Color Selection Algorithm Design for Smart Lighting Application

S. Vinay Kumar#, MSV Sashi Kumar#, Pamidighantam V Ramana#
#Vasavi College of Engineering,
Hyderabad.

***Abstract* --Light-emitting diodes (LEDs) have numerous advantages over conventional lamps or compact fluorescent lamps (CFLs), such as long lifetime, large color gamut, small size, and absence of mercury vapour. Lamps based on high brightness red, green and blue (RGB) light-emitting diodes (LEDs) can produce light of nearly any color. Since LEDs have a short rise and fall times, such array of RGB and white LEDs can be controlled electronically by means of varying the duty cycle of ON time to generate a particular color. Thus, the resultant lamp can be called as a "smart lamp" or intelligent lamp. The ability to change the color point of the lamp provides a new feature to general illumination that has the potential to generate new applications.**

In this manuscript, we present our work on developing an algorithm to select a particular color from an RGB palette on a hand held appliance like a cellular phone. The objective is to design a control code that requires least amount of resources in a mobile platform. The description of human color vision and measurement of color in lighting applications are traditionally defined by the CIE tristimulus values, XYZ whereas RGB is a way of specifying color and a way of viewing color in graphics. The chosen color is converted into CIE chromaticity data and the RGB LED intensity is computed to produce a color mixer to yield the selected color. The color palette and the software are written in Java and Android.

Keywords—Intelligent lighting; RGB LEDs; Color control; Luminaire

I. INTRODUCTION

The developments in LED components over the past few years have resulted in generating a new market for illumination engineering. Nowadays, high power LEDs of the order of 3 to 5 Watts are available with very high luminous efficiency in a very small form factor package. There are several approaches of using LEDs to generate lighting products. The advantages of LED lighting include long life, high reliability, stable poweroutput, high luminous efficiency.

We focus our work on the use of Red, Green, Blue and white LED combinations to produce a light source that can have a variable color point while producing relatively high luminous efficiency.

The present manuscript describes our efforts at developing an algorithm to interlink RGB and XYZ tristimulus values for a high brightness RGBW LED luminaire. We studied different algorithms for conversion and compared them. The LED X and Y coordinates are controlled by keeping the input current to the LED constant. A PWM circuit is generated so that the average LED intensity can be varied. Section II provides a short summary of different algorithms for conversion from RGB to XYZ. Section II describes colors in computers. Section IIIdescribes conversion mechanism from RGB to XYZ. Section IV provides analysis of algorithms for a set of inputs and describes our efforts at generating a huge database of RGB to XYZ look up table for practical implementation.Section V gives the summary.

II. COLOR IN COMPUTERS AND IN LIGHTING

The description of human color vision and measurement of color in lighting applications are traditionally defined by the CIE tristimulus values, XYZ whereas RGB is a way of specifying color and a way of viewing color in graphics. Graphics algorithms manipulate RGB colors, and the images produced by graphics algorithms are encoded as RGB pixels and displayed on devices that render these pixels by emitting RGB light. However, the co-relation between RGB pixel depicted on a digital display device and the tristimulus value is not easily understood.

Color point reproducibility is a major challenge for non-LED light sources and for phosphor based LED sources. The advantages of RGBW LED combination is that the luminaire can have any color point over the color coordinate chart by proper selection of the relative intensity and predefined color rendering index (CRI). In speciality applications, the color point selected should remain fixed over time as variation of color point will result in change of hue and perception of different color. The color point of a LED lamp could vary due to the change in the peak wavelength due to change in junction temperature of LED, change in operating current or due to aging.

The human eye has a highly nonlinear spectral repose as shown in Figure 1 [1]. The eye has a different response for photopic stimulus (daylight) and scotopic stimulus (night light) and both the responses are highly nonlinear [4]. The color point plot is a result of the response of human vision to an incident stimulus and is a variable with spectral wavelength. This is because any spectral distribution is a weighted sum of monochromatic colors. The spectrum created by combining the sources is the sum of their individual spectra.

