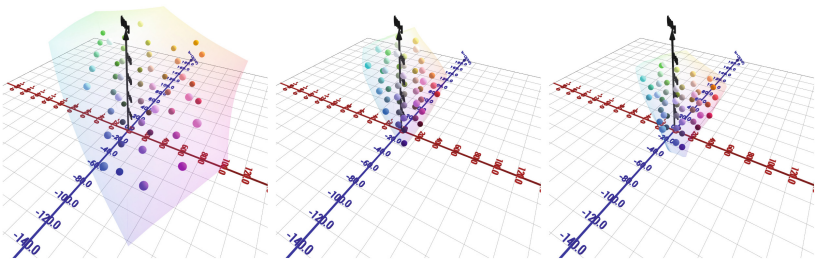

Fig. 3. 49 color samples for a Google Nexus 9 with 3 TCS (ΔE_{1976}, ΔE_{1994} and ΔE_{2000}) from a uniform sampling based on a 3D close packed hexagonal grid.

This characterization method is similar as the one which was proposed in [13], nevertheless now the choice of the target color space significantly changes the color distribution. This enables us to decrease the maximum errors. A more precise description of the implementation of this method is available in Sect. 4.1.

4 A New Tool for Web Browser Color Characterization

We introduced a first color management pipeline in [12] devoted to display calibration that included a screen characterization tool. This tool can only work on operating systems able to drive a colorimeter, i.e. Windows, MacOS and Linux. The main novelty of the new tool that we propose is to remove this limitation and thus to extend the area of use to Android and iOS operating systems. To reach this objective, we developed a web application that is able to drive, on a smartphone (or on a tablet, a VR headset), the display of colors from a computer connected on the same local network.

Most of web browsers support Color Management System (CMS) and ICC color profiles. With some smartphones users can customize the color profile of their display thanks to its Operating System (OS) color management. With other smartphones, the web browser does not have this option due to the operating system used. In this case, users are dependent of the default color management settings set by the manufacturer of this device. To face this issue we propose a characterization method that is placed above the existing CMS already present (or not) in the web browser.

The web application is provided by our software which includes a web server (the web application module). The mobile application module is used to display input color values sent by the web application module (through generated web pages). The colorimeter module is used to measure output color values displayed on the screen of the mobile phone (or the tablet, the VR headset) and to send these values to the color calibration module. The computer drives the colorimeter. The web application module has three main functionalities: (i) colorimetric characterization, (ii) colorimetric calibration and (iii) colorimetric correction.

The second novelty of this new color characterization module is that the new sampling method used for the TCS gives more accurate results than the previous one [13].

4.1 Tool Description

Performing colorimetric characterization of mobile devices can be an issue for end-user because the number of measurements to be done to perform this process is very tedious and time consuming if performed by hand. According [4], in the case of conventional displays connected to a PC with a Windows/Linux operating system this task can be easily automated by means of a simple cross-platform application which instructs a spectroradiometer to take measurements and displays a series of color patches on the display in a synchronized manner. However, there is no common programming environments for all mobile

devices. For mobile devices there are various programming environments, e.g. Objective-C for iOS application on iPhone, Java or C/C++ for Android application, Windows Mobile 8, Firefox OS and Tizen. In [4] the authors proposed a solution that makes use of the WebSocket API in the HTML5 standard which allows bi-directional, full-duplex communications between a web browser and a server. The measurement setup consists of a smartphone which runs a web browser and a PC which runs a web server and the control software of the spectroradiometer. As long as the smartphone's web browser supports WebSockets, the framework can run without any problem.

Fig. 4. Our software characterizing an iPad 2

The application that we propose (see Fig. 4), as the solution proposed by [4] aims to overcome this problem, i.e. to avoid of developing different control applications for all platforms separately. Written in C++ available on MacOS on Windows, we describe below how this tool works:

1. Launch the application on the server computer, enter the following parameter values: N the number of color values that will have to be measured (where N = 51 samples of the 1st learning set + Nb of samples of the optimization set); ΔE_m is the $CIELAB$ color metric which defines the TCS that will be used by the calibration model (e.g. ΔE_{76}, ΔE_{94}, ΔE_{CMC}, ΔE_{BFD} or ΔE_{00}); and lastly the type of device to characterize (e.g. a mobile display).
2. The application displays the URL to be used by the client on his web browser. The user has to enter this URL in the web browser to start the characterization process. The web page generated by the server is set to follow the instructions sent by the server.
3. The server sends a request to the user: "Start the characterization process". If the user is not familiar with the application, an help explains how to use the spectrocolorimeter in order to measure colors displayed on a screen. The spectrocolorimeter must be set in front of the center of the web browser window. The color of the background of this window is set to red and a message "Wait" is displayed in this window.

