A VLSI Implementation of Color Gamut Mapping Method for Real-Time Display Quality Enhancement

Dongil Han
Department of Computer Engineering,
Sejong University, Seoul, 143-747, Korea
Tel : +82-2-3408-3751 Fax : +82-2-3408-3321
E-mail: dihan@sejong.ac.kr

Abstract: The color gamut mapping method that is used for enhancing the color reproduction quality between PC monitor and printer devices is adopted for display quality enhancement. The high definition display devices operate at the clock speed of around 70 MHz ~ 150 MHz and permit several nano seconds for real-time processing. Thus, the concept of three-dimensional reduced resolution look-up table is used. The required hardware can be greatly reduced by look-up table resolution adjustment. The proposed hardware architecture is successfully implemented in ASIC and also successfully adopted in display quality enhancement purposes.

Keywords: Digital Image Processing, VLSI implementation, Color Gamut Mapping, Display Quality Enhancement

1. INTRODUCTION

The conventional CRT type display equipment which was mainly used in analog TV is rapidly replaced by large size, high resolution display devices. However, most of the newly developed display devices offer lower display quality compared to the conversional CRT displays. For example, the CRT has better display quality than the newly developed display devices in most features such as image quality, viewing angle, contrast ratio, brightness, durability, response time, and many other aspects. Thus, several methods have been proposed to improve the display quality of various display devices[1].

And each display panel has its own display characteristics and may require special color adjustments for each display device. For example, a gamma curve for CRT adjustment is not suitable for PDP or LCD TV.

The conventional approach for adjusting display characteristics uses three one-dimensional look-up tables for Red, Blue and Green component adjustment. This one dimensional look-up table, however, cannot correct display gamut exactly. There are many color gamut mapping algorithms which can be used to reduce any differences that might exist among the sets of colors obtainable on different media or display devices [2] – [7].

A spatial gamut mapping technique [2] is used to overcome the shortcomings encountered with standard point-wise gamut mapping algorithms by preserving spatially local luminance variations in the original image. The difference between the original image luminance and gamut mapped image luminance is calculated. A spatial filter is then applied to this difference signal, whose output is added back to the gamut-mapped signal. Finally, all pixels are processed through a color correction function for the output device, and rendered for that device. The algorithm is designed to reduce many of the artifacts arising from standard point-wise techniques and psychophysical experiments give good results.

The lightness mapping and multiple anchor points are proposed by Lee [3]. The lightness mapping minimizes the lightness difference of the maximum chroma between two gamuts and produces the linear tone in bright and dark regions. In the chroma mapping, a separate mapping method, which utilizes multiple anchor points with constant slopes plus a fixed anchor point, is used to maintain the maximum chroma and to produce a uniform tonal dynamic range.

Three-dimensional gamma-compression gamut mapping algorithm (GMA) is proposed by Chen [4]. This article describes the 3D GMA based on the concept of Image-to-Device (I-D). Considering color gamut relationships between source image and printer device, the Gamma-Compression GMA is applied to the 3-D shell shapes in CIE L*a*b* space. It is shown that the GMA coupled with 3D gamut compression and multi-mapping directions resulted in better rendition than 2D nonlinear GMA.

As discussed above, lots of outstanding color mapping algorithms for reducing the gamut differences have been proposed. However, all of these mapping algorithms are very difficult to use in the real-time display quality enhancement applications. About 10-nano second processing speed is required to adopt the gamut mapping algorithms in real-time display quality enhancement applications.

In this paper, we propose a general-purpose gamut mapping method and a hardware architecture.

2. GAMUT MAPPING ARCHITECTURE

The popular color space which is used in color management system, device independent color model, and gamut mapping is CIE L*a*b* color space.

Though, this color space is not a directly displayable format, its gamut encompasses the entire visible spectrum and can represent accurately the colors of any display, print, or input device.