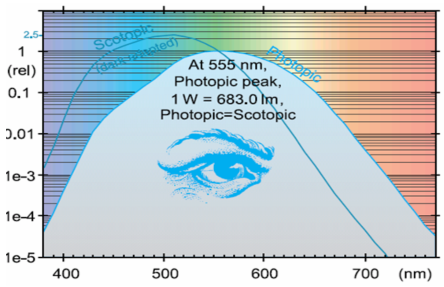

Figure 1. The spectral response of human eye (bright light and dark light conditions)

The color point of a source can be represented by RGB coefficients as in computer graphics (Figure 2) or XYZ as in chromaticity applications (Figure 3). Most of the computer and graphics designers are familiar with RGB terminology while the component manufacturers are familiar with the CIE chromaticity terminology. Color is created by mixing of the light emitted by Red, Green and Blue LEDs [5] as shown in Figure 2. The RGB components mix and produce the pixel color value, uniquely defining the color. Given an identical display, the same RGB values should result in same color on the display.

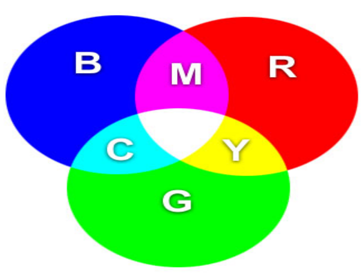

Figure 2. RGB color palette and production of different hues

The CIE tristimulus plot on the other hand is a standard set of color matching functions that form the basis for most of the color and chromaticity measurement instruments existing today. The CIE chromaticity diagram [6] is shown in figure 3. The CIE color stimulus is a function of source wavelength and color correlation temperature of the source. The tristimulus values are additive and scalable and hence are useful information to set the color of a given LED chip.

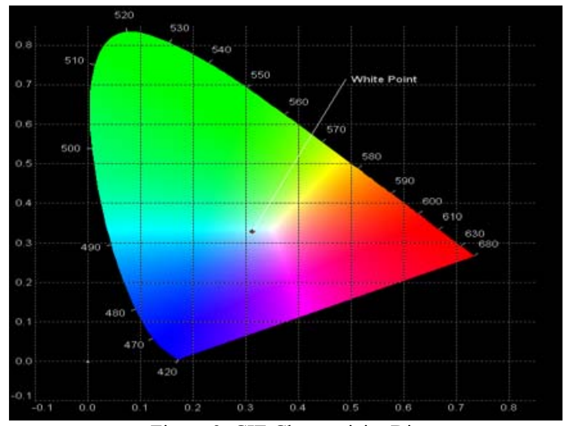

Figure 3. CIE Chromaticity Diagram

III. CONVERSION FROM RGB TO XYZ

Colored LEDs are high brightness devices with a narrow spectral width. The Chromaticity coordinates of a given LED can be extracted from the CIE chromaticity curve. The chromaticity coordinates of the selected LEDs are mentioned in Table I.

TABLE I. THE WAVELENGTH, LUNINOUS FLUX AND CIE X, Y AND Z FOR SELECTED LED'S

	Flux (lumens) at 350 mA	**Wave length**	**Spectral Half Width**	**X**	**Y**	**Z**
R	58	625	18	0.68943	0.31041	0.00016
G	93	528	33	0.21309	0.6897	0.09720
B	28	467	25	0.10458	0.33695	0.55846
W	152	--	--	0.31675	0.33695	0.3463

When, colored LEDs of different colors are driven together, the superposed chromaticity value can be derived as a function of the individual LED flux, chromaticity coordinates and the number of LEDs per each color. Thus,

X value of the RGB lamp =
(Nr.Xr.Fr + Ng.Xg.Fg + Nb.Xb.Fb + Nw.Xw.Fw) /
(Nr.Fr + Ng.Fg + Nb.Fb + Nw.Fw)
Y value of the RGB lamp =
(Nr.Yr.Fr + Ng.Yg.Fg + Nb.Yb.Fb + Nw.Yw.Fw) /
(Nr.Fr + Ng.Fg + Nb.Fb + Nw.Fw) and
Z value of RGB lamp = 1 - X - Y

Where

N r, g, b and w are the number of LEDs for each color.
X and Y r,g,b and w are the chromaticity values for each color LED.
F r,g,b and w are the total lumens for each color.

