RC 21930 (98607) 11 January 2001

Computer Science

IBM Research Report

Implementation of a Color Calibration Method for Liquid Crystal Displays

Albert Cazes, Fred Mintzer
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Implementation of a Color Calibration Method for Liquid Crystal Displays

Albert Cazes, Fred Mintzer

IBM T.J.Watson Reserch Center
Yorktown Heights, NY 10598

ABSTRACT

In this paper we describe the implementation of a color calibration method for liquid crystal displays (LCD) where we assume that the angle of vision is normal to the center of the screen. We change the procedure generally used with cathode ray tubes (CRT) and take into account the high intensity black of an LCD, the variable chromaticities of its primaries and the less than perfect additivity of its colors. This is done by modifying the calibration data and using correction tables. We describe three rendering algorithms which can be used with this implementation. We test them on four IBM LCD models produced in the last four years and conclude that the simplest algorithm is adequate for the latest model.

Keywords: TFT/LCD, Liquid crystal display, display color calibration, monitor color calibration, LCD, cathode ray tube, CRT.

1. INTRODUCTION

The color calibration of LCD monitors[6,9] usually employs methods developed for the CRT which are fully described in the literature[1,2,4,5,7,8]. These methods generally assume that the display satisfies the following conditions:

1. The luminance of a primary is monotonically increasing with respect to its index value. For example, if the display range is set from 0 to 255, then red, green and blue will respectively reach their maximum luminances for display RGB vectors (255, 0, 0), (0, 255, 0) and (0, 0, 255).
2. Colors within the display gamut are additive; the CIE XYZ vector corresponding to a uniform color patch with display RGB coordinates (R, G, B) must be equal to the sum of the CIE XYZ vectors corresponding to the three primary display RGB vectors: (R, 0, 0), (0, G, 0) and (0, 0, B).
3. The CIE chromaticities xyz of the red, green and blue primaries are independent of their luminance levels: for example the CIE xyz vector corresponding to display RGB vectors (255, 0, 0) or (100, 0, 0) must be the same.
4. The display has a high contrast; the ratio of the luminance (CIE Y) of white to that of black must be a large number.
5. The color of a pixel does not depend on the angle of view of the observer.

In a previous paper[3], presented at this conference in 1999, we compared the color characteristics of a CRT with those of an LCD and showed that the former almost satisfied the previous five conditions while the latter had several deficiencies. The chromaticities of its primaries depended on their luminance levels. An LCD unit we tested reached its maximum blue luminance at level 212 instead of 255. Its black luminance was about 0.5 nits, compared to 0.005 nits for a typical CRT, which created a low contrast ratio and finally colors changed with the angle of view of the observer. We also noticed that if we subtracted the CIE XYZ value of black from our calibration measurements, then we could correct some of these problems.

In this paper we describe the implementation of a color calibration method for an LCD where we assume that the angle of view is perpendicular to the center of the screen. First, we determine the useful ranges of the primaries, where their luminance curves behave monotonically. Then, we divide these ranges into an equal number of intervals. For each one of these intervals, we measure the CIE XYZ values of four color patches: red, green, blue and their corresponding gray. The CIE XYZ

value for the black of this display is subtracted from all these measurements, a 3x3 matrix is created and used to convert the resulting data to linear RGB units. Then, we compute two types of error tables: the nonlinear tables which take into account the errors produced by the variable chromas of the primaries and the leakage tables which attempt to correct the color additivity problem. Finally, we map the linear RGB values for each primary into the corresponding display RGBs and obtain the gamma tables. We describe three algorithms which can be used to render an image on the display:

1 . The remove black algorithm (Alg_B) which does not use the error or leakage tables.
2 . The remove black and nonlinear errors algorithm (Alg_BN) which does not use the leakage tables.
3 . The remove black, nonlinear and leakage errors algorithm (Alg_BNE).

Of the three, Alg_B is the one which requires the least computations. We test these algorithms on four desktop IBM LCD models which were marketed during the last four years and observe the color improvements which took place in this period.

2. An Implementation Method.

In this section we describe the implementation of a color calibration method for an LCD. All our measurements were done using a Gamma Scientific RadomaCam model GS 1280 spectroradiometer. The front of its lens was placed 50 cms from the display, its optical axis was perpendicular to the center of the screen, the maximum integration time was 15 seconds and the aperture angle was fixed at two degrees. Each measurement consisted of finding the CIE XYZ value of a uniform colored 5 cm by 5 cm patch displayed at the center of the screen. We developed software which could automatically perform the following functions:

- Measure color patches and file the results.
- Calibrate the display and create a profile.
- Validate the calibration using one of the three rendering algorithms.

2.0 Notation

In what follows, vectors will be represented in lower case bold, arrays in upper case bold and scalars in lower case regular characters. The initial index for vectors or arrays is zero.Thus, $\mathbf{v}_2$ represents the third element of vector $\mathbf{v}$ and $\mathbf{A}_{3,0}$ refers to the element in the fourth row and first column of array $\mathbf{A}$. $\mathbf{A}^{<2>}$ represents the third column of array $\mathbf{A}$, and $\mathbf{A}^T$ the transpose of the same array. The first row of $\mathbf{A}$ is written as $\mathbf{A}^{T<0>}$. Prefixes are used with the names of vectors or arrays to describe the type of units used:

- **D_** stands for display RGB, in our case this will range between 0 and 255,
- **C_** stands for CIE XYZ which are nonnegative floating point numbers.
- **M_** stands for linear RGB, these floating point numbers are obtained by multiplying a 3x3 matix by a CIE XYZ vector.

