To the Graduate Council:

I am submitting herewith a thesis written by Shaofan Xu entitled "Minimization of Color Errors in Television Receivers using Neural Networks." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Dr. Paul Benjamin Crilly, Major Professor

We have read this thesis
and recommend its acceptance:

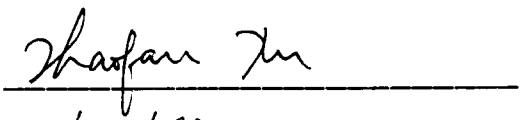
Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Requests for permission for extensive quotation from or reproduction of this thesis in whole or in parts may be granted by the copyright holder.

Signature 

Date 12/4/92

MINIMIZATION OF COLOR ERRORS IN TELEVISION RECEIVERS USING NEURAL NETWORKS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Shaofan Xu

May 1993

DEDICATION

This thesis is dedicated to my wife

Jie Zhu (朱洁)

who has given me consistent love, encouragement and support

with my goals in academia and life;

and also to my parents

Mr. Xu Fubi and Mrs. Liu Aiyin

(徐辅弼, 刘艾荫)

who have given me invaluable educational opportunities

and spiritual support.

ACKNOWLEDGMENTS

This research was conducted at the University of Tennessee, Knoxville and Philips Laboratories, North American Philips Corporation from February 1992 through November 1992. Research grants from Philips Laboratories made this project possible.

I would like to thank my major adviser, Dr. Paul B. Crilly, for his support, guidance and thoughtfulness. I would also like to thank Dr. T. Vaughn Blalock and Dr. Donald W. Bouldin, the members of my graduate committee, and Dr. Daniel B. Koch, for their advice, teaching and help during the past years. Thanks also goes to many of the graduate students for their friendship.

I would like to express my special thanks to Mr. Ken Skinner of Philips Laboratories, for his wise advice, patience and kindness. My gratitude also goes to Mr. Ralph Booker for his initial tutorial and continuous assistance.

Last, but not least, I would like to acknowledge my wife, Jie Zhu, for her loving support, helpful suggestions and unending encouragement which have given me much motivation during the whole process.

ABSTRACT

Minimization of color errors in television receivers due to different display white set-up and primaries from NTSC standards was investigated. Neural networks, implemented with NeuralWorks software, were the proposed solution. Different structures of neural networks were built and tested with different training presentations. The best result was obtained with a backpropagation neural network with one hidden layer of seven processing elements. About 70% of color errors were reduced. The color error reduction was displayed with a CIE chromaticity diagram. The color error reduction was also tested with a video graphics system. Color measurement was made with this video color test system set-up that included a personal computer, a video graphics board and associated software called STAGE and ONSTAGE, and image data processing programs written in C++, an analog monitor, and a chroma meter. Both the simulation chromaticity diagram and the video color test system demonstrated the effects of the neural network.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Problem Description

Presently most of the color television manufacturers have changed the display White set-up from the NTSC (National Television System Committee) standard to a more bluish-white set-up. This was done because of customers' preference. In addition, different display primaries from those of the NTSC standard were used for a brighter display. However, in doing these, many other colors significantly differed from the NTSC standard.

To illustrate the color errors, a CIE chromaticity diagram [1], whose theory is discussed in greater detail in Chapter II, was produced as shown in Figure 1.1 with a computer simulation program called CHIPS7 [2], which models an ideal TV channel from the source scene to the receiver CRT display. In a CIE chromaticity diagram, any color is represented with its (x, y) coordinates. The triangle encloses all the colors a TV receiver can display [3], with Red, Green, and Blue at the vertices. In Figure 1.1, the little squares represent source colors, while the diamonds represent display colors. The distances between the squares and diamonds indicate the color errors.

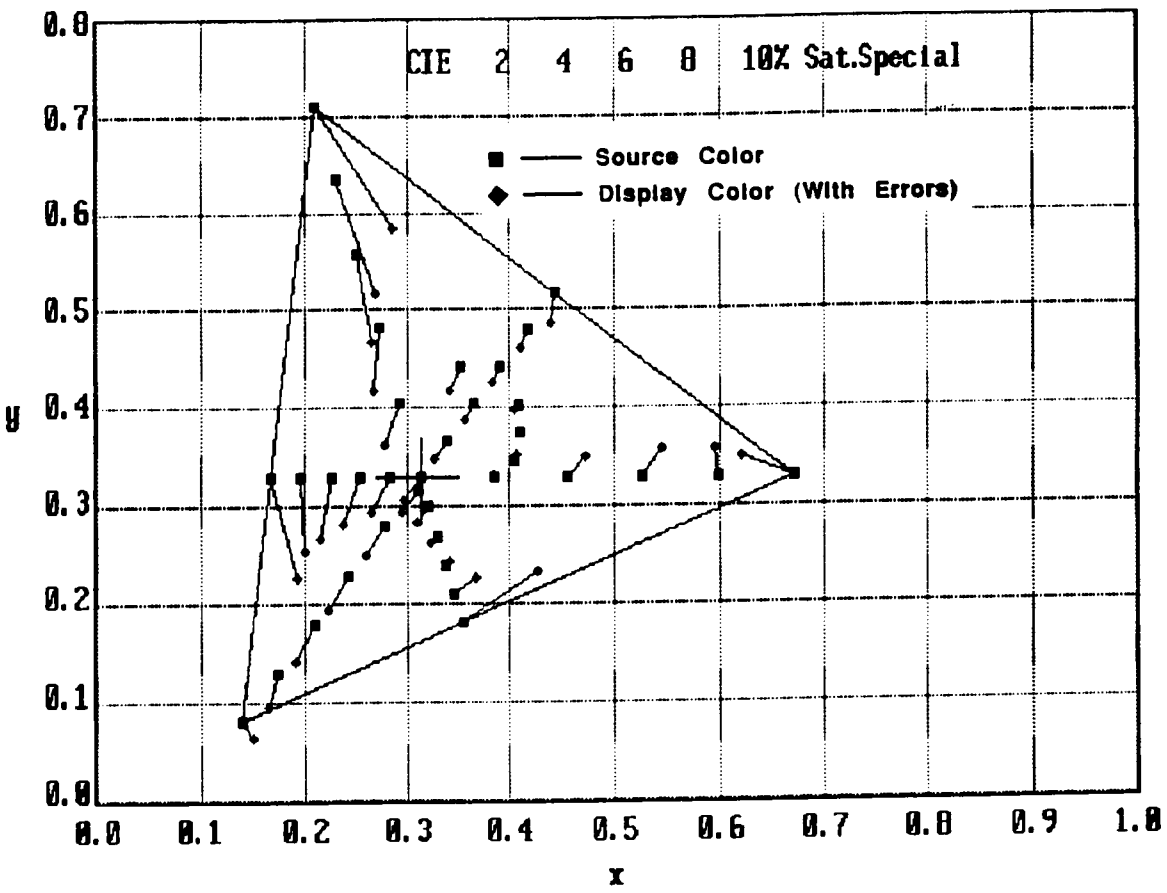


Figure 1.1 Chromaticity Diagram Showing Color Errors.
(White Set-up at 8200^0 K)