The luminous flux F (r,g,b,w) of an LED can be written as a function of constant LED current If, PWM ON pulse time T(r,g,b,w), a linear slope factor for flux vs current and a constant offset value as given by:

F(r, g, bw) = F(r, g, b, w@350mA) *
((If/0.35) * (T(r, g, bw)/Tperiod) * m - c)

Thus, the cumulative performance of RGBW luminaire can be evaluated. The X, Y and Z chromaticity needs to be converted into a suitable RGB value so that the LED color can be programmed.

The method to characterize an RGB color pixel value from XYZ tristimulus value is done through a 3x3 matrix linear transformation as shown in equation 1 below [3].

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Where

$$[M] = \begin{bmatrix} SrXr & SgXg & SbXb \\ SrYr & SgYg & SbYb \\ SrZr & SgZg & SbZb \end{bmatrix}$$

$$Xr = Xr / Yr$$
$$Yr = 1$$
$$Zr = (1 - Xr - Yr) / Yr$$
$$Xg = Xg / Yg$$
$$Yg = 1$$
$$Zg = (1 - Xg - Yg) / Yg$$
$$Xb = Xb / Yb$$
$$Yb = 1$$
$$Zb = (1 - Xb - Yb) / Yb$$

$$\begin{bmatrix} Sr \\ Sg \\ Sb \end{bmatrix} = \begin{bmatrix} Xr & Xg & Xb \\ Yr & Yg & Yb \\ Zr & Zg & Zb \end{bmatrix} \begin{bmatrix} Xw \\ Yw \\ Zw \end{bmatrix}$$

The critical element in the computation of the RGB coefficients is the 3x3 matrix. We compared four methods, viz., Delphi method, Stack Overflow method and Lindbloom method to generate the X, Y and Z values for each RGB palette selected.

IV. ANALYSIS OF VARIOUS ALGORITHMS

We have done the analysis of Delphi Method [2] and Stack Overflow Method.

A. Algorithm 1 – Delphi Implementation

d := xr * yg * zb - xg * yr * zb - xr * yb * zg + xb * yr * zg +
xg * yb * zr - xb * yg * zr;
R := (-xg * yc * zb + xc * yg * zb + xg * yb * zc - xb * yg * zc -
xc * yb * zg + xb * yc * zg) / d;
G := (xr * yc * zb - xc * yr * zb - xr * yb * zc + xb * yr * zc +
xc * yb * zr - xb * yc * zr) / d;
B := (xr * yg * zc - xg * yr * zc - xr * yc * zg +
xc * yr * zg + xg * yc * zr - xc * yg * zr) / d

The following Table.II demonstrates the output generated for the above algorithm.

TABLE II. SAMPLE OUTPUT OF DELPHI

X	Y	Z	R	G	B
0.7	0.21	0.09	255	0	-3
0.295	0.74	-0.035	-1	255	-31

0.005	0.05	0.945	1	-4	240
0.31675	0.33695	0.3463	79	88	72
0.349688	0.374659	0.275653	86	101	53
0.431548	0.298924	0.269528	134	62	50

B. Algorithm 2 – Stack Overflow Method

var_X = X / 100 //X from 0 to 95.047
// (Illuminant = D65)
var_Y = Y / 100 //Y from 0 to 100.000
var_Z = Z / 100 //Z from 0 to 108.883
var_R = var_X * 3.2406 + var_Y * -1.5372 + var_Z * -0.4986
var_G = var_X * -0.9689 + var_Y * 1.8758 + var_Z * 0.0415
var_B = var_X * 0.0557 + var_Y * -0.2040 + var_Z * 1.0570
R = var_R * 255
G = var_G * 255
B = var_B * 255

The following table Table.III demonstrates the output generated for the above algorithm.

