A Color Management Process for Real Time Color Reconstruction of Multispectral Images

Philippe Colantoni[1,2] and Jean-Baptiste Thomas[3,4]

[1] Université Jean Monnet, Saint-Étienne, France
[2] Centre de recherche et de restauration des musées de France, Paris, France
[3] Université de Bourgogne, LE2I, Dijon, France
[4] Gjøvik university College, The Norwegian color research laboratory, Gjøvik, Norway

Abstract. We introduce a new accurate and technology independent display color characterization model for color rendering of multispectral images. The establishment of this model is automatic, and does not exceed the time of a coffee break to be efficient in a practical situation. This model is a part of the color management workflow of the new tools designed at the C2RMF for multispectral image analysis of paintings acquired with the material developed during the CRISATEL European project. The analysis is based on color reconstruction with virtual illuminants and use a GPU (Graphics processor unit) based processing model in order to interact in real time with a virtual lighting.

1 Introduction

The CRISATEL European Project [4] opened the possibility to the C2RMF of acquiring multispectral images through a convenient framework. We are now able to scan in one shot a much larger surface than before (resolution of 12000×20000) in 13 different bands of wavelengths from ultraviolet to near infrared, covering all the visible spectrum.

The multispectral analysis of paintings via a very complex image processing pipeline, allows us to investigate a painting in ways that were totally unknown until now [6].

Manipulating these images is not easy considering the amount of data (about 4GB by image). We can either use a pre-computation process, which will produce even bigger files, or compute everything on the fly.

The second method is complex to implement because it requires an optimized (cache friendly) representation of data and a large amount of computations. This second point is not anymore a problem if we use parallel processors like graphic processor units (GPU) for the computation. For the data we use a traditional multi-resolution tiled representation of an uncorrelated version of the original multispectral image.

The computational capabilities of GPU have been used for other applications such as numerical computations and simulations [7]. The work of Colantoni and

A.-B. Salberg, J.Y. Hardeberg, and R. Jenssen (Eds.): SCIA 2009, LNCS 5575, pp. 128–137, 2009.

al. [2] demonstrated that a graphic card can be suitable for color image processing and multispectral image processing.

In this article, we present a part of the color flow used in our new software (PCASpectralViewer): the color management process. As constraints, we want the display color characterization model to be as accurate as possible on any type of display and we want the color correction to be in real time (no pre-processing). Moreover, we want the model establishment not to exceed the time of a coffee break.

We first introduce a new accurate display color characterization method. We evaluate this method and then describe its GPU implementation for real time rendering.

2 Color Management Process

The CRISATEL project produces 13 planes multispectral images which correspond to the following wavelengths: 400, 440, 480, 520, 560, 600, 640, 680, 720, 760, 800, 900 and 1000nm. Only the 10 first planes interact with the visible part of the light. Considering this, we can estimate the corresponding $XYZ()$ tri-stimulus values for each pixel of the source image using Equation 1:

$$\begin{cases} X = \sum_{\lambda=400}^{\lambda=760} x(\lambda)R(\lambda)L(\lambda) \\ Y = \sum_{\lambda=400}^{\lambda=760} y(\lambda)R(\lambda)L(\lambda) \\ Z = \sum_{\lambda=400}^{\lambda=760} z(\lambda)R(\lambda)L(\lambda) \end{cases} \quad (1)$$

where $R(\lambda)$ is the reflectance spectrum and $L(\lambda)$ is the light spectrum (the illuminant).

Using a GPU implementation of this formula we can compute in real-time the XYZ and the corresponding $L^*a^*b^*$ values for each pixel of the original multispectral image with a virtual illuminant provided by the user (standard or custom illuminants).

If we want to provide a correct color representation of these computed XYZ values, we must apply a color management process, based on the color characterization of the display device used, in our color flow. We then have to find which RGB values to input to the display in order to produce the same color stimuli than the retrieved XYZ values represents, or at least the closest color stimuli (according to the display limits).

In the following, we introduce a color characterization method which gives accurate color rendering on all available display technologies.