Previous Solutions

Much work has been done so far to try to correct these color errors [4, 5]. N. W. Parker [4], for instance, developed a method of color error corrections for non-standard receiver primaries. This method was developed for the Sulphide green and blue phosphors. It was general in nature and might be used to develop a correction matrix for phosphors with arbitrary color coordinates. A linear matrix, which gave an approximate correction, could be simulated by adjusting the demodulating gains and angles of the color decoder. However, with this method, only two colors, e.g., flesh and grass, can be set to be free of color error.

C. Bailey Neal [5] also presented methods for computing corrective matrices and introduced an error evaluation procedure using a coordinate system. He also described how data could be generated so that errors could be determined and alternative corrective matrices compared. However, it has been found that a linear correction matrix at or following the chrominance signal demodulator is not adequate. It seems that a nonlinear correction is necessary.

Possible Approaches

To probe other methods for color error corrections, another computer program, CHIPS8, was developed. This program determines the color difference data $(R\text{-}Y)_d$, $(G\text{-}Y)_d$, and $(B\text{-}Y)_d$ at the color demodulator output for a set of given TV receiver characteristics as well as the required color difference data $(R\text{-}Y)_r$, $(G\text{-}Y)_r$ and $(B\text{-}Y)_r$ to produce zero color errors from the given CRT display. The required color difference data were calculated backwards from the display, given zero color errors. At this point, a possibility for color error reduction exists if some way is found to transform the color difference signals from $(R\text{-}Y)_d$, $(G\text{-}Y)_d$, and $(B\text{-}Y)_d$ to $(R\text{-}Y)_r$, $(G\text{-}Y)_r$, and $(B\text{-}Y)_r$, which produce zero color errors.

Different methods were proposed for the color difference signal transformation. Two of them were given serious consideration.

A. Nonlinear Correction Matrices

The following correction matrix method was proposed for investigation, wherein, a combination of linear and nonlinear operations was performed on the color difference signals:

$$\begin{bmatrix}(R-Y)_r\\(G-Y)_r\\(B-Y)_r\end{bmatrix}=\begin{bmatrix}K11 & K12 & K13\\K21 & K22 & K23\\K31 & K32 & K33\end{bmatrix}*\begin{bmatrix}(R-Y)_d^2\\(G-Y)_d^2\\(B-Y)_d^2\end{bmatrix}+\begin{bmatrix}L11 & L12 & L13\\L21 & L22 & L23\\L31 & L32 & L33\end{bmatrix}*\begin{bmatrix}(R-Y)_d\\(G-Y)_d\\(B-Y)_d\end{bmatrix}+\begin{bmatrix}M11 & M12 & M13\\M21 & M22 & M23\\M31 & M32 & M33\end{bmatrix}*\begin{bmatrix}(R-Y)_d(G-Y)_d\\(G-Y)_d(B-Y)_d\\(B-Y)_d(R-Y)_d\end{bmatrix}$$

where subscript d -- demodulator output
subscript r -- resultant after correction

The goal was to determine the coefficients of matrices **[K]**, **[L]** and **[M]** that minimized the errors between the demodulator outputs and the required color difference signals.

An analytical solution is difficult since both the $(R-Y)_d$, $(G-Y)_d$, $(B-Y)_d$ and $(R-Y)_r$, $(G-Y)_r$, $(B-Y)_r$ are so varied. Numerical transformation techniques have been considered. However, it is difficult to derive a solution since there is such a large number of demodulator output values.

B. Neural Network Method

Neural Networks have been extensively studied during the last decade [6]. A neural network is a computing system which is useful for complicated problems where an exact mathematical model is hard to establish. Since an analytical or numerical solution is difficult to obtain for the color error reduction problem, the neural network method was proposed for this purpose. The basic theory of neural network technology is discussed in Chapter III.

Thesis Goal

The purpose of this thesis was to investigate the possibility and approaches of minimization of color errors in color television receivers. The emphasis focused on neural network method.

Chapter I is an introduction to the color error problem, which gives some background knowledge.

Chapter II presents television colorimetry and the basic theory of color television receivers. This sets up the basis for the reader to understand the following procedure for the color error reduction.

Chapter III describes the basic theory, structure and applications of neural networks.

Chapter IV is the main part of this thesis. The neural network solution is described in detail. The design, training, improvement, testing, and results are given in this chapter.

Chapter V describes a color test system. Color errors as well as their reductions were displayed using this video graphics system.

Finally, conclusions and future work are discussed in Chapter VI.

CHAPTER II

COLOR TELEVISION THEORY

Television Colorimetry

The science of colorimetry states that three independent quantities are sufficient to specify a color, and that color quantities add linearly. Therefore, colors can be represented as points in a three-dimensional space, using any convenient set of coordinates [7]. These coordinate systems include the intensities of three color primaries, or luminance, and two color quantities such as wavelength and purity. In a television system, similar electrical coordinates such as R, G and B channel voltages, or luminance and two color difference signals (Y, R-Y, and B-Y) can be used.

The fundamental quantities in colorimetry are the tristimulus values which are defined by [7, 8, 9]:

$$\begin{aligned} X &= \int E(\lambda)R(\lambda)x(\lambda)d\lambda \\ Y &= \int E(\lambda)R(\lambda)y(\lambda)d\lambda \\ Z &= \int E(\lambda)R(\lambda)z(\lambda)d\lambda \end{aligned} \qquad (2\text{-}1)$$

where X, Y, Z are the tristimulus values.

$E(\lambda)$ is the frequency spectrum of the illuminant.

$R(\lambda)$ is the frequency spectrum of the reflection of the object.