TABLE III. SAMPLE OUTPUT OF STACK OVERFLOW METHOD

X	Y	Z	R	G	B
0.263524	0.290922	0.445553	238	232	170
0.237596	0.274535	0.487869	79	88	72
0.244401	0.308853	0.446746	86	101	53
0.413961	0.298406	0.287633	134	62	50

C. Algorithm3 – Lindbloom Implementation

a) Set Nr =5, N g = 4, Nb=4, Nw=5

b) Compute LED flux Fr, Fg, Fb, Fw considering the following

	X	Y	Z
R	0.68943	0.31041	0.00016
G	0.213095	0.6897	0.097205
B	0.104585	0.33695	0.558465
W	0.31675	0.33695	0.3463

c) Set Tr=0 to 1.0, step 0.05

d) Set Tg = 0 to 1.0, step 0.05

e) Set Tb = 0 to 1.0, step 0.05

f) Set Tw = 0 to 1.0, step 0.05

g) Ignore if Tr + Tg + Tb + Tw > 1.0

h) Compute

Xlamp = (Nr.Xr.Fr + Ng.Xg.Fg + Nb.Xb.Fb + Nw.Xw.Fw)/ (Nr.Fr + Ng.Fg + Nb.Fb + Nw.Fw)

i) Compute

Ylamp = (Nr.Yr.Fr + Ng.Yg.Fg + Nb.Yb.Fb + Nw.Yw.Fw) / (Nr.Fr + Ng.Fg + Nb.Fb + Nw.Fw)

j) Find a match with the X, Y & Z output from RGB code earlier

k) Write a Table of X, Y and Z, RGB &Tr, Tb, Tg, Tw values.

TABLE IV. SAMPLE OUTPUT OF LINDBLOOM METHOD

Flux total	**X**	**Y**	**Z**	**Rint**	**Gint**	**Bint**	**Tr**	**Tg**	**Tb**	**Tw**	**Color Palette**
151.4032	0.7	0.295	0.005	243	10	0	0.65	0	0	0	

175.7	0.641	0.328	0.03	223	25	5	0.65	0.05	0.05	0	
276.0395	0.404	0.37	0.226	122	79	52	0.35	0.15	0.15	0.2	
301.973	0.417	0.466	0.117	120	107	26	0.45	0.35	0.05	0.1	
289.0915	0.398	0.395	0.207	116	89	48	0.35	0.2	0.1	0.2	
174.7813	0.373	0.269	0.358	114	53	86	0.25	0.05	0.6	0.05	
278.8507	0.365	0.43	0.205	94	109	50	0.25	0.25	0.05	0.2	
181.2069	0.325	0.485	0.189	64	141	48	0.15	0.25	0.2	0.05	
117.4378	0.282	0.237	0.481	61	63	129	0.05	0	0.4	0.1	
152.8088	0.3	0.369	0.331	59	107	88	0.15	0.15	0.55	0	
338.4333	0.3	0.622	0.077	31	203	19	0.2	0.7	0	0.05	
166.1166	0.238	0.289	0.473	23	95	136	0	0.05	0.5	0.15	
90.36	0.194	0.118	0.689	8	38	207	0.05	0	0.7	0	
323.4283	0.242	0.495	0.262	1	177	75	0	0.45	0.3	0.2	

Among the above three implementations the third algorithm based on Lindbloom proved to be promising for our approach. The above table Table.IV gives the look-up table generated from RGB to XYZ based on Timing signalsTr, Tf, Tb & Tw.

V. ANDROID APP

To validate the results generated with our algorithm, we have developed an android app. The algorithm will calculate the XYZ tristimulus from the selected color shown on the app. To build this, we have drawn a color wheel, which enables the user to select a particular color. The color wheel was built by drawing two concentric circles. The outer circle was filled up with the color span across the colors available spectrum and the inner circle is used for displaying the selected color.