2.1 Measurements.

For old LCD models we first have to determine for each primary the part of the luminance curve which is monotonically increasing with respect to its display RGB level. This is done by rendering patches at several levels and measuring their CIE XYZ values. The number of samples per measurement depends on the stability of the backlamp and electronics of the display. For the oldest IBM LCD model in our case study, the **9516A03,** we obtained "useful" display RGB ranges of [0, 255], [0, 255] and [0, 241] for red, green and blue respectively. We also had to use several samples per measurement because the coefficient of variation (CV) for this unit was close to 3%. We found that this step was not necessary with the newer models, their luminance curves were monotonic for the full [0, 255] range and their CVs were less than 0.6 %.

Next, we select an integer N at least greater than 16, for our tests we took N equal to 64, divide each one the three monotonic ranges into N-1 equal parts and obtain the N sample points for each primary. These are used to define the four 3xN display RGB arrays for red, green, blue and related grays:

D_Red, D_Green, D_Blue, D_Gray

where the related gray for (251, 0, 0), (0, 251, 0), (0, 0, 238) is (251,251,238). The 4xN column vectors of the above arrays are used to sequentially display uniform colored patches at the center of the screen which are measured with the spectroradiometer. From this operation we obtain the corresponding four 3xN CIE XYZ arrays:

C_Red, C_Green, C_Blue, C_Gray

These two sets of arrays are stored in a file for further processing.

2.2 Calibration

In this section, we will describe the processing of the measurements and the creation of the correction or gamma tables. We will also list the steps required for the three rendering algorithms.

2.2.1 Processing the measurements.

- First, we compute the average CIE XYZ black vector:
  $\mathbf{C\_k} = 0.25 * (\mathbf{C\_Red}^{<0>} + \mathbf{C\_Green}^{<0>} + \mathbf{C\_Blue}^{<0>} + \mathbf{C\_Gray}^{<0>})$ (1)
  Then, we subtract it from all the column vectors of the CIE XYZ measurement arrays and obtain the four 3xN arrays **C_Redk, C_Greenk, C_bluek, C_Grayk** such that for index i ranging from 0 to N-1 their column vectors are defined by:
  $\mathbf{C\_Redk}^{<i>} = \mathbf{C\_Red}^{<i>} - \mathbf{C\_k}$
  $\mathbf{C\_Greenk}^{<i>} = \mathbf{C\_Green}^{<i>} - \mathbf{C\_k}$
  $\mathbf{C\_Bluek}^{<i>} = \mathbf{C\_Blue}^{<i>} - \mathbf{C\_k}$
  $\mathbf{C\_Grayk}^{<i>} = \mathbf{C\_Gray}^{<i>} - \mathbf{C\_k}$

- Using the last column vectors of **C_Redk, C_Greenk** and **C_Bluek**, we create the display matrix **Md** :

$$\mathbf{Md} = \begin{pmatrix} \mathbf{C\_Redk}_{0,N-1} & \mathbf{C\_Greenk}_{0,N-1} & \mathbf{C\_Bluek}_{0,N-1} \\ \mathbf{C\_Redk}_{1,N-1} & \mathbf{C\_Greenk}_{1,N-1} & \mathbf{C\_Bluek}_{1,N-1} \\ \mathbf{C\_Redk}_{2,N-1} & \mathbf{C\_Greenk}_{2,N-1} & \mathbf{C\_Bluek}_{2,N-1} \end{pmatrix} \quad (2)$$

  Then we compute its inverse $\mathbf{Md}^{-1}$.

- We multiply $\mathbf{Md}^{-1}$ by the arrays **C_Redk, C_Greenk, C_Bluek** and **C_Grayk** and obtain the four 3xN linear RGB arrays: **M_Red, M_Green, M_Blue, M_Gray** :
  $\mathbf{M\_Red} = \mathbf{Md}^{-1} * \mathbf{C\_Redk}$
  $\mathbf{M\_Green} = \mathbf{Md}^{-1} * \mathbf{C\_Greenk}$
  $\mathbf{M\_Blue} = \mathbf{Md}^{-1} * \mathbf{C\_Bluek}$
  $\mathbf{M\_Gray} = \mathbf{Md}^{-1} * \mathbf{C\_Grayk}$ (3)

- We decompose the 3xN array **M_Gray** into three 3xN arrays **M_RedGray, M_GreenGray, M_BlueGray**. For index i ranging from 0 to N-1, their column vectors are defined as follows:

$$\mathbf{M\_RedGray}^{\langle i \rangle} = \begin{pmatrix} \mathbf{M\_Gray}_{0,i} \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{M\_GreenGray}^{\langle i \rangle} = \begin{pmatrix} 0 \\ \mathbf{M\_Gray}_{1,i} \\ 0 \end{pmatrix} \quad \mathbf{M\_BlueGray}^{\langle i \rangle} = \begin{pmatrix} 0 \\ 0 \\ \mathbf{M\_Gray}_{2,i} \end{pmatrix}$$

  By construction we obtain the following equality between the 3xN arrays:
  $\mathbf{M\_Gray} = \mathbf{M\_RedGray} + \mathbf{M\_GreenGray} + \mathbf{M\_BlueGray}$ (4)

The next steps are only necessary if the rendering algorithms Alg_BN or Alg_BNE are selected:

- We sum the three arrays **M_Red, M_Green** and **M_Blue** to obtain the 3xN linear RGB array **M_SumPrim**:
  $\mathbf{M\_SumPrim} = \mathbf{M\_Red} + \mathbf{M\_Green} + \mathbf{M\_Blue}$ (5)
  We subtract from the result **M_Gray** and obtain the 3xN linear RGB array **M_SumLeak**, which represents the leakage or difference between the sum of the primaries and their corresponding grays:
  $\mathbf{M\_SumLeak} = \mathbf{M\_SumPrim} - \mathbf{M\_Gray}$ (6)