$x(\lambda)$, $y(\lambda)$, $z(\lambda)$ are the psychophysical spectra of the standard observer as defined by the Commission Internationale de l'Eclairage (CIE) and as shown in Figure 2.1.

Figure 2.2 shows the (X, Y, Z) tristimulus space, the unit plane (X + Y + Z = 1) and the location of the (x, y) chromaticity diagram -- a projection of the unit plane onto the XY plane. The tristimulus values can be converted to chromaticity coordinates by transferring the point where a ray from the origin to the point X, Y, Z in space intersects the unity plane to the XY plane. The following equations give the chromaticity coordinates in terms of the tristimulus values:

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z} \qquad (2\text{-}2)$$
$$z = \frac{Z}{X+Y+Z}$$

The dominant wavelength and purity of a color are specified by its x and y coordinates. There is a relationship among x, y and z:

$$x + y + z = 1 \qquad (2\text{-}3)$$

because the values are determined by the point of intersection with the unity plane.

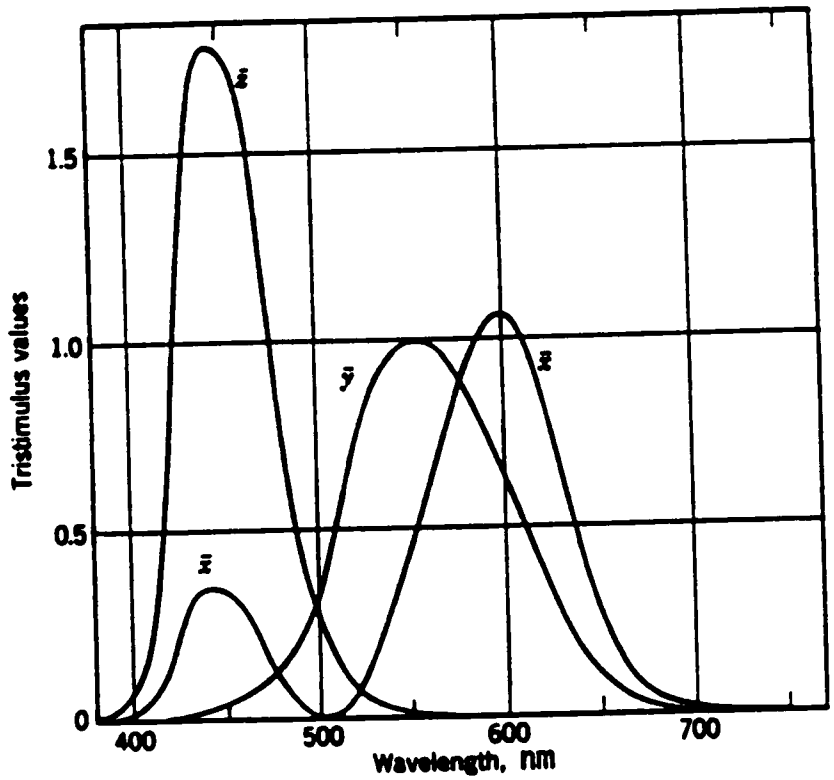


Figure 2.1 Tristimulus Values of the 1931 CIE Standard Observer for the (X), (Y), (Z) Primaries and Equal-energy White Stimulus.

Source: C. Bailey Neal, "Television Colorimetry for Receiver Engineers", IEEE Transactions of BTR, Vol. BTR-19, No. 3, August, 1973, pp. 149-162.

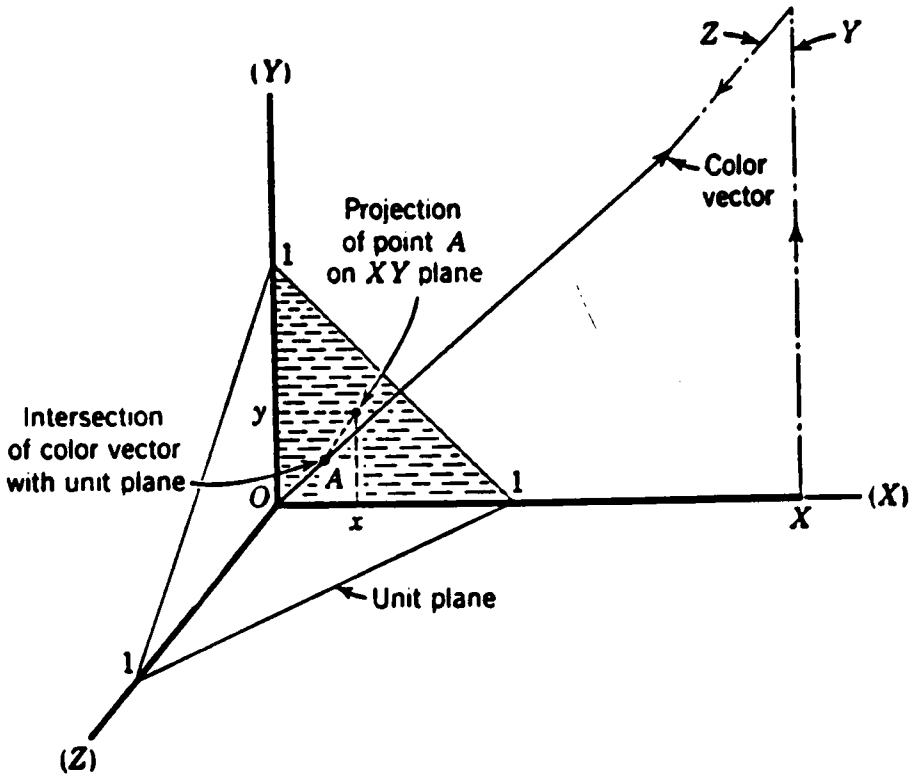


Figure 2.2 The (X, Y, Z) Tristimulus Space and the Location of the (x, y) Chromaticity Diagram.

Source: C. Bailey Neal, "Television Colorimetry for Receiver Engineers", IEEE Transactions of BTR, Vol. BTR-19, No. 3, August, 1973, pp. 149-162.

The inverse relationships are needed at the camera because the starting point is the chromaticity coordinates of the scene color:

$$X = Y \times \frac{x}{y}$$
$$Y = Y \qquad (2\text{-}4)$$
$$Z = Y \times \frac{1-x-y}{y}$$

where Y is the luminance value of the color. The Y signal is a proportional combination of red, green, and blue signals. For the NTSC color system, the relationship is described as follows:

$$1.00Y = 0.30R + 0.59G + 0.11B \qquad (2\text{-}5)$$

The three values, x, y and Y, specify a color completely.