2.1 Display Characterization

A display color characterization model aims to provide a function which estimates the displayed color stimuli for a given 3-tuple RGB input to the display. Different approaches can be used for this purpose [5], based on measurements of input values (i.e. RGB input values to a display device) and output values (i.e.

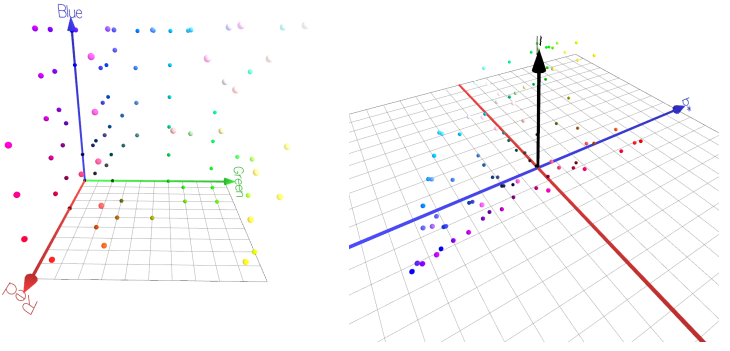

Fig. 1. Characterization process from *RGB* to $L^*a^*b^*$

XYZ or $L^*a^*b^*$ values measured on the screen by a colorimeter or spectrometer) (see figure.1).

The method we present here is based on the generalization of measurements at some position in the color space. It is an empirical method which does not consider any assumptions based on display technology. The forward direction (*RGB* to $L^*a^*b^*$), is based on RBF interpolation on an optimal set of measured patches. The backward model ($L^*a^*b^*$ to *RGB*) is based on tetrahedral interpolation. An overview of this model is shown in Figure 2.

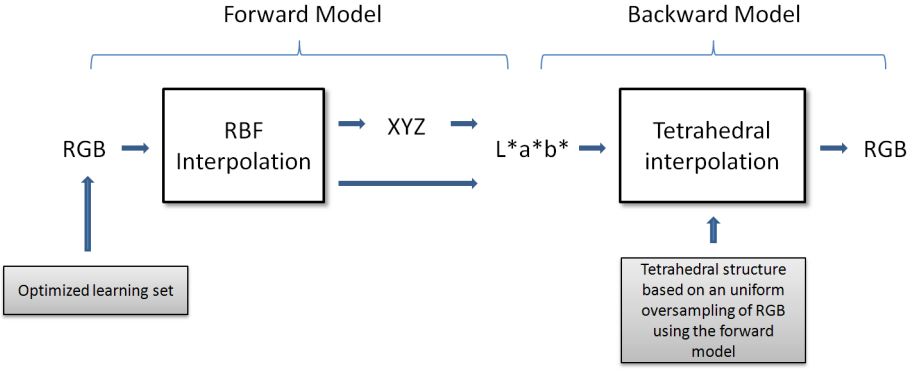

Fig. 2. Overview of the display color characterization model

2.2 Forward Model

Traditionally a characterization model (or forward model) is based on an interpolation or an approximation method. We found that radial basis function interpolation (RBFI) was the best model for our purpose.

RBF Interpolation. is an interpolation/approximation [1] scheme for arbitrarily distributed data. The idea is to build a function f whose graph passes

through the data and minimizes a bending energy function. For a general M-dimensional case, we want to interpolate a valued function $f(X) = Y$ given by the set of values $f = (f_1, ..., f_N)$ at the distinct points $X = x_1, ..., x_N \subset \Re^M$. We choose $f(X)$ to be a Radial Basis Function of the shape:

$$f(x) = p(x) + \sum_{i=1}^{N} \lambda_i \phi(||x - x_i||) \quad x \in \Re^M$$

where p is a polynomial, λ_i is a real-valued weight, ϕ is a basis function, $\phi : \Re^M \rightarrow \Re$, and $||x - x_i||$ is the euclidean norm between x and x_i. Therefore, a RBF is a weighted sum of translations of a radially symetric basis function augmented by a polynomial term. Different basis functions (kernel) $\phi(x)$ can by used.