But, most display devices generate color information by using combinations of the red, green and blue primary colors. In Rec. ITU-R BT. 709 [8], the RGB system primaries are defined using the CIE 1931 system of colorimetry in x, y coordinates. However, the improper selections of color phosphor, color filter or color signal gain generate different set of display RGB primaries.

Thus, the conventional gamut mapping methods use sequential color space conversion and individual color gamut mapping functions.

For real time hardware implementation, we can consider several approaches. Firstly, two color space conversion functions and the gamut mapping functions be implemented in hardware, respectively. The color space conversion function between RGB, YCbCr, YIQ, YUV color space requires 3x3 matrix operation and can be implemented easily in hardware. But RGB to L*a*b* conversion requires a highly nonlinear conversion function and is difficult to be implemented in hardware.

The relation between RGB signals and CIE XYZ tristimulus values is described in SMPTE RP177-1993 [9] and it can be calculated by using 3x3 matrix multiplication as follows.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

But, XYZ to L*a*b* conversion requires complex and highly nonlinear equations as follows.

$$L^* = 116 f\left(\frac{Y}{Y_n}\right) - 16 \quad (2)$$

$$a^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right] \quad (3)$$

$$b^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right]. \quad (4)$$

$$\text{Where, } f(q) = q^{1/3} \quad for \quad q > 0.008856 \quad (5)$$

$$f(q) = 7.787q + \frac{16}{116} \quad for \quad q \le 0.008856 \quad (6)$$

and X_n, Y_n, and Z_n are the CIE 1931 tristimulus values of the reference white under the reference illumination. These values are typically the white of a perfectly reflecting diffuser under CIE standard D65 illumination defined by $x = 0.3127$ and $y = 0.3290$ in the CIE chromaticity diagram [10].

Most of the gamut-mapping algorithms also require very complex mapping functions and are very difficult to be implemented in hardware. In Figure 1, the 3-D gamuts are computed in L*a*b* space and visualized simultaneously with the color gamut obtained from the ITU-R BT. 709 HDTV standard parameters [8] shown as a wire mesh and a sample PDP gamut as a solid.

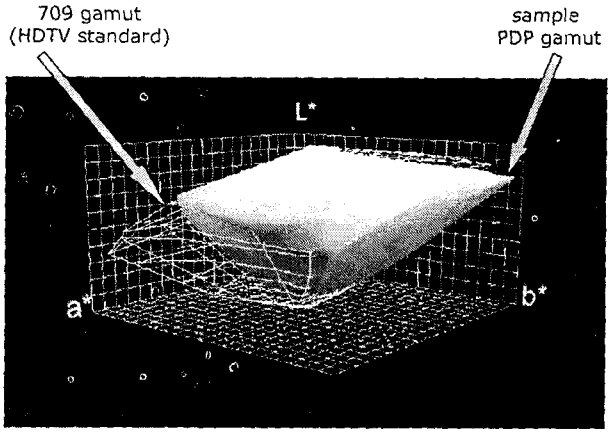

Fig. 1. An example of color gamut difference

This figure shows that there are significant differences between two display gamuts. To correct these mismatches, highly nonlinear color gamut mapping is required. And this non-linearity makes it difficult to implement in hardware. Furthermore, the developed color gamut-mapping algorithm could be changed and upgraded later in order to fix possible problems and defects. In this case, the hard-wired solution could not adapt the necessary modifications.

The second approach for real-time hardware implementation, we can consider a 3-dimensional look-up table which can convert two color space conversion rules and gamut mapping rules into one mapping rule and can define all mapping functions for each red, green and blue input and output. Thus, in setup stage, the mapping function is loaded into 3-dimensional look-up table. And, in normal stage, it can generate transformed R_{out}, G_{out} and B_{out} data for each R_{in}, G_{in} and B_{in} image input. In this case, the gamut mapping itself can be loaded in software manner and renders the architecture for adapting the modification and upgrade of gamut-mapping rules. But a major disadvantage of this approach is that the required memory space could be calculated as 256 x 256 x 256 x 3 bytes = 50,331,648 bytes. Such a large amount of memory cannot be implemented in real-time hardware or ASIC.