Figure 2.3 shows the CIE (x, y) chromaticity diagram which shows the spectrum locus and purple (spectral red/violet mixture) boundary.

A color television display, usually a cathode ray tube, provides red, green and blue primary light sources controlled by three electrical quantities. These quantities are often the channel voltages referred to as R, G, and B. It is assumed that the primary outputs are

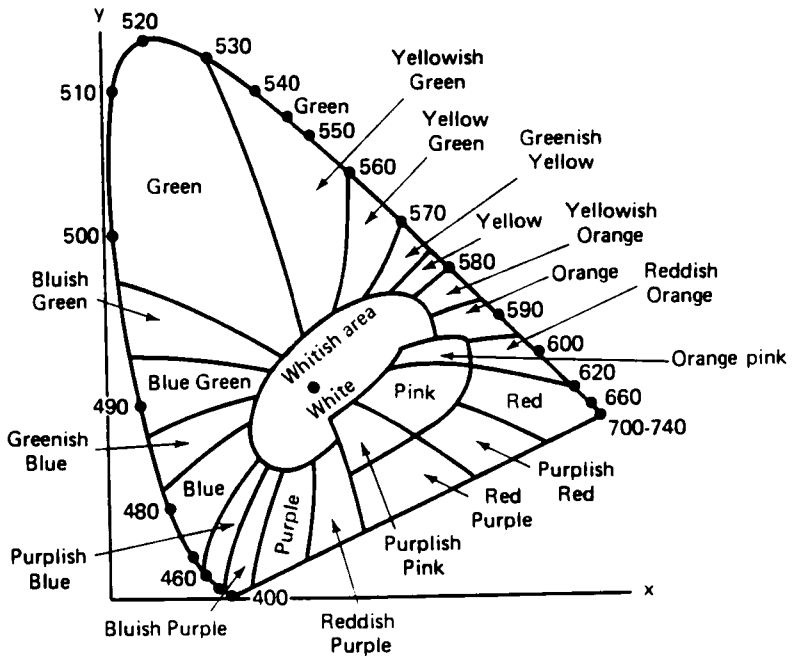


Figure 2.3 A Standard Chromaticity Diagram.

Source: Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

linearly related to the channel voltages as [3]:

$$[S] = [A]\,[V] \tag{2-6}$$

where $[S]=\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$, $[V]=\begin{bmatrix} R \\ G \\ B \end{bmatrix}$

and [A] is a 3×3 matrix which is determined by the phosphor chromaticity coordinates and normalizing white illuminant (e.g. D6500). Inversely, the R, G, B channel voltages can be given by

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = [A]^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2-7}$$

A color television camera is intended to provide R, G, B channel voltages suitable for a display with specified primary chromaticity coordinates and specified reference white. The NTSC/FCC (Federal Communications Commission) standards define the primary chromaticities as:

R: $x = .67, \quad y = .33$

G: $x = .21, \quad y = .71$

B: $x = .14, \quad y = .08$

The NTSC/FCC standards give numerical values for the signal specifications [10, 11] which assume the reference white to be illuminant D6500 (x = .3127, y = .3291) with temperature 6500^0 K. Color temperature is a term sometimes used to describe a radiant energy source whose color matches that of a black body radiator at that temperature. A black body radiator at any given temperature emits light, according to Planck's radiation law, having a certain color characteristic.

Basic Theory of Color Television Systems

To transmit the R, G, and B signals produced by cameras by means of electromagnetic waves, three television channels could be employed. This was the method used by the pioneer workers. However, if three 6-MHz television channels were allocated to each color TV transmitter, only one-third as many TV broadcast stations could have been accommodated in the VHF and UHF bands. Moreover, a three-channel transmission system would not have been compatible with existing black-and-white TV receivers. Therefore, a more sophisticated system was developed with the objective of providing complete compatibility between color and black-and-white receiver operation. The basic plan of the NTSC system is to add hue and saturation information to the brightness information contained

in a standard black-and-white TV signal. A modulator is needed for this purpose.

A color modulator is a subsystem that processes the outputs from three color cameras to form the NTSC color signal. To modulate means to translate. One basic function of a color modulator is to change the red, green, and blue primary signals into R-Y, G-Y and B-Y transmission primary signals [12]. Figure 2.4 shows the basic plan of a color modulator.

In a color television receiver, the opposite function is done. The color signal received is demodulated. A basic block diagram for a color television receiver is shown in Figure 2.5. Observe that the chroma signal section in this figure operates to amplify the chroma signal and demodulate the signal into its R-Y, G-Y and B-Y components, as shown in Figure 2.6. These R-Y, G-Y and B-Y chroma signals combine with the Y signal at the picture tube to form the red, green and blue primary hues. This operation is seen in Figure 2.7. A complete block diagram of a color television receiver is seen in Figure 2.8.

In summary, a complete color television channel operates like this:

1. Three color TV cameras are used to take scene data.
2. The outputs from these cameras are processed with matrix and modulators to produce color difference signals such as (R-Y) and (B-Y) and luminance Y.

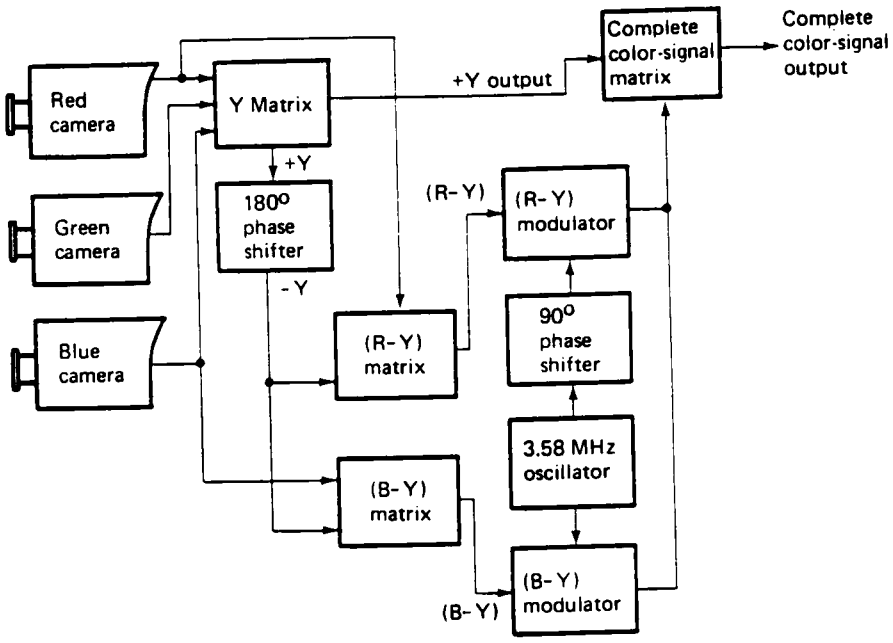


Figure 2.4 Basic Plan of a Color Modulator.