4. The characterization tool process the 51 colors of the learning set by sending them, one by one, from the server to the mobile device. For each color, the mobile device receives an instruction from the server to refresh the window in order to set the color of the background with the current color. The client web browser informs the server that it is ready for a new measure. Then, the spectrocolorimeter receives an instruction from the server to proceed the color measurement which send the measured values to the server. Next, the server proceeds to the next color or ends the process.
5. A first estimation model is computed by the server computer from these 51 pairs of input and output values.
6. A first optimization dataset of N colors is computed by the server computer.
7. Next, the N-51 colors ($CIELAB$ based on Δ_m values sequentially backward transformed in RGB values) of the optimization set are sent, one by one, by the server to the client web browser of the mobile device. For each color, the client receives an instruction from the server to refresh the window in order to set the color of the background with the targeted color. The client informs the server that it is ready for a new measure. Then, the spectrocolorimeter receives an instruction from the server to proceed color measurement and send the measured values to the server. Next, the server proceeds to the next color or ends the process.
8. The learning dataset is updated by the server computer according the iterative process defined in Sect. 3.4.
9. A refined estimation model (the forward and backward transformations) is computed by the server computer from the new learning dataset created. From one iteration to the next one, the learning set is enriched by a new color with a lower lightness than the previous one added before, as a consequence the optimization dataset must be improved also.
10. The current optimization dataset is updated by the server computer according to the new learning dataset computed, next step 7 is re-iterated until all the N-51 colors are measured.
11. During the validation process the 64 colors of the validation set (see Sect. 3.4) are sent, one by one, by the server to the client web browser of the mobile device. For each color, the client web browser receives an instruction from the server to refresh the window in order to set the color of the background with the current color. The client informs the server that it is ready for a new measure. Then, the spectrocolorimeter receives an instruction from the server to proceed color measurement and send the measured values to the server. Next, the server proceeds to the next color or ends the process.
12. Optimization of the model parameters (color space used for the transformation $CIELab$ or $CIEYXZ$, polyharmonic kernels and smoothing factor) by the server computer using a brute-force process, as in [13].
13. Saving of the resulting file and of the corresponding ICC profile (in a data format corresponding to a 3D LUT).

4.2 Calibration Files

The file format that we propose to encode our calibration data is not compatible with ICC 4.2 specifications but can be easily converted in this format. We have opted for a calibration file format from which the set of colors used (for forward and backward transforms) are directly readable by a text editor. It is based on pairs of values:

- the RGB device-dependent color values that the output device can reproduce. This set of color values will be addressed by our color management system to ensure correct color reproduction on the output device.
- the corresponding xyY device-independent color values that the output device can reproduce. This set of color values corresponds to the set of color values measured during the characterization step which will be processed in the Target Color Space used.

The files also contains the additional set of 64 pairs of values to ensure the accuracy of the color management solution. This additional set of color values will be first used as a test set to check the accuracy of the calibration test, next will be integrated in the learning base to refine the accuracy of the forward and backward models. Additional data containing the parameters of the calibration model are also included in our files (these parameters can be found in [13]). They allow, when reading the files, to generate an optimal calibration.

5 Results

We chose to test our characterization process with 4 devices: 1 smartphone with an OLED display (an OnePlus 5 from OnePlus) and 3 tablets with LCD screens (an Apple iPad 2, a Google Pixel C and a Google Nexus 9). For all these devices we used Mozilla Firefox as web browser.

To test the accuracy of our characterizations, we have chosen to avoid any external lighting source and to use absolute colorimetry. For this purpose, all our CIEXYZ to CIELAB conversions use as white reference the XYZ value corresponding to the RGB triplet (1, 1, 1) (each channels of RGB values vary between 0 and 1). This allows us to avoid any chromatic adaptation.