Thus in [11], the concept of three-dimensional reduced resolution look-up table (RRLT) is introduced. Figure 2 shows the block diagram of the RRLT. Each R, G, B component is divided into 8 intervals. The three most significant bits of each component are used to select

the corresponding cube position and to generate the eight mapping values of each vertex of the cube. Five least significant bits of each component are used for interpolating the new mapping value for each R, G and B input image.

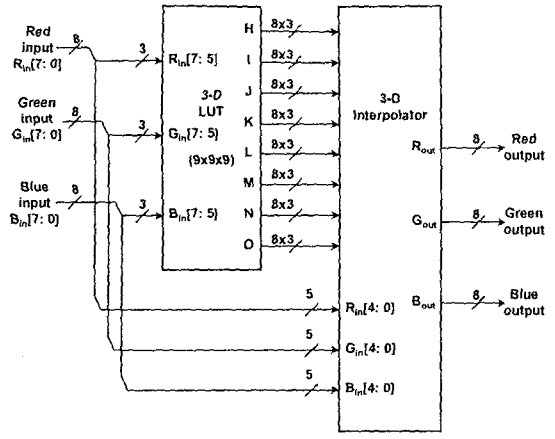

Fig. 2. The 3-dimensional reduced resolution look-up table architecture

In this example, the three most significant bits of each component are used for direct mapping. The final gamut-mapped R_{out}, G_{out}, B_{out} values are calculated by 3-dimensional interpolator. In this case, using the 3-bit MSB information for each R_{in}, G_{in}, and B_{in} triplet, the 3-D LUT generates gamut-mapping values of eight vertex points that encompass the R_{in}, G_{in}, and B_{in} input triplet.

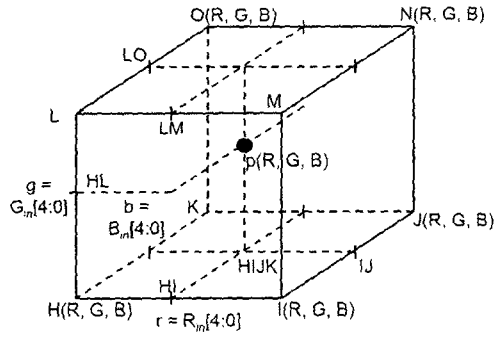

Fig. 3. Three-dimensional interpolation method

The interpolation scheme is shown in figure 3. The eight vertex data and the five least significant bits of each component are used to determine the internal cube position and the final mapping value. A simple consecutive bi-linear interpolation for each component can generate a new mapping value at the image input position 'p'. The direct mapping value H(R, G, B), and I(R, G, B) from the 3-D LUT and $R_{in}[4:0]$ value are used to calculate HI(R, G, B) value which denotes the new gamut mapping value in HI position. The KJ(R, G, B) value can be calculated in the same manner. Thus, if we let the gamut mapping function of each components be $gm_{component}(.)$, the final gamut mapping value of red signal at 'p' point can be calculated as follows.

$$R_{HI} = (R_H \times (32 - r) + R_I \times r) / 32 \quad (7)$$

$$R_{KJ} = (R_K \times (32 - r) + R_J \times r) / 32 \quad (8)$$

here

$$r = R_{in}[4:0] \quad (9)$$

$$R_H = gm_{red}(H(R,G,B)) \quad (10)$$

$$R_I = gm_{red}(I(R,G,B)). \quad (11)$$

In the second step,

$$R_{HIJK} = (R_{HI} \times (32 - b) + R_{KJ} \times b) / 32 \quad (12)$$

And finally we can get

$$R_p = (R_{HIJK} \times (32 - g) + R_{LMNO} \times g) / 32 \quad (13)$$

here

$$b = B_{in}[4:0] \quad (14)$$

$$g = G_{in}[4:0] \quad (15)$$

The green and blue mapping values are calculated as follows.

$$G_p = (G_{HIJK} \times (32 - g) + G_{LMNO} \times g) / 32 \quad (16)$$

$$B_p = (B_{HIJK} \times (32 - g) + B_{LMNO} \times g) / 32 \quad (17)$$

The simple integer operations are sufficient to interpolate and generate the final mapping values.