- We decompose the 3xN array **M_SumLeak** into three 3xN arrays **M_RedLeak, M_GreenLeak, M_BlueLeak**. For index i ranging from 0 to N-1, their column vectors are defined as follows:

$$\mathbf{M\_RedLeak}^{\langle i \rangle} = \begin{pmatrix} \mathbf{M\_SumLeak}_{0,i} \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{M\_GreenLeakG}^{\langle i \rangle} = \begin{pmatrix} 0 \\ \mathbf{M\_SumLeak}_{1,i} \\ 0 \end{pmatrix}$$

$$\mathbf{M\_BlueLeakG}^{\langle i \rangle} = \begin{pmatrix} 0 \\ 0 \\ \mathbf{M\_SumLeak}_{2,i} \end{pmatrix}$$

Also, by construction we obtain the following equality between these 3xN arrays:

$$\mathbf{M\_SumLeak} = \mathbf{M\_RedLeak} + \mathbf{M\_GreenLeak} + \mathbf{M\_BlueLeak} \tag{7}$$

- We compute the 3xN arays **M_RedError, M_GreenError, M_BlueError:**

$$\mathbf{M\_RedError} = \mathbf{M\_Red} - \mathbf{M\_RedGray} - \mathbf{M\_RedLeak} \tag{8}$$

$$\mathbf{M\_GreenError} = \mathbf{M\_Green} - \mathbf{M\_GreenGray} - \mathbf{M\_GreenLeak} \tag{9}$$

$$\mathbf{M\_BlueError} = \mathbf{M\_Blue} - \mathbf{M\_BlueGray} - \mathbf{M\_BlueLeak} \tag{10}$$

- The 3xN array **M_SumError** is defined as the sum of **M_RedError, M_GreenError, M_BlueError:**

$$\mathbf{M\_SumError} = \mathbf{M\_RedError} + \mathbf{M\_GreenError} + \mathbf{M\_BlueError}$$

By substituting equations (8), (9), (10) then using (4), (5), (7) we obtain:

$$\mathbf{M\_SumError} = \mathbf{M\_SumPrim} - \mathbf{M\_Gray} - \mathbf{M\_SumLeak}$$

then using equation (2) we get the 3xN array **M_AllZeros** where all elements are zeros:

$$\mathbf{M\_SumError} = \mathbf{M\_AllZeros}$$

The model we are using can be summarized by the following 3xN array additions:

M_Red	=	**M_RedGray**	**+ M_RedError**	**+ M_RedLeak**
M_Green	=	**M_GreenGray**	**+ M_GreenError**	**+ M_GreenLeak**
M_Blue	=	**M_BlueGray**	**+ M_BlueError**	**+ M_BlueLeak**
M_SumPrim	=	**M_Gray**	**+ M_AllZeros**	**+ M_SumLeak**

The following step is only done if the rendering algorithm Alg_BNE is used:

- We normalize the **M_Gray** algorithm and obtain the 3xN **M_GrayNorm** array. For index i ranging from 0 to N-1, its column vectors are defined by:

$$\mathbf{M\_GrayNorm}^{\langle i \rangle} = \frac{1}{\mathbf{M\_Gray}_{0,i} + \mathbf{M\_Gray}_{1,i} + \mathbf{M\_Gray}_{2,i}} \mathbf{M\_Gray}^{\langle i \rangle}$$

2.2.2 Creating the correction tables.

This section is not required for the Alg_B rendering algorithm because it does not use correction tables. These tables are created using the linear interpolation function:

yt = LinInter(Scale, **x, y)**

where Scale is a large integer constant, we usually take it equal to 4095. Vectors **x** and **y** have N elements with floating point values. The values of the elements of **x** range between 0 and 1. Vector **y** contains the corresponding values we want to interpolate.The resulting vector **yt** has Scale + 1 elements. This function first multiplies each element of **x** by Scale. Then, for each integer j, between 0 and Scale, it determines the index value i, between 0 and N-1, such that j is contained in the interval $[x_{i-1}, x_i]$. It computes yt_j using the familiar linear interpolation formula:

$$yt_j = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \cdot \left(j - x_{i-1}\right)$$

- To interpolate the 3xN non-linear error array **M_RedError** we apply **LinInter()** to each one of its three rows separately, using the same **x** vector $\mathbf{M\_Red}^{T<0>}$, which represents the first row of **M_Red**. We obtain the 3x(Scale+1) array **M_InterRedError,** such that for index i, ranging from 0 to 2, its rows are defined by:

  $\mathbf{M\_InterRedError}^{T<i>} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Red}^{T<0>}, \mathbf{M\_RedError}^{T<i>})$

  Likewise, we interpolate the 3xN arrays **M_GreenError** and **M_BlueError** using $\mathbf{M\_Green}^{T<1>}$ and $\mathbf{M\_Blue}^{T<2>}$ respectively as the **x** vectors, and obtain the 3x(Scale+1) arrays **M_InterGreenError** and **M_InterBlueError**:

  $\mathbf{M\_InterGreenError}^{T<i>} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Green}^{T<1>}, \mathbf{M\_GreenError}^{T<i>})$

  $\mathbf{M\_InterBlueError}^{T<i>} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Blue}^{T<2>}, \mathbf{M\_BlueError}^{T<i>})$

The following correction tables are only used by the Alg_BNE rendering algorithm.