Table 1 shows the results we obtained for the 4 devices tested with 100 color samples. For each of these devices we carried out 3 characterizations corresponding to the following 3 TCS: $CIELAB$ based on Δ_{1976}, $CIELAB$ Δ_{1994} and $CIELAB$ Δ_{2000}. Each of these characterizations took 5 minutes with a X-Rite i1 Pro 2 Spectrocolorimeter. Table 2 shows the results we obtained for 2 of these devices tested with 2 TCS ($CIELAB$ Δ_{1976} and $CIELAB$ Δ_{2000}) with 100, 150 and 200 samples, respectively. For these 2 tables, each line, which corresponds to a complete characterization, displays 3 accuracy indexes (the average error, the maximum error and the 95 percentile value) of the forward model's accuracy. The model accuracy was calculated from the N samples when compared to the 64 colors of the test base.

Table 1. Forward transform ΔE accuracy - Results for the 4 devices tested with 100 samples

Devices	Samples	TCS	Average ΔE_{1976}	ΔE_{1994}	ΔE_{2000}	Max TCS	ΔE_{1976}	ΔE_{1994}	ΔE_{2000}	95 TCS	ΔE_{1976}	ΔE_{1994}	ΔE_{2000}
iPad 2	100	1976	0.686	0.369	0.354	1976	3.448	3.343	3.969	1976	1.603	0.668	0.642
iPad 2	100	1994	0.632	0.320	0.301	1994	2.215	1.330	1.019	1994	1.388	0.618	0.695
iPad 2	100	2000	0.874	0.448	0.417	2000	4.565	1.448	1.361	2000	1.824	1.194	1.125
Nexus 9	100	1976	1.482	0.675	0.628	1976	4.364	2.404	2.120	1976	3.049	1.407	1.343
Nexus 9	100	1994	1.475	0.639	0.597	1994	4.754	1.383	1.402	1994	2.829	1.253	1.176
Nexus 9	100	2000	1.780	0.765	0.708	2000	7.484	2.279	1.996	2000	3.905	1.750	1.507
Pixel C	100	1976	0.986	0.558	0.546	1976	2.392	1.933	2.568	1976	1.995	1.263	1.208
Pixel C	100	1994	0.894	0.522	0.486	1994	2.199	1.637	1.324	1994	1.912	1.142	1.056
Pixel C	100	2000	1.058	0.610	0.556	2000	3.166	1.760	1.562	2000	2.193	1.424	1.352
OnePlus 5	100	1976	1.155	0.597	0.565	1976	4.860	3.073	2.510	1976	2.388	1.217	1.327
OnePlus 5	100	1994	1.271	0.602	0.583	1994	3.163	1.633	1.611	1994	2.580	1.264	1.345
OnePlus 5	100	2000	1.521	0.674	0.650	2000	3.737	2.254	1.857	2000	3.474	1.265	1.450

Table 2. Forward transform ΔE accuracy - Results for 2 devices tested (Google Pixel C and Google Nexus 9) with 100, 150 and 200 samples

Devices	Samples	TCS	Average ΔE_{1976}	ΔE_{1994}	ΔE_{2000}	Max TCS	ΔE_{1976}	ΔE_{1994}	ΔE_{2000}	95 TCS	ΔE_{1976}	1994	2000
Pixel C	100	1976	0.986	0.558	0.546	1976	2.392	1.933	2.568	1976	1.995	1.263	1.208
Pixel C	150	1976	0.894	0.520	0.480	1976	2.847	1.587	1.347	1976	2.172	1.130	0.995
Pixel C	200	1976	0.739	0.406	0.383	1976	1.930	1.266	1.252	1976	1.454	0.977	0.995
Nexus 9	100	2000	1.780	0.765	0.708	2000	7.484	2.279	1.996	2000	3.905	1.750	1.507
Nexus 9	150	2000	1.327	0.606	0.574	2000	4.789	1.741	1.676	2000	2.848	1.268	1.101
Nexus 9	200	2000	1.115	0.561	0.529	2000	4.027	1.691	2.182	2000	2.338	1.289	1.199

Table 3. Backward transform *RGB* accuracy - Results for the 4 devices tested with 100 samples