Considering the color problem, we want to establish three three-dimensional functions $f_i(x, y, z)$. The idea is to build a function $f(x, y, z)$ whose graph passes through the tabulated data and minimizes the following bending energy function:

$$\iiint_{\Re^3} (f^3_{xxx} + f^3_{yyy} + f^3_{zzz} + 3f^3_{xxy} + 3f^3_{xxz} + 3f^3_{xyy} + 3f^3_{xzz} + 3f^3_{yyz} + 3f^3_{yzz} + 6f^3_{xyz}) dxdydz \tag{2}$$

For a set of data $\{(x_i, y_i, z_i, w_i)\}_{i=1}^n$ (where $w_i = f(x_i, y_i, z_i)$) the minimizing function is such as:

$$f(x, y, z) = b_0 + b_1 x + b_2 y + b_3 z + \sum_{j=1}^{n} a_j \phi(||(x - x_j, y - y_j, z - z_j)||) \tag{3}$$

where the coefficients a_j and $b_{0,1,2,3}$ are determined by requiring exact interpolation using the following equation

$$w_i = \sum_{j=1}^{n} \phi_{ij} a_j + b_0 + b_1 x_i + b_2 y_i + b_3 z_i \tag{4}$$

for $1 \leq n$ where $\phi_{ij} = \phi(||(x_i - x_j, y_i - y_j, z_i - z_j)||)$. In matrix form this is

$$h = Aa + Bb \tag{5}$$

where $A = [\phi_{ij}]$ is an $n \times n$ matrix and where B is an $n \times 4$ matrix whose rows $[1 \quad x_i \quad y_i \quad z_i]$. An additional implication is that

$$B^T a = 0 \tag{6}$$

These two vector equations can be solved to obtain

$$a = A^{-1}(h - Bb) \text{ and } b = (B^T A^{-1} B)^{-1} B^T A^{-1} h.$$

It is possible to provide a smoothing term. In this case the interpolation is not exact and becomes an approximation. The modification is to use the equation

$$h = (A + \lambda I)a + Bb \tag{7}$$

$$a = (A + \lambda I)^{-1}(h - Bb) \text{ and } b = (B^T(A + \lambda I)^{-1}B)^{-1}B^T(A + \lambda I)^{-1}h.$$

where $\lambda > 0$ is a smoothing parameter and I is the $n \times n$ identity matrix.

In our context we used a set of 4 real functions as kernel, the biharmonic ($\phi(x) = x$), triharmonic ($\phi(x) = x^3$), thin-plate spline 1 ($\phi(x) = x^2 log(x)$) and thin-plate spline 2 ($\phi(x) = x^2 log(x^2)$), with x the distance from the origin. The use of a given basis function depends on the display device which is characterized, and gives some freedom to the model.

Color Space Target. Our forward model uses $L^*a^*b^*$ as target ($L^*a^*b^*$ is a target well adapted for the gamut clipping that we use). This does not imply that we have to use $L^*a^*b^*$ as target for the RBF interpolation. In fact we have two choices. We can use either $L^*a^*b^*$ which seems to be the most logical target or XYZ associated with a XYZ to $L^*a^*b^*$ color transformation.

The use of different color spaces as target gives us another degree of freedom.

Smooth Factor Choice. Once the kernel and the color space target are fixed, the smooth factor, includes in the RBFI model used here, is the only parameter which can be used to change the properties of the transformation. With a zero value the model is a pure interpolation. With a different smooth factor, the model becomes an approximation. This is an important feature because it helps us to deal with the measurement problems due to the display stability (a color rendering for a given *RGB* value can change with the time) and to the measure repeatability of the measurement device.

2.3 Backward Model Using Tetrahedral Interpolation

While the forward model defines the relationship between the device "color space" and the CIE system of color measurement, we present in this section the inversion of this transform. Our problem is to find, for a $L^*a^*b^*$ values computed by the GPU from the multispectral image and the chosen illuminant, the corresponding *RGB* values (for a display device previously characterized).