Source: Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

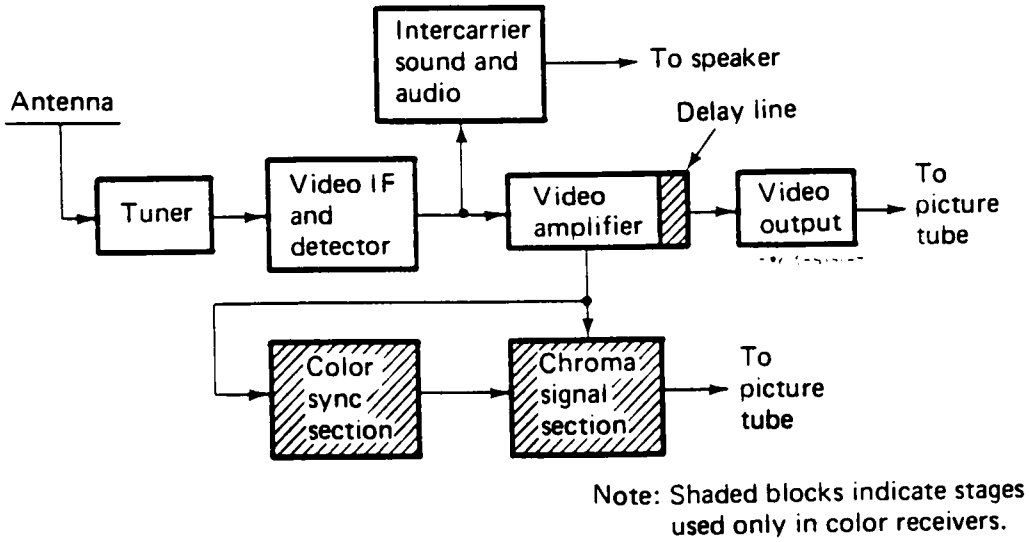


Figure 2.5 Basic Block Diagram for a Color Television Receiver.

Source: Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

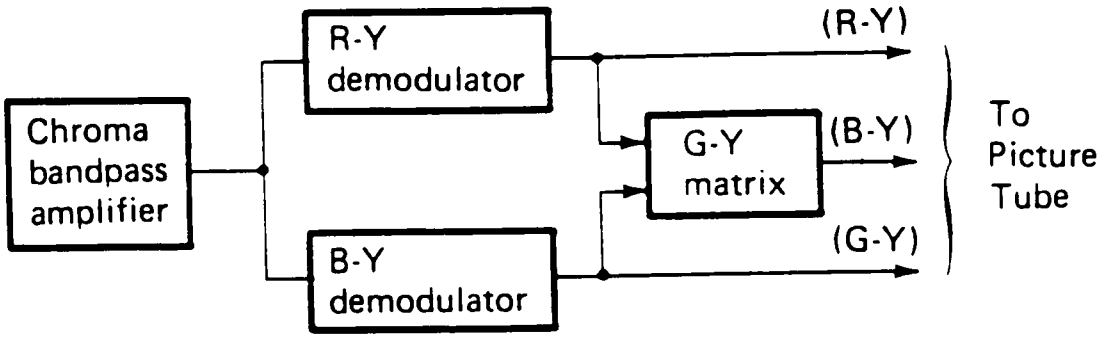


Figure 2.6 Plan of the Chroma Signal Section.

Source: Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

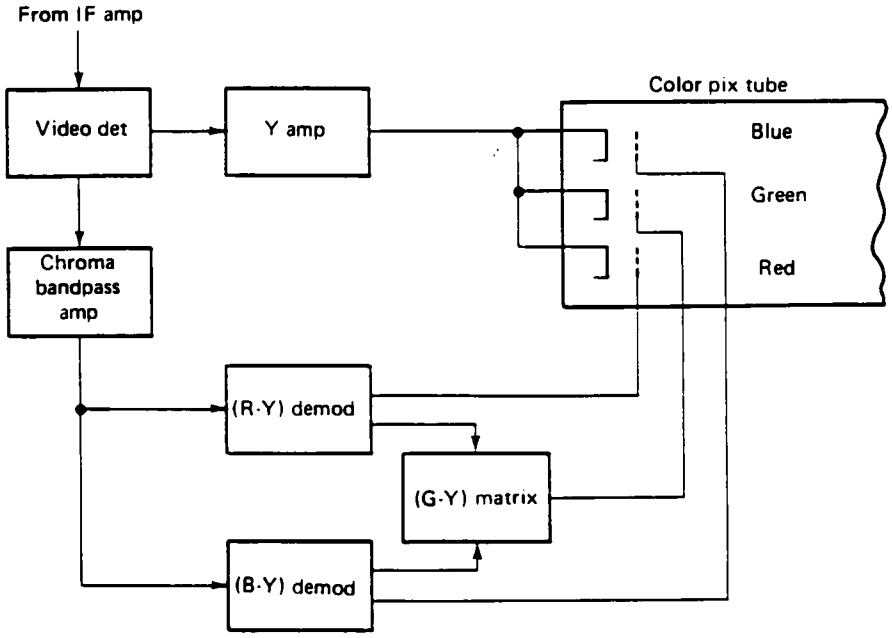


Figure 2.7 Color Picture Tube Operation as an RGB Matrix.