3. HARDWARE IMPLEMENTATION

The proposed hardware architecture, described in the previous section, has some difficult problems to be implemented in real-time hardware. The red, green, and blue mapping values of 8 vertex points are required simultaneously to calculate the new mapping value at p point. In every pixel clock, the mapping data at eight vertex points are used to calculate the new mapping value.

Even though, the size of 3-D LUT is greatly reduced, it is very difficult to generate mapping data of 8-vertex point for each pixel by using a simple 9x9x9 three-dimensional look-up table. The pipelined architecture could be considered for generating the 8-vertex point sequentially. But in this case, the mapping value of current pixel data and that of next pixel data should be generated simultaneously and it is also very difficult to implement them. Thus, the pipelined architecture could not solve this problem.

3.1. Real-Time Architecture

In this paper, as shown in figure 4, the 3-D real-time hardware architecture is implemented by using an address decoder, eight one-dimensional look-up tables and a data switch. The 9x9x9 = 729 vertex points are separated and stored into eight one-dimensional look-up table. For example, the mapping values of R_{in} [7:5] = 0, G_{in} [7:5] = 0, B_{in} [7:5] = 0 position are stored into first address of look-up table 0(LUT0).

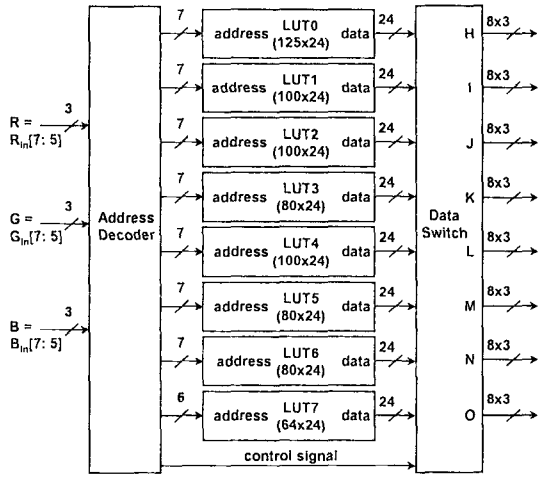

Fig. 4. Real time one-dimensional implementation of 9x9x9 3-D look-up table

And the mapping values of R_{in} [7:5] = 1, G_{in} [7:5] = 0, B_{in} [7:5] = 0 position are stored into first address of look-up table 1(LUT1). The eight vertex points that include origin are stored into first address of eight one-dimensional look-up tables. In this manner, the 729 vertex points can be stored into 8 one-dimensional look-up table.

This addressing logic makes it possible to use eight one-dimensional look-up tables instead of one three-dimensional look-up table. Figure 9 and figure 10 show the look-up table selection rules dependent on the cubic positions.

According to the image input, the same vertex can be selected in the different cube positions. For example, the first address of LUT1, as discussed in the previous paragraphs, can be selected when R_{in} = 10, G_{in} = 10, B_{in} = 10 input and R_{in} = 40, G_{in} = 10, B_{in} = 10 input. Thus, the precise selection of one-dimensional look-up table is required for storing the mapping values of each cube vertex positions.