- We use the same method to linearly interpolate the 3xN arrays **M_RedLeak, M_GreenLeak, M_BlueLeak** into the 3x(Scale+1) arrays **M_InterRedLeak, M_InterGreenLeak, M_InterBlueLeak.** Less computations are required here because many of the rows are equal to zero. Once again, for index i ranging from 0 to 2:

  $\mathbf{M\_InterRedLeak}^{T<i>} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Red}^{T<0>}, \mathbf{M\_RedLeak}^{T<i>})$

  $\mathbf{M\_InterGreenLeak}^{T<i>} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Green}^{T<1>}, \mathbf{M\_GreenLeak}^{T<i>})$

  $\mathbf{M\_InterBlueLeak}^{T<i>} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Blue}^{T<2>}, \mathbf{M\_BlueLeak}^{T<i>})$

- The rows of the 3xN array **M_GrayNorm** are separately interpolated into three vectors with Scale + 1 elements, giving **M_InterGrayRedNorm, M_InterGrayGreenNorm, M_InterGrayBlueNorm** where:

  $\mathbf{M\_InterGrayRedNorm} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Red}^{T<0>}, \mathbf{M\_GrayNorm}^{T<0>})$

  $\mathbf{M\_InterGrayGreenNorm} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Green}^{T<1>}, \mathbf{M\_GrayNorm}^{T<1>})$

  $\mathbf{M\_InterGrayBlueNorm} = \mathbf{LinInter}(\text{Scale}, \mathbf{M\_Blue}^{T<2>}, \mathbf{M\_GrayNorm}^{T<2>})$

2.2.3 Creating the gamma tables.

The gamma tables are created using the **LogLogLinInter()** function:

yt = LogLogLinInter(Scale, **x, y)**

which has the same input parameters Scale and **x** as **LinInter().** Vectors **y** and **yt** have integer value elements and in our case they range from 0 to 255. N and Scale+1 are the respective number of elements for **y** and **yt.** This function first multiplies each element of **x** by Scale. Then it computes the logarithmic values of **x** and **y** and stores the results in the N elements vectors **lx** and **ly** respectively. Two Scale+1 elements vectors **lxt, lyt** are defined and their first elements $\mathbf{lxt_0}$, $\mathbf{lyt_0}$ are set to zero. For index j, ranging from 1 to Scale, **LogLogLinInter()** determines index i, ranging from 0 to N-1, such that $\mathbf{lxt}_j$ is contained in the interval $[lx_{i-1}, lx_i]$ then the linear function is applied giving:

$$lyt_j = ly_{i-1} + \frac{ly_i - ly_{i-1}}{lx_i - lx_{i-1}} \cdot \left(lxt_j - lx_{i-1}\right)$$

Finally yt_j is obtained by taking the inverse logarithm of lyt_j and rounding the result to an integer value.

For the Alg_B and Alg_BN rendering algorithms we use the following gamma tables:

- First we scale the 3xN array **M_Gray** such that the elements of its highest column are equal to one. This is done by defining the following matrix:

$$\mathbf{GrayScalingMatrix} = \begin{bmatrix} \left(\mathbf{M\_Gray}_{0,N-1}\right)^{-1} & 0 & 0 \\ 0 & \left(\mathbf{M\_Gray}_{1,N-1}\right)^{-1} & 0 \\ 0 & 0 & \left(\mathbf{M\_Gray}_{2,N-1}\right)^{-1} \end{bmatrix}$$

Then we compute the 3xN array:

$\mathbf{M\_ScaledGray} = \mathbf{GrayScalingMatrix} \cdot \mathbf{M\_Gray}$

To create the three Scale+1 elements vectors: **RedGammaSimple, GreenGamaSimple** and **BlueGammaSimple,** we respectively use for the x vectors of **LogLogLinInter()** the first, second and third rows of **M_ScaledGray** and for the y vectors the first row of display RGB array **D_Red**, the second row of **D_Green** and the third row of **D_Blue**:

$\mathbf{RedGammaSimple} = \mathbf{LogLogLinInter}(\text{Scale}, \mathbf{M\_ScaledGray}^{T<0>}, \mathbf{D\_Red}^{T<0>})$

$\mathbf{GreenGammaSimple} = \mathbf{LogLogLinInter}(\text{Scale}, \mathbf{M\_ScaledGray}^{T<1>}, \mathbf{D\_Green}^{T<1>})$

$\mathbf{BlueGammaSimple} = \mathbf{LogLogLinInter}(\text{Scale}, \mathbf{M\_ScaledGray}^{T<2>}, \mathbf{D\_Blue}^{T<2>})$

The following gamma tables are only used with the Alg_BNE rendering algorithm:

- In this case we don't scale **M_Gray** and obtain the three Scale+1 vectors **RedGammaFull, GreenGammaFull, BlueGammaFull** as follows:

$\mathbf{RedGammaFull} = \mathbf{LogLogLinInter}(\text{Scale}, \mathbf{M\_Gray}^{T<0>}, \mathbf{D\_Red}^{T<0>})$

$\mathbf{GreenGammaFull} = \mathbf{LogLogLinInter}(\text{Scale}, \mathbf{M\_Gray}^{T<1>}, \mathbf{D\_Green}^{T<1>})$

$\mathbf{BlueGammaFull} = \mathbf{LogLogLinInter}(\text{Scale}, \mathbf{M\_Gray}^{T<2>}, \mathbf{D\_Blue}^{T<2>})$

2.3 Rendering

This section describes the three rendering algorithms. Given a CIE XYZ vector **C_pixel** we want to find its corresponding display RGB vector **D_pixel**.