Source: Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

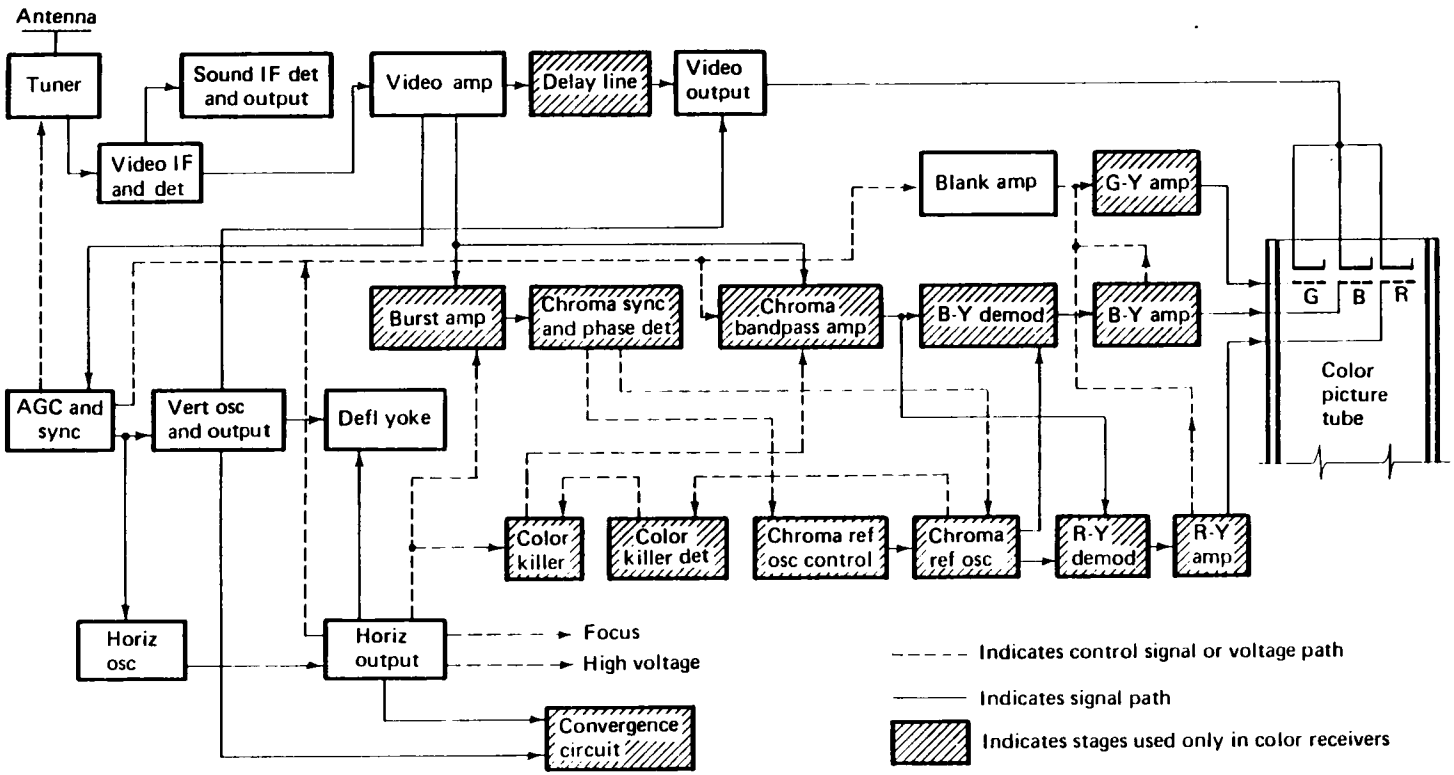


Figure 2.8 Block Diagram of a Color Television Receiver.

Source: Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

3. The color difference and luminance signals are combined through a matrix to produce complete color-signal output, which is transmitted through transmission channel.
4. In a color TV receiver, the chroma signal is processed in a chroma signal section, where it is amplified, and demodulated into color difference signals.
5. Finally, the color difference signals and the luminance signal from video section are combined in the picture tube to reproduce the color scene.

CHAPTER III

NEURAL NETWORK THEORY

What Are Neural Networks

Neural Networks have been extensively studied during the last decade. To understand why and how artificial neural networks behave as they do, it is desirable to understand some of the fundamental characteristics of biological neural networks, commonly known as the brain and nervous system. The brain is composed of billions of nerve cells, or neurons, arranged in layers or groups. Each biological neuron is made up of one axon and one or more dendrites. Figure 3.1 shows a biological neuron. If a biological neuron is considered as a processing element (PE), an axon, which carries impulses away from the neuron, can be thought of as an output. Similarly, dendrites, which carry impulses toward the neuron, can be considered as inputs. Each neuron's output branches out to the inputs of many other neurons. The points at which they connect with other neurons form synapses. Synapses provide pathways of varying strength for the transmission of impulses from one neuron to another, forming the network.

The basic processing element in an artificial neural network is the neuron, a model of which is shown in Figure 3.2. Inputs from

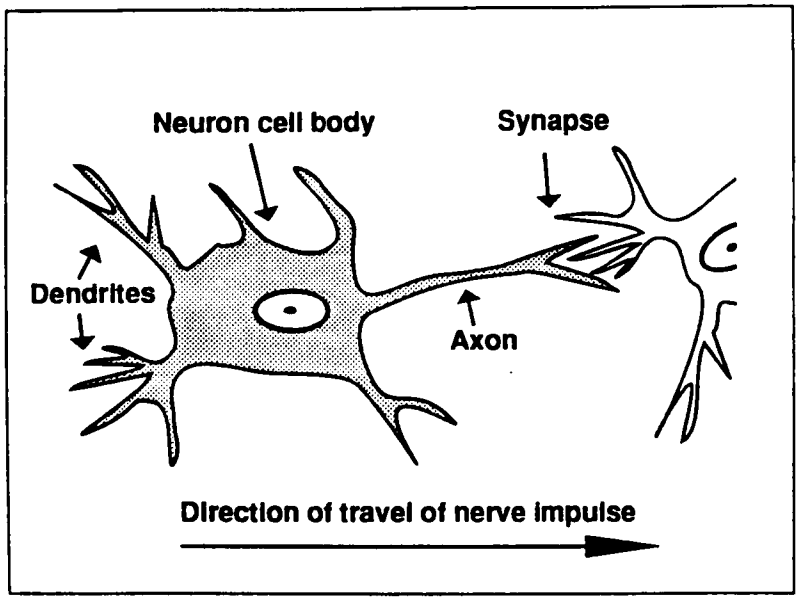


Figure 3.1 A Biological Neuron.
Source: NeuroDynamX, Inc., DynaMind User's Guide, 1991.