Figure 5 shows the selected look-up tables when B_{in} [7:5] equals 0, 2, 4, 6, and special mapping value 8. In this figure, x-axis denotes R_{in} [7:5] position and y-axis denotes G_{in} [7:5] position. The look-up table 0 ~ 3 are used for storing the mapping values for each R_{in} [7:5] and G_{in} [7:5] values. Figure 6 shows the selected look-up tables when B_{in} [7:5] equals 1, 3, 5, and 7. In this case, look-up table 4 ~ 7 are used. Thus, the 3-dimensional look-up table can be constructed by piling up the structure as shown in figure 5 and 6, repeatedly. This architecture generates 8 one-dimensional look-up tables with different look-up table sizes. For example, as shown in figure 5, 25 position data is stored in LUT0. And several B_{in} value such as B_{in} [7:5] equals 0, 2, 4, 6, and special mapping value 8 also have same look-up table selection rules. Therefore, total 5x5x5 = 125 position data is stored in LUT0. In the case of look-up table 7(LUT7), 16 position data is stored for each B_{in} values when B_{in} [7:5] equals 1, 3, 5, and 7. Thus, totally 4x4x4 = 64 position data is stored in LUT7.

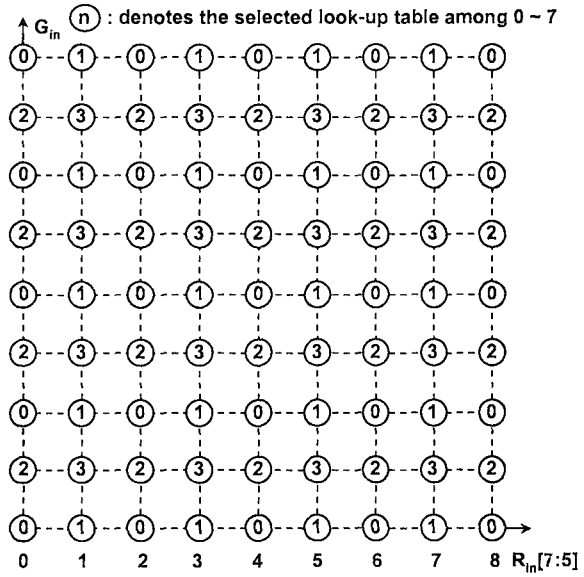

Fig. 5. The selected one dimensional look-up table when B_{in}[7:5] = 0, 2, 4, 6, and special value 8

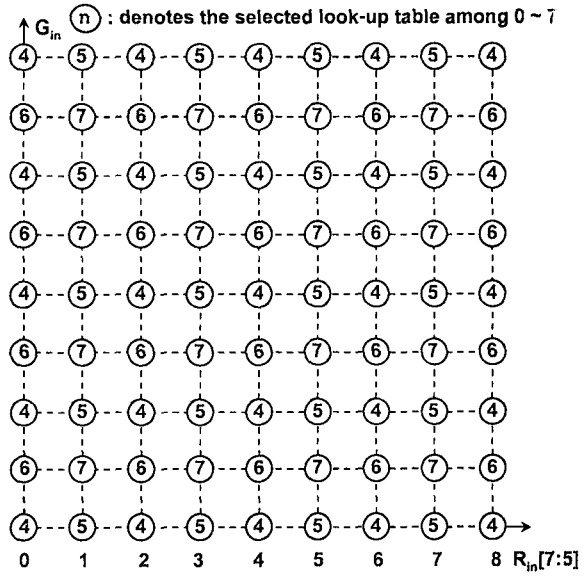

Fig. 6. The selected one dimensional look-up table when B_{in}[7:5] = 1, 3, 5, and 7

3.2. Initial Gamut Mapping Data Setup

In the setup stage, arbitrary gamut mapping rules can be loaded into the 3-D RRLT. The address decoder selects the proper look-up table and its address for all R, G, B input values. The address decoding logic can be described as follows.