2.3.1 The remove black only algorithm (Alg-B).

This is done as follows:

1) Remove average black, (1), from **C_pixel** giving the vector **C_pixelk**: $\mathbf{C\_pixelk} = \mathbf{C\_pixel} - \mathbf{C\_k}$
2) Multiply 3x3 matrix $\mathbf{Md}^{-1}$, (2) by **C_pixelk** giving the vector **M_pixel**: $\mathbf{M\_pixel} = \mathbf{Md}^{-1} * \mathbf{C\_pixelk}$
3) Multiply **M_pixel** by Scale (4095 in our case), then round its three elements, which we name r, g, b, to integer values ranging from 0 to Scale.
4) Compute the display RGB vector **D_pixel** using the 1x(Scale+1) vectors **RedGammaSimple, GreenGammaSimple, BlueGammaSimple** and the three elements r, g, b, of the vector **M_pixel:**

$$\mathbf{D\_pixel} = \begin{pmatrix} \mathbf{RedGammaSimple}_{r} \\ \mathbf{GreenGammaSimple}_{g} \\ \mathbf{BlueGammaSimple}_{b} \end{pmatrix}$$

2.3.2 The remove black and non-linear errors algorithm (Alg_BN).

The first three steps are identical to those of Alg_B, we will only describe the following steps:

4) Use the elements r, g, b, of the vector **M_pixel** to select the column vectors of the 3x(Scale+1) arrays: **M_InterRedError, M_InterGreenError, M_InterBlueError.** Their sum gives the error vector **M_pixelerr.**

$$\mathbf{M\_pixelerr} = \mathbf{M\_InterRedError}^{<r>} + \mathbf{M\_InterGreenError}^{<g>} + \mathbf{M\_InterBlueError}^{<b>}$$

5) Correct the vector **M_pixel** and obtain the vector **M_pixelc**:

$$\mathbf{M\_pixelc} = \mathbf{M\_pixel} - \mathbf{M\_pixelerr}$$

6) As in step 3, multiply **M_pixelc** by Scale and round its three elements, which we name rc, gc, bc, to integer values ranging fom 0 to Scale.

7) As in step 4 of Alg_B, use the elements rc, gc, bc of vector **M_pixelc** to compute the display RGB vector **D_pixel**:

$$\mathbf{D\_pixel} = \begin{pmatrix} \mathbf{RedGammaSimple}_{rc} \\ \mathbf{GreenGammaSimple}_{gc} \\ \mathbf{BlueGammaSimple}_{bc} \end{pmatrix}$$

2.3.3 The remove black, non-linear and leakage errors algorithm (Alg_BNL)

The first four steps are identical to those of Alg_BN.

5) To estimate the error due to leakage, first multiply vector **M_pixel** by the inverse sum of its elements, r, g, b, giving vector **M_pixelnorm**:

$$\mathbf{M\_pixelnorm} = (r + g + b)^{-1} * \mathbf{M\_pixel}$$

6) Compute the three scalar valued coefficients RedCoeff, GreenCoeff, BlueCoeff using the following formulas:

$$RedCoeff = (\mathbf{M\_pixelnorm}_0 - \mathbf{M\_InterGrayRedNorm}_r)^{-1} * (1 - \mathbf{M\_InterGrayRedNorm}_r)$$

$$GreenCoeff = (\mathbf{M\_pixelnorm}_1 - \mathbf{M\_InterGrayRedNorm}_g)^{-1} * (1 - \mathbf{M\_InterGrayRedNorm}_g)$$

$$BlueCoeff = (\mathbf{M\_pixelnorm}_2 - \mathbf{M\_InterGrayRedNorm}_b)^{-1} * (1 - \mathbf{M\_InterGrayRedNorm}_b)$$

If any one value of RedCoeff , GreenCoeff, BlueCoeff is negative then replace it by zero.
The RedCoeff formula is chosen such that if **M_pixel** is equal to one of the column vectors of the M_Red array described in (3) then RedCoeff is equal to 1 while GreenCoeff and BlueCoeff are equal to zero. Likewise, if **M_pixel** is equal to one of the column vectors of **M_Gray** then all three coefficients will be equal to zero.

7) Use the following formulas, to compute the three elements of the leakage vector **M_pixelleak**:

$$\mathbf{M\_pixelleak}_0 = RedCoeff * \mathbf{M\_InterRedLeak}^{<r>}$$

$$\mathbf{M\_pixelleak}_1 = GreenCoeff * \mathbf{M\_InterGreenLeak}^{<g>}$$

$$\mathbf{M\_pixelleak}_2 = BlueCoeff * \mathbf{M\_InterBlueLeak}^{<b>}$$

8) Correct the vector **M_pixel** giving the vector **M_pixelc**:

$$\mathbf{M\_pixelc} = \mathbf{M\_pixel} - \mathbf{M\_pixelerr} - \mathbf{M\_pixelleak}$$

9) As in step 3, multiply **M_pixelc** by Scale and round its three elements, which we name rc, gc, bc, to integer values ranging from 0 to Scale.

10) Use the elements rc, gc, bc of vector **M_pixelc** and the 1x(Scale+1) vectors: **RedGammaFull, GreenGammaFull, BlueGammaFull**, to compute the display RGB vector **D_pixel**:

$$\mathbf{D\_pixel} = \begin{pmatrix} \mathbf{RedGammaFull}_{rc} \\ \mathbf{GreenGammaFull}_{gc} \\ \mathbf{BlueGammaFull}_{bc} \end{pmatrix}$$

3. A CASE STUDY

The implementation method described in the previous section was tested with the following four desktop IBM LCD models which were marketed between the years 1996 and 2000:

Model	Diagonal (inches)	Resolution (pixels)	Luminance (nits) Black	White	Type of LCD	Year first marketed
9516A03	16	1280x1024	0.7	108	TFT, TN	1996
T85A	18.1	1280x1024	0.6	127	TFT,TN	1998
T55A	15	1024x768	0.6	138	TFT,TN	1999
T86A	18.1	1280x1024	0.6	105	TFT, IPS single domain	2000

(TFT = Thin film transistor, IPS = In plane switching)

Our objective was to compare the three rendering algorithms using these four models. A set of 216 display RGB vectors was created which contained the permutations of index levels: 0, 50, 100, 150, 200, 255 for the three coordinates. The measurements were performed with the same spectroradiometer and computer. The following processing was done for each one of the samples:

1 . It was displayed as a patch at the center of the screen and we measured its CIE XYZ vector.
2 . The result was used as input for the three rendering algorithms and we obtained from each one a display RGB vector.
3 . We consecutively displayed a patch representing the color of each one of these vectors and measured their CIE XYZ values.
4 . We compared each one of these measurements to the CIE XYZ vector of step one and computed their respective Delta E values.