This backward model could use the same interpolation methods previously presented but we used a new and more accurate method [3]. This new method uses the fact that if our forward model is very good then it is associated with an optimal patch database (see 2.4). Basically, we use a hybrid method; a tetrahedral interpolation associated with an over-sampling of the *RGB* cube (see Figure 3). We have chosen the tetrahedral interpolation method because of its geometrical aspect (this method is associated with our gamut clipping algorithm).

We build the initial tetrahedral structure using an uniform over sampling of the *RGB* cube ($n \times n \times n$ samples). This over sampling process uses the forward model to compute the corresponding structure in the $L^*a^*b^*$ color space. Once this structure is built, we can compute, for an unknown C_{Lab} color, the associated C_{RGB} color in two steps: First, the tetrahedron which encloses the point C_{Lab} to be interpolated should be found (the scattered point set is tetrahedrized); and then, an interpolation scheme is used within each tetrahedron.

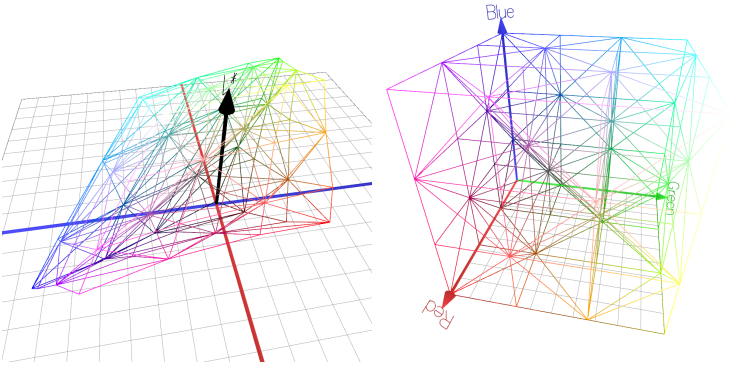

Fig. 3. Tetrahedral structure in $L^*a^*b^*$ and the correponding structure in *RGB*

More precisely, the color value C of the point is interpolated from the color values C_i of the tetrahedron vertices. A tri-linear interpolation within a tetrahedron can be performed as follows:

$$C = \sum_{i=0}^{3} w_i C_i$$

The weights can be calculated by $w_i = \frac{V_i}{V}$ with V the volume of the tetrahedron and V_i the volume of the sub-tetrahedron according to:

$$V_i = \frac{1}{6}(P_i - P)[(P_{i+1} - P)(P_{i+2} - P)]; i = 0, ..., 3$$

where P_i are the vertices of the tetrahedron and the indices are taken modulo 4.

The over-sampling used is not the same for each axis of *RGB*. It is computed according to the shape of the display device gamut in the $L^*a^*b^*$ color space. We found that than an equivalent to $36 \times 36 \times 36$ samples was a good choice. Using such a tight structure linearizes locally our model which becomes perfectly compatible with the used of a tetrahedral interpolation.

2.4 Optimized Learning Data Set

In order to increase the reliability of the model, we introduce a new way to determine the learning data set for the RBF based interpolation (e.g. the set of color patches measured on the screen). We found that our interpolation model was most efficient when the learning data set used to initialize the interpolation was regularly distributed in our destination color space ($L^*a^*b^*$). This new method is based on a regular 3D sampling of $L^*a^*b^*$ color space combined with a forward - backward refinement process after the selection of each patch. This algorithm allows us to find the optimal set of RGB colors to measure.

This technique needs to select incrementally the RGB color patches that will be integrated into the learning database. For this reason it has been integrated into a custom software tool which is able to drive a colorimeter. This software also measures a set of 100 random test patches equiprobably distributed in *RGB* used in order to determine the accuracy of the model.

2.5 Results

We want to find the best backward model which allows us to determine, with a maximum of accuracy, the *RGB* values for a computed *XYZ*. In order to complete this task we must define an accuracy criteria. We chose to multiply the average ΔE_{76} by the standard deviation (STD) of ΔE_{76} of the set of 100 patches evaluated with a forward model. This criteria makes sense because the backward model is built up on the forward model.