$$LUT0\_address = (R+1)/2 + 5*((G+1)/2) + 25*((B+1)/2) \quad (18)$$

$$LUT1\_address = (R\ )/2 + 4*((G+1)/2) + 20*((B+1)/2) \quad (19)$$

$$LUT2\_address = (R+1)/2 + 5*((G\ )/2) + 20*((B+1)/2) \quad (20)$$

$$LUT3\_address = (R\ )/2 + 4*((G\ )/2) + 16*((B+1)/2) \quad (21)$$

$$LUT4\_address = (R+1)/2 + 5*((G+1)/2) + 25*((B\ )/2) \quad (22)$$

$$LUT5\_address = (R\ )/2 + 4*((G+1)/2) + 20*((B\ )/2) \quad (23)$$

$$LUT6\_address = (R+1)/2 + 5*((G\ )/2) + 20*((B\ )/2) \quad (24)$$

$$LUT7\_address = (R\ )/2 + 4*((G\ )/2) + 16*((B\ )/2) \quad (25)$$

For interpolation, extra mapping values when R_{in} = 256, G_{in} = 256, B_{in} = 256 must be loaded into the look-up table. These mapping values are used for interpolating the final mapping values when input image is located at the outer cube position of 3-D LUT. The simple integer operations are applied to the address generation logic.

3.3. Real-Time Color Gamut Mapping

For any input values, 8-vertex points that come from the vertex of cube that encompass the input triplet can be generated from the eight one-dimensional look-up tables. For example, let us consider a R_{in} = 10, G_{in} = 10, and B_{in} = 10 input. In this case, a cube that encompasses the origin is selected. For proper operation, LUT0 generates mapping values in H point, and LUT1, LUT2, LUT3 generate mapping values in I, J, K points, respectively. But, let's consider another example, the R_{in} = 80, G_{in} = 100, B_{in} = 80 input case. From the figure 9 and figure 10, we can find that the LUT0, LUT1, LUT2, and LUT3 generate mapping values in J, K, H, I point, respectively.

Thus, the final problem of this architecture is that the generated mapping data cube positions for any input can be changed arbitrary. As discussed in the previous paragraph, LUT0 can generate mapping data of H, and J position of cube and it is dependent on the input value. Each LUT can generate all positions of cube vertex. Thus the data switch block in figure 8 changes all mapping data positions to a fixed position which is suitable for interpolation. This reduces the complexity of the 3-D interpolator.

The address decoding logic for real-time gamut is the same as that of the initial gamut mapping data setup. The same equation as in (18) ~ (25) is used for real-time gamut mapping.

4. EXPERIMENTAL RESULTS

The proposed hardware architecture is firstly simulated in C. The 729 direct mapping data is stored first, and then other positions are calculated by interpolation. Simulation results show that even this method reduces the gamut mapping quality, this does not deteriorate the image quality. That is, the interpolation does not deteriorate the image quality.

The proposed hardware architecture is implemented in VHDL and the VHDL simulation shows the same results in C-simulation. Using the hardware emulation, many still and moving images are tested. Finally, the proposed architecture is implemented in FPGA and ASIC. The required memory and logic gate are shown in Table 1. Less than the 100,000 gates are required for implementing the real-time 3-D RRLT.

Table. 1 The used gate count for implementing 3-D RRLT

	Memory	Logic	Total
Gate count	47,303	45,658	92,961

The operation frequency of the 3-D RRLT is faster than 162MHz and can be applied for 1920x1080 progressive HDTV display purposes. Finally, the 3-D RRLT architecture is successfully included in the Video Display Processor for HDTV. The 0.18 μm CMOS technology is used for fabrication. Figure 7 shows the layout plot of the video display processor that includes the proposed 3-D RRLT for real-time color gamut mapping.