We obtained the following average Delta E for each LCD model and display algorithm:

Model	Alg_B	Alg_BN	Alg_BNL
9516A03	3.364	2.777	3.235
T85A	0.804	0.735	0.661
T55A	1.788	1.434	1.908
T86A	1.112	1.124	1.050

These results suggest the following observations:

1 . The color generation of LCD displays has shown in the last four years improved conformance to CRT color production models; the moderately priced T55A of 1999 gave better results than the expensive 9516A03 of 1996.
2 . Alg_BN gave better results than Alg_B for three out of four models. The assumption made in our implementation that given a random display RGB vector it is possible to predict the errors introduced by the variable chromaticities of the primaries from their calibration data appears to be statistically correct.
3 . Alg_BNL gave worse results than Alg_BN in two out of four cases. Even, in the two cases which produced slight improvements, the amount of extra computations made it less attractive.
4 . Alg_B, the simplest of the three rendering algorithms, gave good results for the newer displays. If this trend continues, it should become the method of choice.

The Appendix illustrates the color characteristics of these four models and how they can be improved when black is subtracted from the measurement data.

4.CONCLUSION

In this paper, we compared the color characteristics of a CRT to those of an LCD. We observed that the calibration method usually applied to the former must be modified before it can be used with the latter. We described an implementation which performed three types of corrections to the calibration data of an LCD:

1 . It subtracted the CIE XYZ value of black from all the measurements.
2 . It computed non-linear error tables which were used to compensate for the variable chromaticities of the primaries.
3 . It created leakage tables which improved the color additivity problem.

Next we described three image rendering algorithms for a display calibrated with such a method:

1 . Alg_B was the least complex and took only into account the black subtraction.
2 . Alg_BN also used the non-linear error tables.
3 . Alg_BNL required the most computations because it furthermore employed the leakage tables.

A case study of this method was done using four IBM LCD models produced in the last four years. We noticed the color

improvements which took place during this period. We concluded that Alg_BNL was inefficient for most models, that Alg_BN could be generally used to improve a calibration and that Alg_B was sufficient for the newest product.

ACKNOWLEDGMENTS

We would like to thank Paul Borrel, our department manager, for supporting this study. Richard Austen and Eric Nelson of Gamma Scientific for lending us a GS 1280 spectroradiometer. The LCD group at IBM Research, which includes Evan Colgan, Alan Lien, Kai Schleupen, Robert L. Wisnieff and Steve L. Wright, for sharing their expertise with us. Our colleagues Louis Herzberg for editorial assistance and Oconel Johnson Jr. for keeping our machines running.

REFERENCES

1. R. S. Berns, M. E. Gorzynski, R. J. Motta, "CRT colorimetry part 1: theory and practice". *Color Research and Application*, **Vol. 18, No. 5,** pp. 299-314, Oct. 93.
2. R. S. Berns, M. E. Gorzynski, R. J. Motta, "CRT colorimetry part 2: metrology", *Color Research and Application*, **Vol. 18, No. 5,** pp. 315-325, Oct. 93.
3. A. Cazes, G. Braudaway, J. Christensen, M. Cordes, D. DeCain, A. Lien, F. Mintzer, S. L. Wright, "On the color calibration of liquid crystal displays", *Proceedings of SPIE Flat Panel Display Technology and Display Metrology*, **Vol. 3636,** pp. 154-161, Jan. 99.
4. H. R. Kang, *Color Technology for Electronic Imaging Devices,* SPIE Optical Engineering Press, 1997.
5. International Electrotechnical Commission, "Colour measurement and management in multimedia systems and equipment. Part 3 equipment using cathode ray tubes", *IEC*, 1998.
6. International Electrotechnical Commission, "Colour measurement and management in multimedia systems and equipment. Part 4 equipment using liquid crystal display panels", *IEC*, 1998.
7. F. C. Mintzer, G. Goertzel, G. R. Thompson, "Display of images with calibrated color on a system featuring monitors with limited color palettes", *SID International Symposium Boston, Mass.* **Vol. 23,** pp.377-380, May 17-22 1992.
8. D. L. Post, C. S. Calhoun, "An evaluation of methods for producing desired colors on CRT monitors", *Color Research and Application*, **Vol. 14, No. 4,** pp. 173-186, Aug. 89.
9. Video Electronics Standards Association, *Flat Panel Display Measurements Standard Version 1*, VESA, May 1998.

APPENDIX

In this appendix we illustrate the effects of subtracting black from the calibration data on the luminance curves, color additivity of the grays and chromaticities of the four LCD models used in our case study.

APP.1 Luminance Curves.