Devices	Samples	TCS	Average	Max	95
iPad 2	100	ΔE_{1976}	0.0062	0.0163	0.0116
iPad 2	100	ΔE_{1994}	0.0064	0.0222	0.0140
iPad 2	100	ΔE_{2000}	0.0086	0.0240	0.0202
Nexus 9	100	ΔE_{1976}	0.0291	0.1616	0.0885
Nexus 9	100	ΔE_{1994}	0.0282	0.1766	0.0845
Nexus 9	100	ΔE_{2000}	0.0319	0.1331	0.0924
Pixel C	100	ΔE_{1976}	0.0095	0.0233	0.0203
Pixel C	100	ΔE_{1994}	0.0103	0.0264	0.0201
Pixel C	100	ΔE_{2000}	0.0105	0.0290	0.0208
OnePlus 5	100	ΔE_{1976}	0.0094	0.0328	0.0189
OnePlus 5	100	ΔE_{1994}	0.0113	0.0348	0.0243
OnePlus 5	100	ΔE_{2000}	0.0121	0.0280	0.0262

Tables 3 and 4 show the results for the backward model with the same 3 accuracy indexes (the average error, the maximum error and the 95 percentile value).

Table 4. Backward transform *RGB* accuracy - Results for 2 devices tested (Google Pixel C and Google Nexus 9) with 100, 150 and 200 samples

Devices	Samples	TCS	Average	Max	95
Pixel C	100	ΔE_{1976}	0.0095	0.0233	0.0203
Pixel C	150	ΔE_{1976}	0.0082	0.0205	0.0175
Pixel C	200	ΔE_{1976}	0.0067	0.0202	0.0147
Nexus 9	100	ΔE_{2000}	0.0319	0.1331	0.0924
Nexus 9	150	ΔE_{2000}	0.0242	0.1426	0.0867
Nexus 9	200	ΔE_{2000}	0.0158	0.1276	0.0579

As we can see in Tables 1 and 2, the results are in line with our assumptions done in Sect. 3.3. The use of a given TCS allows to obtain much lower maximum error and 95 percentile values when these accuracy indexes are calculated with the corresponding metric. The results are particularly good with the TCS based on ΔE_{1994} regardless of the device characterized. The TCS based on ΔE_{2000} is lagging because of the lower quality of its sampling (see Fig. 2 and Ref. [5]).

Unfortunately, as one can see in Tables 3 and 4, the improvement of the forward model does not benefit from its inverse model for TCS based on ΔE_{1994} and ΔE_{200}. On the contrary, we even notice a loss of precision due to 2 concomitant factors: for these 2 TCS the gamuts undergo significant distortions which degrade the tetrahedral interpolation used in the calculations of this model; the inverse model being based on a sampling obtained with the classical model it undergoes an accumulation of numerical errors (due to the resampling of the CIELAB space among others).

Tables 2 and 4 show us that in any case the increase in the number of colors improves the accuracy of the forward and reverse models. However, the increase in the number of colors mechanically increases the time required for characterization (5 min for 100 colors, 8 min for 200 colors). However, the accuracy of the models is limited by the stability of the display device (its ability to reproduce the same color identically) and the quality of the spectrophotometer used to perform the measurements.

6 Usage and Perspectives

The first possible usage of this tool is to create, from the characterization files we have produced, standard ICC profiles in order to use them in classic color workflows based on Microsoft, Adobe or Apple Color Management Modules or possibly open source CMMs like lcms2. We have chosen to offer a higher level of integration by implementing a color workflow based on a CMM entirely executed by web browsers (a WebCMM) with a simple javascript interface.

In the Sect. 3.2 we presented the two transformations which define our calibration model, our Color Management Module (CMM). Written in C++, theses transformations are not linked to any external software module. The entire calibration process is therefore performed by our code independently of any function of the system on which it is performed. It is for this reason that we were able to generate a particular version of our library entirely compiled in WebAssembly. WebAssembly, or wasm, is a low-level binary programming language for developing applications in web browsers. This is a standard of the "World Wide Web" consortium that has been designed to complement JavaScript with superior performance. 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. The transformation of our code into wasm allowed us to define interface functions in Javascript. These functions now allow us to run our color management system within web browsers using standard html pages. We therefore have a color management module dedicated to the Web (a WebCMM).