Optimal Model. The selection of the optimal parameters can be done using a brute force method. We compute for each kernels (ie. biharmonic, triharmonic, thin-plate spline 1, thin-plate spline 2), each color space target ($L^*a^*b^*$, *XYZ* and several smooth factors (0, 1e-005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1) the values of this criteria and we select the minimum.

For example the following table shows the report obtains for a SB2070 Mitsubishi DiamondPro with a triharmonic kernel for $L^*a^*b^*$ (Table 1) and *XYZ* (Table 2) as color space target (using a learning data set of 216 patches):

According to our criteria the best kernel is the triharmonic with a smooth factor of 0.01 and *XYZ* as target.

Table 1. Part of the report obtained in order to evaluate the best model parameters. The presented results are considering $L^*a^*b^*$ as target color space, and a triharmonic kernel for a CRT monitor SB2070 Mitsubishi DiamondPro.

smooth factor	0	0.0001	0.001	0.01	0.1
ΔE Mean	0.379	0.393	0.376	0.386	0.739
ΔE STD	0.226	0.218	0.201	0.224	0.502
ΔE Max	1.374	1.327	1.132	1.363	2.671
ΔE 95%	0.882	0.848	0.856	0.828	1.769
ΔRGB Mean	0.00396	0.00459	0.00438	0.00421	0.00826
ΔRGB STD	0.00252	0.00323	0.00316	0.00296	0.00728
ΔRGB Max	0.01567	0.02071	0.01768	0.01554	0.05859
ΔRGB 95%	0.00886	0.01167	0.01162	0.01051	0.01975

Table 2. Part of the report obtained in order to evaluate the best model parameters. The presented results are considering *XYZ* as target color space, and a triharmonic kernel for a CRT monitor SB2070 Mitsubishi DiamondPro.

smooth factor	0	0.0001	0.001	0.01	0.1
ΔE Mean	0.495	0.639	0.539	0.332	0.616
ΔE STD	0.293	0.424	0.360	0.179	0.691
ΔE Max	1.991	2.931	2.548	1.075	4.537
ΔE 95%	1.000	1.427	1.383	0.7021	1.751
ΔRGB Mean	0.00674	0.00905	0.00720	0.00332	0.00552
ΔRGB STD	0.00542	0.00740	0.00553	0.00220	0.00610
ΔRGB Max	0.02984	0.03954	0.03141	0.01438	0.04036
ΔRGB 95%	0.01545	0.02081	0.01642	0.00597	0.01907

The measurement process took about 5 minutes and the optimization process took 2 minutes (with a 4 cores processor). We reached our goal which was to provide an optimal model during a coffee break of the user.

Our different experimentation showed that a 216 patches learning set was a good compromise (equivalent to a $6\times6\times6$ sampling of the *RGB* cube). A smaller data set gives us a degraded accuracy, a bigger gives us similar results because we are facing the measurement problems introduced previously.

Optimized Learning Data Set. Table 3 and Table 4 show the results obtained with our model for two displays of different technologies. These tables show clearly how the optimized learning data set can produce better results with the same number of patches.

Table 3. Accuracy of the model established with 216 patches in forward and backward direction for a LCD Wide Gamut display (HP2408w). The distribution of the patches plays a major role for the model accuracy.

	Forward model		Backward model	
	ΔE Mean	ΔE Max	ΔRGB Mean	ΔRGB Max
Optimized	1.057	4.985	0.01504	0.1257
Uniform	1.313	9.017	0.01730	0.1168

Table 4. Accuracy of the model established with 216 patches in forward and backward direction for a CRT display (Mitsubishi SB2070). The distribution of the patches plays a major role for the model accuracy.

	Forward model		Backward model	
	ΔE Mean	ΔE Max	ΔRGB Mean	ΔRGB Max
Optimized	0.332	1.075	0.00311	0.01267
Uniform	0.435	1.613	0.00446	0.01332

Table 5. Accuracy of the model established with 216 patches in forward and backward direction for three other displays. The model performs well on all monitors.