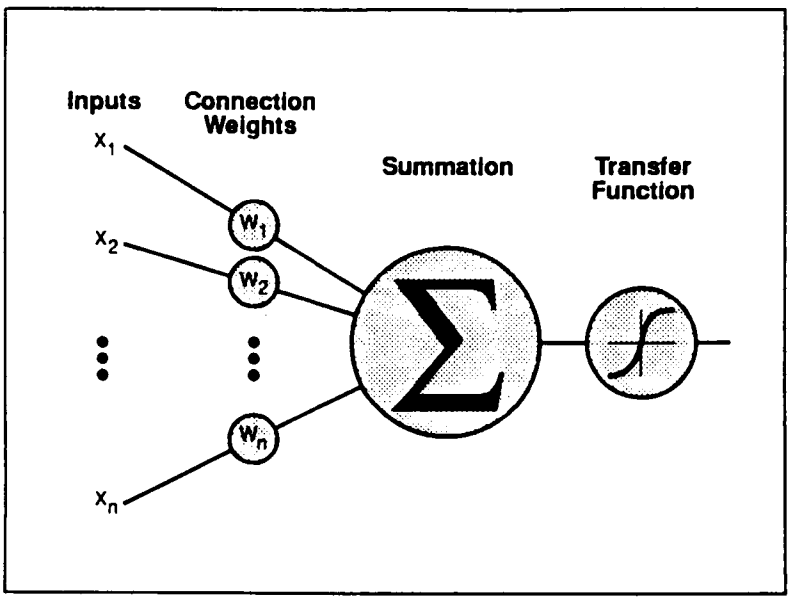


Figure 3.2 An Artificial Neuron.
Source: NeuroDynamX, Inc., DynaMind User's Guide, 1991.

other neurons or the outside world are first multiplied by stored connection weights. The weighted inputs are then summed and processed through a transfer function. The transfer function, typically nonlinear, modifies the weighted sum of the input values to a reasonable value before passing the signal on to other neurons.

Neurons, or PEs, are organized into layers. A neural network is made up of a few layers, such as, input layer, hidden layers and output layer, with a number of simple, highly interconnected PEs. It processes information by its dynamic state response to external inputs. Figure 3.3 is an example of neural network architecture. The PEs are connected via weights that are typically adapted during use to improve performance (i.e., "training" or "learning").

A neural network learns from being shown examples. Learning is achieved through a learning rule which adapts the connection weights of the network in response to the example inputs in unsupervised learning, and optionally, the desired outputs of those inputs in supervised learning. For different structures and learning rules, neural networks can be classified as Perceptron, Hopfield, Backpropagation, Counterpropagation, and Adaptive Resonance networks. There are also different linear and nonlinear transfer functions, such as, Threshold, Step, Sigmoid, Hyperbolic Tangent and Gaussian.

An often used learning system in neural networks is backpropagation. A backpropagation element transfers its inputs as

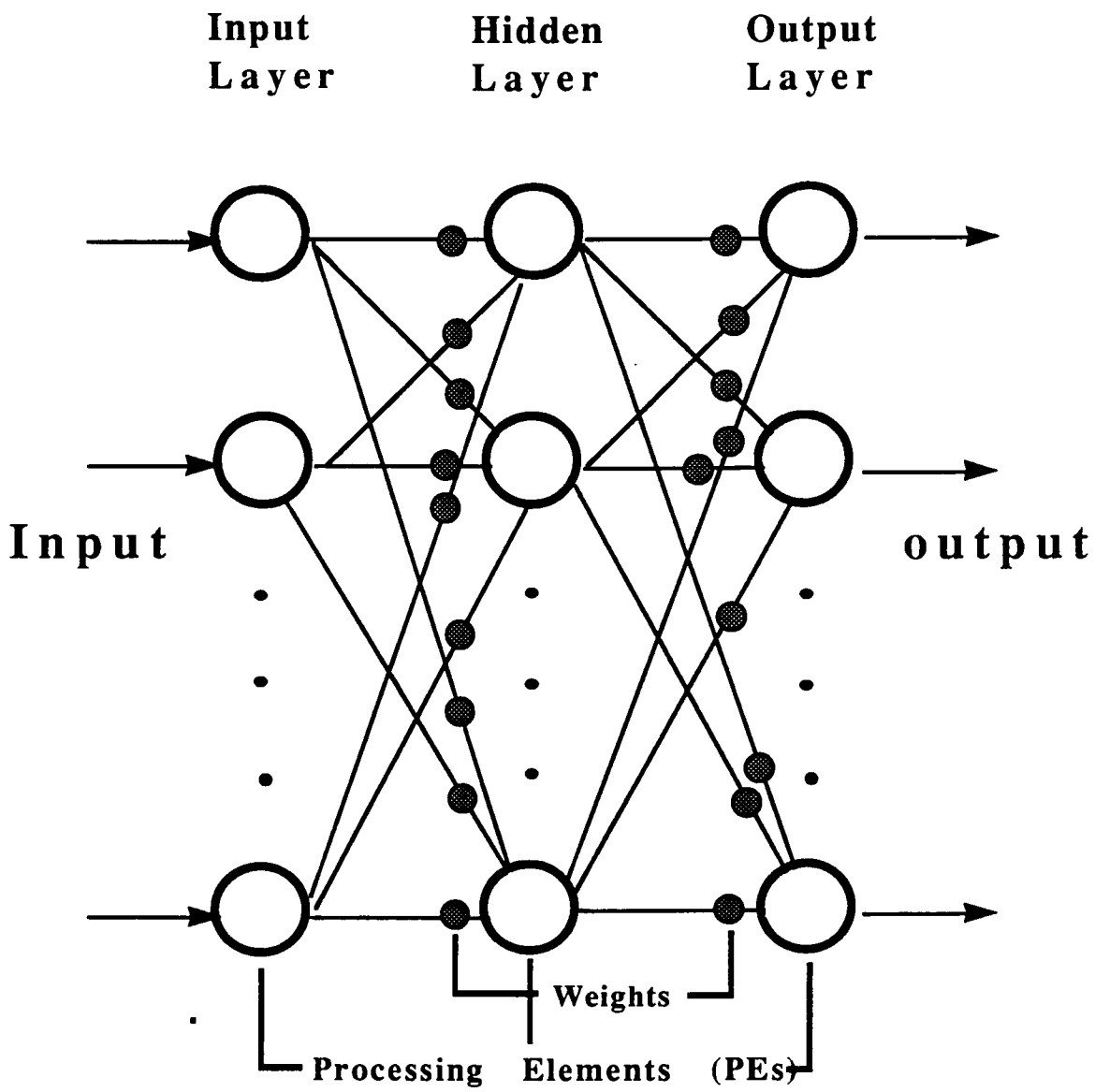


Figure 3.3 Neural Network Architecture.

follows [15]

$$X_j^{[s]} = f(\sum_i (W_{ji}^{[s]} * X_i^{[s-1]})) = f(I_j^{[s]}) \quad (3\text{-}1)$$

Where $X_j^{[s]}$: Current output of jth neuron in layer s.