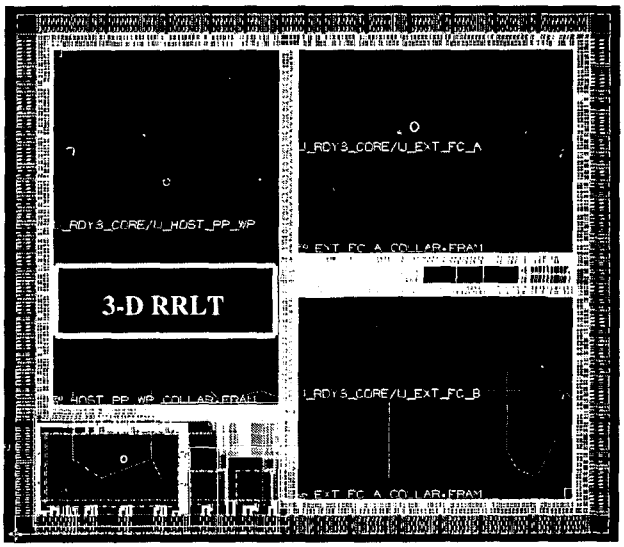

Figure 7. The layout plot of the video display processor for HDTV

5. CONCLUSION

The color gamut mapping method that is used to enhance the color reproduction quality between PC monitor ad printer devices is adopted for digital TV display quality enhancement. For high speed real-time processing and affordable system integration, the concept of three-dimensional reduced resolution look-up table is proposed and successfully implemented in hardware.

The proposed architecture is not dependent on the gamut mapping algorithm, and arbitrary gamut mapping rules can be accommodated by the proposed 3-dimensional reduced resolution look-up table. And it can be easily implemented in the real time at the reasonable hardware cost. The number of intervals for approximating the three-dimensional look-up table can be changeable and it can be selected by affordable memory size and required color mapping precision.

6. ACKNOWLEDGMENT

This work is supported by grant No R01-2003-000-10785-0 from the Basic Research Program of the Korea Science & Engineering Foundation, and by the IC Design Education Center.

References

[1] LeRoy DeMarsh, "Colorimetry for HDTV", *IEEE Trans. on Consumer Electronics*, vol. 37, no. 1, pp. 1-6, 1991.

[2] Raja Bala, Ricardo deQueiroz, Reiner Eschach and Wencheng Wu, "Gamut Mapping to Preserve Spatial Luminance Variations," *Journal of Image Science and Technology*, Vol. 45, no. 5, pp.436-443, September/October 2001.

[3] Chae-Soo Lee, Yang-Woo Park, Seok-Je Cho and Yeong-Ho Ha, "Gamut Mapping Algorithm Using Lightness Mapping and Multiple Anchor Points for Linear Tone and Maximum Chroma Reproduction," *Journal of Image Science and Technology*, Vol. 45, no. 3, pp.209-223, May/June 2001.

[4] Hung-Shing Chen and Hiroaki Kotera, "Three-dimensional Gamut Mapping Method Based on the Concept of Image Dependence," *Journal of Image Science and Technology*, Vol. 46, no. 1, pp44-52, January/February 2002

[5] B. Pham and G. Pringle, "Color Correction for an Image Sequence," *IEEE Computer Graphics and Applications*, pp.38-42, 1995.

[6] H. Haneishi, K. Miyata, H. Yaguchi and Y. Miyake, "A New Method for Color Correction in Hardcopy from CRT Images," *Journal of Image Science and Technology*, Vol. 37, no. 1, pp.30-36, 1993.

[7] Byoung-Ho Kang, Jan Morovic, M. Ronnier Luo, and Maeng-Sub Cho, "Gamut Compression and Extension Algorithms Based on Observer Experimental Data," *ETRI Journal*, Vol. 25, no. 3, pp.156-170, 2003.

[8] Rec. ITU-R BT 709, " Parameter Values for the HDTV Standards for Production and International Programme Exchange", 2000

[9] SMPTE Recommended Practice RP177-1993, "Derivation of Basic Television Color Equations", 1993

[10] CIE, "Uniform Color Spaces - Color Difference Equations Psychometric Color Terms", Commission Internationale de L'Eclairage, Publication No. 15, Supplement No. 2, Paris, 1978.

[11] Dongil Han, "A Novel Color Gamut Mapping Method Based on the 3-Dimensional Reduced Resolution Look-Up Table", International Technical Conference on Circuit/Systems, Computers and Communications, pp 507-510, July 2003.