The second possible usage of this tool is for characterizing Virtual Reality headsets. To do this, we have developed a specific procedure for generating web pages in our characterization tool. The generated pages use *a-frame* [9] a Javascript library that allows to produce and manipulate 3D objects in a virtual environment displayed in a virtual reality headset. The pages we produce generate a virtual environment with a uniform color that is displayed on the screens

integrated into the headset using the *WebVR* standard. To make colorimetric measurements we use a simple fixing system that allows us to hold the sensor of our spectrocolorimeter on the surface of one of the helmet lenses. We thus have a tool capable of characterizing all the display devices that modern web browsers can manage.

Now that we have a complete color management system dedicated to browsers (and VR headset driven by *WebVR*) we can start porting other tools to the Web. Our objective in the coming months will be to offer the first web framework to natively manage multi and hyperspectral images in a web browser. This framework will have a complete management of all calculations (performed in real time using WebGL shaders) that will allow the following transformations: reflectance to color (using a color reconstruction and an illuminant) then color to RGB using our WebCMM.

7 Conclusion

In this paper we have presented an innovative software tool dedicated to color characterization of web browsers. This tool, is able of driving any Web browser running on the same computer or located on the same local network. This tool was developed to characterize many types of display devices: smartphones, tablets, monitors, virtual reality headsets. It integrates an algorithm that allows it to dynamically select color patches according to several criteria. These criteria, which depend of the number of sample colors and of the TCS, define the precision of the models (forward and backward) that will be produced.

The quality of the generated models depends largely on the stability of the display device that is characterized. It cannot be possible to produce a good predictive model if the color measurements are not stable. However, we shown in this study that the increase in the number of colors significantly improves the accuracy of the models produced. If a characterization must be performed to produce a quality forward model, we advise to use a number of sample colors greater than 200 associated with a TCS based on the ΔE_{1994}, meanwhile for the reverse model we advise to also use a number of sample colors greater than 200 but associated this time with a TCS based on the ΔE_{1976}.

References

1. Thomas, J.-B., Hardeberg, J.Y., Foucherot, I., Gouton, P.: The PLVC display color characterization model revisited. Color Res. Appl. **33**(6), 449–460 (2008). https://doi.org/10.1002/col.20447
2. Poljicak, A., Dolic, J., Pibernik, J.: An optimized Radial Basis Function model for color characterization of a mobile device display. Displays **41**, 61–68 (2016)
3. Morovic, J.: Color Gamut Mapping. Wiley, Chichester (2008). ISBN: 978-0-470-03032-5
4. Byshko, R., Li, S.: Characterization of iPhone displays: a comparative study. In: Workshop Farbbildverarbeitung, Darmstadt (2012)

5. Colantoni, P., Thomas, J.B., Trémeau, A.: Sampling CIELAB color space with perceptual metrics. Int. J. Imaging Robot. **16**(3), 1–22 (2016)
6. Javorsek, D., Mocnik, J., Staresinic, M.: Analysis of colour appearances on different display devices. Tekstilec **58**(2), 100–107 (2015)
7. Velea, R., Gordon, N.: Evaluating gamut coverage metrics for ICC color profiles. Int. J. Chaotic Comput. (IJCC) **4**(2), 113–117 (2016)
8. Aristova, A.: Smoothness of color transformations. Master i Teknologi - Medieteknikk, Hogskolen, Gjovik (2010)
9. https://aframe.io/. Accessed 26 Sept 2018
10. https://data.couleur.org/deltaE/. Accessed 26 Sept 2018
11. https://viva-arts.univ-st-etienne.fr/spectralviewer.html. Accessed 26 Sept 2018
12. Colantoni, P., Thomas, J.-B.: A color management process for real time color reconstruction of multispectral images. In: Salberg, A.-B., Hardeberg, J.Y., Jenssen, R. (eds.) SCIA 2009. LNCS, vol. 5575, pp. 128–137. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02230-2_14
13. Colantoni, P., Thomas, J.B., Hardeberg, J.Y.: High-end colorimetric display characterization using an adaptive training set. J. Soc. Inf. Disp. **19**, 520–530 (2011)