	Forward model		Backward model	
	ΔE Mean	ΔE Max	ΔRGB Mean	ΔRGB Max
EIZO CG301W (LCD)	0.783	1.906	0.00573	0.01385
Sensy 24KAL (LCD)	0.956	2.734	0.01308	0.06051
DiamondPlus 230 (CRT)	0.458	2.151	0.00909	0.06380

Results for Different Displays. Table 5 presents different results obtained for 3 others displays (2 LCD and 1 CRT).

Considering that non trained humans can not discriminate ΔE less than 2, we can see here that our model gives very good results on a wide range of display.

2.6 Gamut Mapping

The aim of gamut mapping is to ensure a good correspondence of overall color appearance between the original and the reproduction by compensating for the mismatch in the size, shape and location between the original and reproduction gamuts.

The $L^*a^*b^*$ computed color can be out of gamut (i.e. the destination display cannot generate the corresponding color). To ensure an accurate colorimetric rendering, considering $L^*a^*b^*$ color space, and low computational requirements, we used a geometrical gamut clipping method based on the pre-computed tetrahedral structure (generated in our backward model) and more especially on the surface of this geometrical structure (see figure.3).

The clipped color is defined by the intersection of the gamut boundaries and the segment between a target point and the input color. The target point used here is an achromatic $L^*a^*b^*$ color with a luminance of 50.

3 GPU-Based Implementation

Our color management method is based on a conversion process which will compute for a XYZ values the corresponding RGB.

It is possible to implement the presented algorithm with a specific GPU language, like CUDA, but our application will only works with CUDA compatible GPU (nVIDIA^{TM} G80, G90 and GT200). Our goal was to have a working application on a large number of GPU (AMD and nVIDIA^{TM} GPUs), for this reason we choose to implement a classical method using a 3D lookup table.

During an initialization process we build a three dimensional $RGBA$ floating point texture which cover the $L^*a^*b^*$ color space. The alpha channel of the $RGBA$ values saves the distance between the initial $L^*a^*b^*$ value and $L^*a^*b^*$ value obtained after the gamut mapping process. If this value is 0 the $L^*a^*b^*$ color which will have to be converted is in the gamut of the display otherwise this color is out gamut and we are displaying the closest color (according to our gamut mapping process). This allows us to display in real time the color errors due to the screen inability to display every visible colors.

Finaly our complete color pipeline includes: a reflectance to XYZ conversion then a XYZ to $L^*a^*b^*$ conversion (using the white of the screen as reference) and our color management process based on the 3D lookup table associated with a tri-linear interpolation process.

4 Conclusion

We presented a part of a large multispectral application used at the C2RMF. It has been shown that it is possible to implement an accurate color management process even for a real time color reconstruction. We showed a color management process based only on colorimetric consideration. The next step is to introduce a color appearance model in our color flow. The use of such color appearance model, built up on our accurate color management process, will allows us to do virtual exhibition of painting.

References

[1] Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and Representation of 3D Objects with Radial Basis Functions. In: SIGGRAPH, pp. 12–17 (2001)

[2] Colantoni, P., Boukala, N., Da Rugna, J.: Fast and Accurate Color Image Processing Using 3D Graphics Cards. In: Vision Modeling and Visualization, VMV 2003, pp. 383–390 (2003)

[3] Colantoni, P., Stauder, J., Blond, L.: Device and method for characterizing a colour device Thomson Corporate Research, European Patent, EP 05300165.7 (2005)

[4] Ribés, A., Schmitt, F., Pillay, R., Lahanier, C.: Calibration and Spectral Reconstruction for CRISATEL: an Art Painting Multispectral Acquisition System. Journal of Imaging Science and Technology 49, 563–573 (2005)

[5] Bastani, B., Cressman, B., Funt, B.: An evaluation of methods for producing desired colors on CRT monitors. Color Research & Application 30, 438–447 (2005)

[6] Colantoni, P., Pitzalis, D., Pillay, R., Aitken, G.: GPU Spectral Viewer: analysing paintings from a colorimetric perspective. In: The 8th International Symposium on Virtual Reality, Archaeology and Cultural Heritage, Brighton, United Kingdom (2007)

[7] `http://www.gpgpu.org`