Steps involved in app development

1. Design the basic layout of the application, with 2 concentric circles, outer circle used for selecting the color & inner circle displaying the selected color, configuring activity_main.xml
2. Fill the outer circle while overriding onDraw() method, calculating the radius of the circle, filling the circle with a set of colors, using Shader class.
3. Overriding the onTouchEvent() method present in the View class, read the selected color from the ArcTangent value from the color circle, when the user does a ACTION_MOVE to get the sweep angle of selection. Then convert the angle into unit by computing unit = (angle/2π)
4. Interpolating the color across the moved region while multiplying the units with the initial color of the sweep to get the fractional constant.
5. Use step 4 to multiply with the difference of the R, G& B component of the selected colors performing the sweep and add it to the base color value of the sweep.
6. Finally return a color-int from alpha, red, green, blue components from step 5.

7. Use this color to highlight the inner circle, which is the selected color from the color wheel.

When the App is run the home screen is shown as in Figure 4. Figure 5 shows the display of the equivalent XYZ for the selected RGB.

Figure 4. App home screen

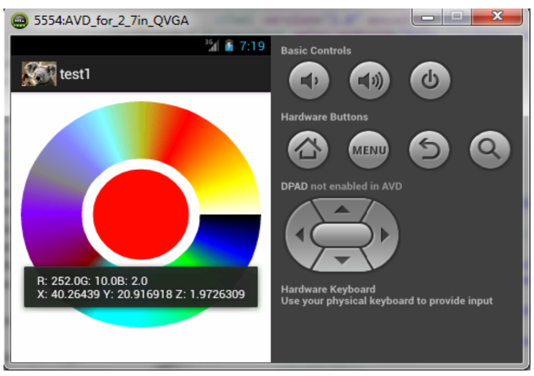

Figure 5. Selection of color in App

Using the Lindbloom method, we have derived the results and the tristimulus is retrieved from the app which is verified fromfewtrusted sources [6].

Android app occupies 636KB space and generates response within 350ms.

VI. SUMMARY

We have presented our work on the design of an algorithm for controlling an RGBW LED luminaire using Pulse Width Modulated constant driving current to generate a selected color out of RGB color wheel. The algorithm consists of

a. Programmable luminous flux,

b. Number of LEDs,

c. Programmable pulse width

d. Cumulative luminaire luminous flux

Thus, the algorithm is suitable for applying for various configurations. The algorithm computes the maximum flux for a given pulsing condition.

We compared four methods of X, Y Z to RGB conversion and found that Lindbloom with normalization suits our algorithm the most. The algorithm has been validated through a number of selected X, Y and Z and RGB palettes. The hardware development is in progress.

ACKNOWLEDGEMENT

The authors wish to express their gratitude to the management, Principal and Head Department of Computer Science of Vasavi College of Engineering for their constant encouragement and guidance.

REFERENCES

[1] SubramaniamMuthu, et al., "Red, Green, and Blue LEDs for White Light Illumination", IEEE journal on selected topics in Quantum Electronics, Vol. 8, No. 2, March/April 2002.

[2] http://www.fourmilab.ch/documents/specrend

[3] H. Nam., A color compensation algorithm to avoid color distortion in LCD displays, IEEE Transactions on Consumer Electronics, Vol. 56, No.4, November, 2010.

[4] Light Measurement Handbook

[5] http://www.arch.virginia.edu/arch569/content/lectues/lec-05/images/rgb-circles.jpg

[6] http://www.brucelindbloom.com/index.html?Eqn_RGBXYZ_Matrix.html

[7] Fu Cheng Wang, et al., Multivariable Robust Control for a Red–Green–Blue LED Lighting System, IEEE Transactions on Power Electronics, Vol. 25, No. 2, Feb 2010.