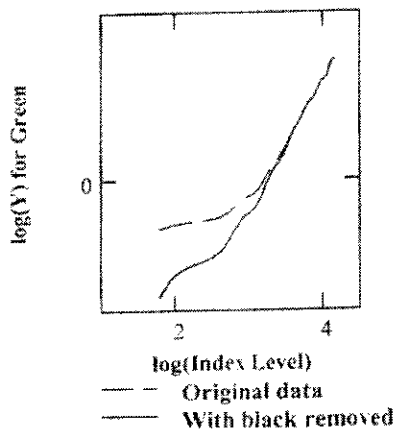

Figure 1A. 9516A03

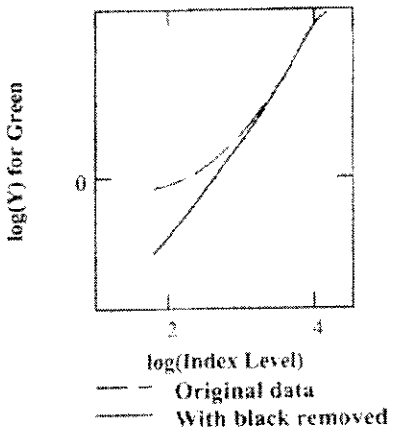

Figure 1B. T85A

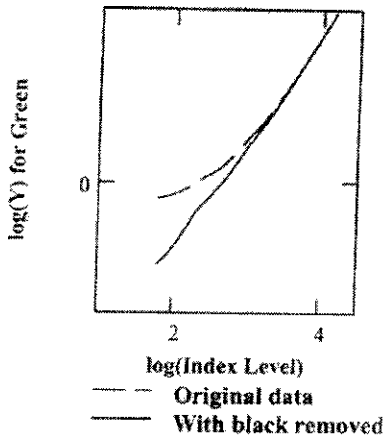

Figure 1C. T55A

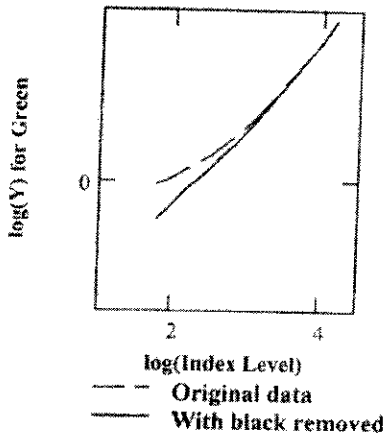

Figure 1D. T86A

Luminance curves for the primaries of a display are supposed to have an exponential form, such that if they are represented on a log-log graph the result should be a straight line. The slope of this line is usually called the gamma of the display. Figures 1A through 1D illustrate these curves for the green primaries of the four models. We notice that removing black from the data improves the straightness of the curve. This is more evident with the models of the last three years.

APP.2 Color Additivity of the grays.

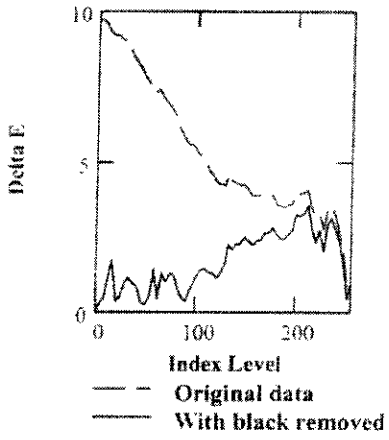

Figure 2A. 9516A03

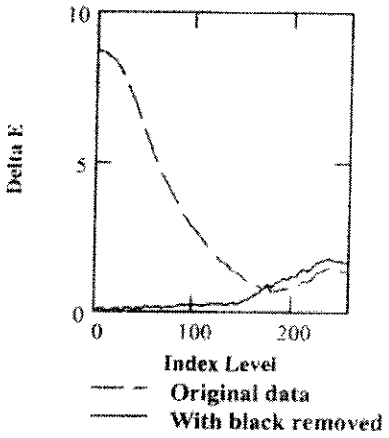

Figure 2B. T85A

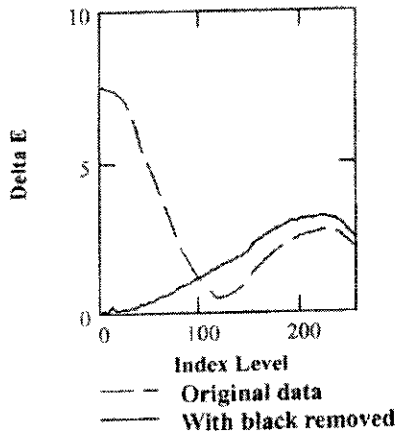

Figure 2C. T55A

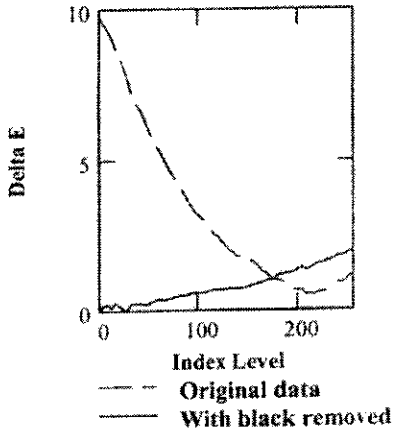

Figure 2D. T86A

For each index level, we sum the CIE XYZ data of the primaries then compare it to the corresponding gray and compute their Delta E. The same operation is repeated after we subtract black from the primaries and grays. Figures 2A through 2D illustrate the results for the four models.

APP.3 Chromaticities

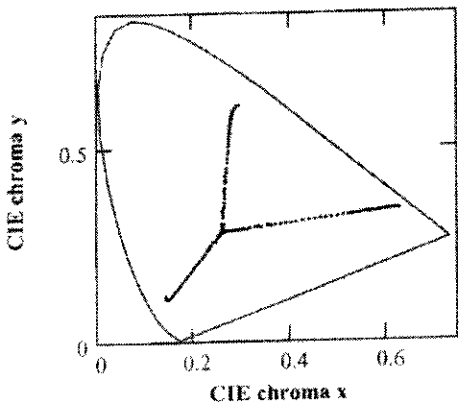

Figure 3Aa. 9516A03 (Original data)

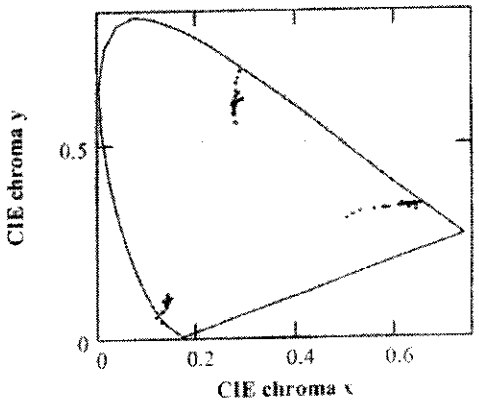

Figure 3Ab. 9516A03 (Black removed)

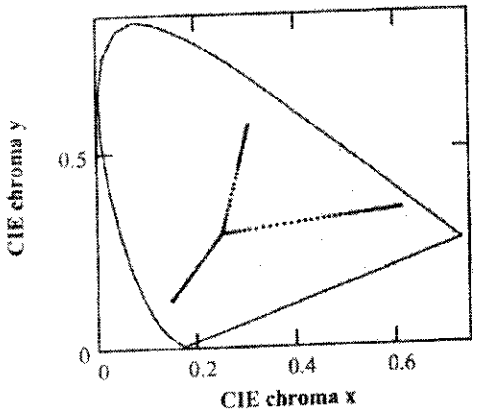

Figure 3Ba. T85A (Original data)

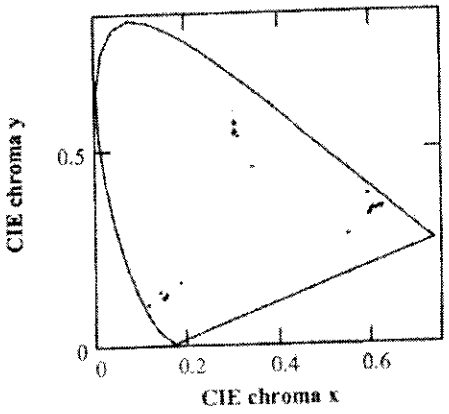

Figure 3Bb. T85A (Black removed)

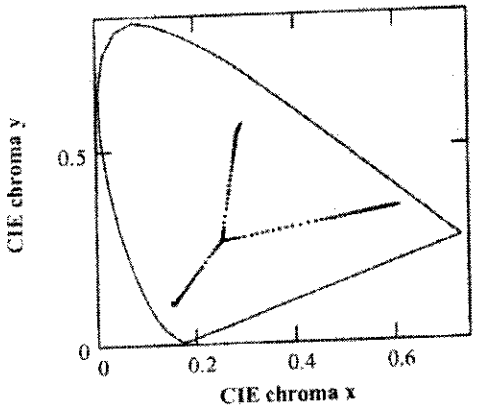

Figure 3Ca. T55A (Original data)

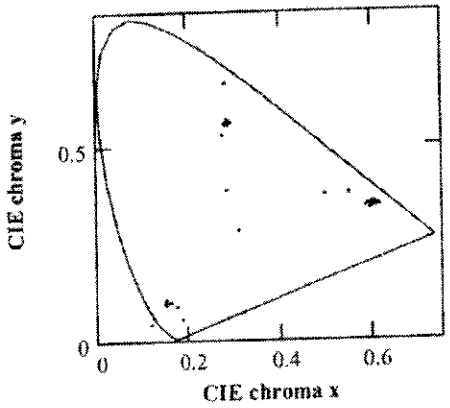

Figure 3Cb. T55A (Black removed)

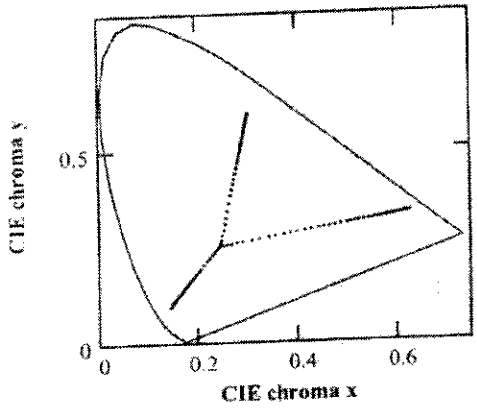

Figure 3Da. T86A(Original data)

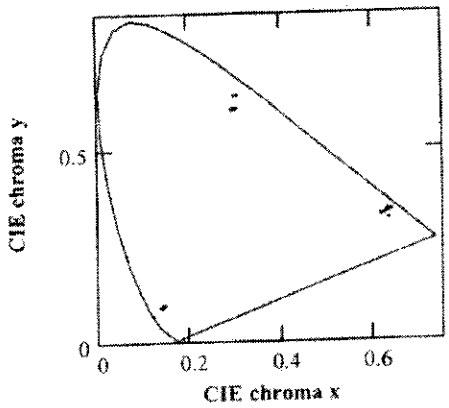

Figure 3Db. T86A (Black removed)

Figures 3Aa through 3Db illustrate the chromaticities of the primaries at 65 luminance levels. For the ideal display, all intensity levels of a primary should be represented by a single point. However, this is not the case for an LCD as shown in Figures 3Aa, 3Ba, 3Ca and 3Da. Subtracting black from the measurements of a primary improves the situation as illustrated in the corresponding Figures 3Ab, 3Bb, 3Cb and 3Db. The most recent model, T86A, shows the best results.