$W_{ji}^{[s]}$: Weight on connection joining ith neuron in layer (s-1) to jth neuron in layer s.

$X_i^{[s-1]}$: Current output of ith neuron in layer s-1.

$I_j^{[s]}$: Weighted summation of inputs to jth neuron in layer s.

f: Sigmoid function that is defined as

$$f(z) = (1 + e^{-z})^{-1} \quad (3\text{-}2)$$

The main mechanism in this network is to forward propagate the input through the layers to the output layer, determine the error between the actual output and the desired output, and then back propagate the errors through the network from the output layer to the input layer. This backward propagation of errors is used to modify the connection weights with Delta rule [15] until the actual outputs are close enough to the desired outputs. The Delta rule, which follows the gradient of a curve, is called a gradient-descent learning rule based on a least mean square error criterion. The adjustment of the weights $w_{ij}^{[s]}$ can be done as follows [15]:

$$\Delta w_{ji}^{[s]} = lcoef \times e_j^{[s]} \times x_i^{[s-1]} \qquad (3\text{-}3)$$

where *lcoef* is a learning coefficient

$e_j^{[s]}$ is the local error at processing element j in level s

What Makes Neural Networks Different

Neural networks differ from either conventional computer computation or expert systems [13]. Conventional computers approach problems from a digital orientation where information is represented by either 1's or 0's, answers are either yes or no, and data are presented in unequivocal mathematical and logical output. To achieve such certainty when dealing with complex problems or ambiguous information, however, traditional computers require sophisticated programs, and even then may not provide the desired result. In expert systems, knowledge is made explicit in the form of rules, which are known beforehand.

However, neural networks generate their own rules by learning from being shown examples. Learning is achieved through a learning rule which adapts the connection weights of the network in response to the example inputs, and optionally, the desired outputs of those inputs in supervised learning. It turns out that neural networks are most useful for complicated problems where an exact mathematical model is hard to establish [14].

Neural Network Applications

The special characteristics of neural computing systems make neural computing especially useful in a variety of applications. Neural computing provides an approach which is closer to human perception and recognition than traditional computing. In situations where input is noisy or incomplete, a neural network can still produce reasonable results. A sampling of applications that have met with some success using neural networks is listed in the following [15, 16].

. Language Processing
. Image (data) Compression
. Character Recognition
. Combinatorial Problems
. Pattern Recognition in Images
. Signal Processing
. Financial and Economic Modeling
. Servo Control
. Functional Synthesis
. Robotic Arms

Above are just a few examples. More and more applications of neural networks appear increasingly.

CHAPTER IV

NEURAL NETWORK SOLUTION

On the basis of the brief description of neural networks in the previous chapter, it is natural to consider the possibility of applying neural networks to the color error problem. Many neural network programs have been developed so far. A backpropagation program is widely used and would be well suited for the color error problem. However, one must keep in mind that in general, neural networks do not do precise, numerical computations well [6]. A question arises that to what degree a neural network will solve this color error problem. Some work has been done to answer this question.

Software Selection

Software for neural networks is commercially available. The following has been evaluated:

1. NeuralWorks
2. BrainMaker
3. ExploreNet

NeuralWorks, developed by Neural Ware Inc., is a proven neural network software. There are many different programs in this

software. Among them, Back propagation is the most useful one for our purpose. Since we have some experience with this software, neural network designs for the color error problem have been quickly established.

Newly developed by California Scientific Software, BrainMaker is quite popular in recent years [17]. It is chosen by Intel for its neural network chip. A network for the color error problem was created with this version. However, it took a long time to train with the large number of color difference data.

ExploreNet is a relatively new software. It requires MS-Windows for a PC and large disk space [18]. It appears to be a powerful software.

On the basis of the investigation of these three kinds of software and our PC condition (MAGNAVOX 386-SX), the NeuralWorks was chosen for our purpose.

Design of Neural Networks

To design a neural network, one must know what the application is, and what the input and output to the network will be. In the application of the color error minimization, there are three inputs, the color difference signals $(R\text{-}Y)_d$, $(G\text{-}Y)_d$ and $(B\text{-}Y)_d$. Also there are three outputs, the desired color difference signals $(R\text{-}Y)_r$, $(G\text{-}Y)_r$ and $(B\text{-}Y)_r$. A block diagram for this application is shown in

Figure 4.1. The number of PEs for input layer and output layer is fixed at 3. Only the number of PEs for the hidden layer is to be determined. There is not an analytical solution for how to determine the PE number in the hidden layer. A rule of thumb [19] is

$$n = 2m + 1$$

Where n is the upper bound of the PE number for the hidden layer and m is the PE number for the input layer.

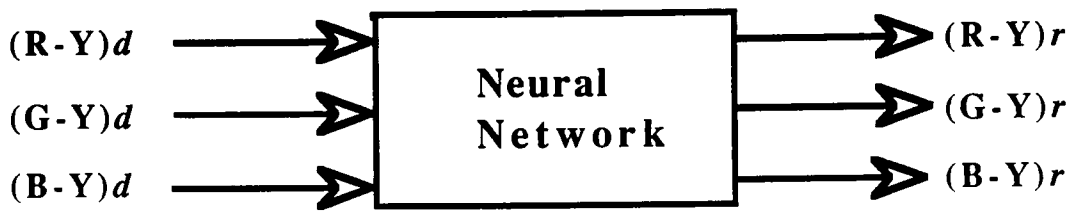


Figure 4.1 Block Diagram of the Color Network.

The neural network for this application should be like that in Figure 4.2. This is a supervised back-propagation neural network that uses the sigmoid transfer function.

Many neural networks have been built with different numbers of hidden layers and different PE numbers in the hidden layers.

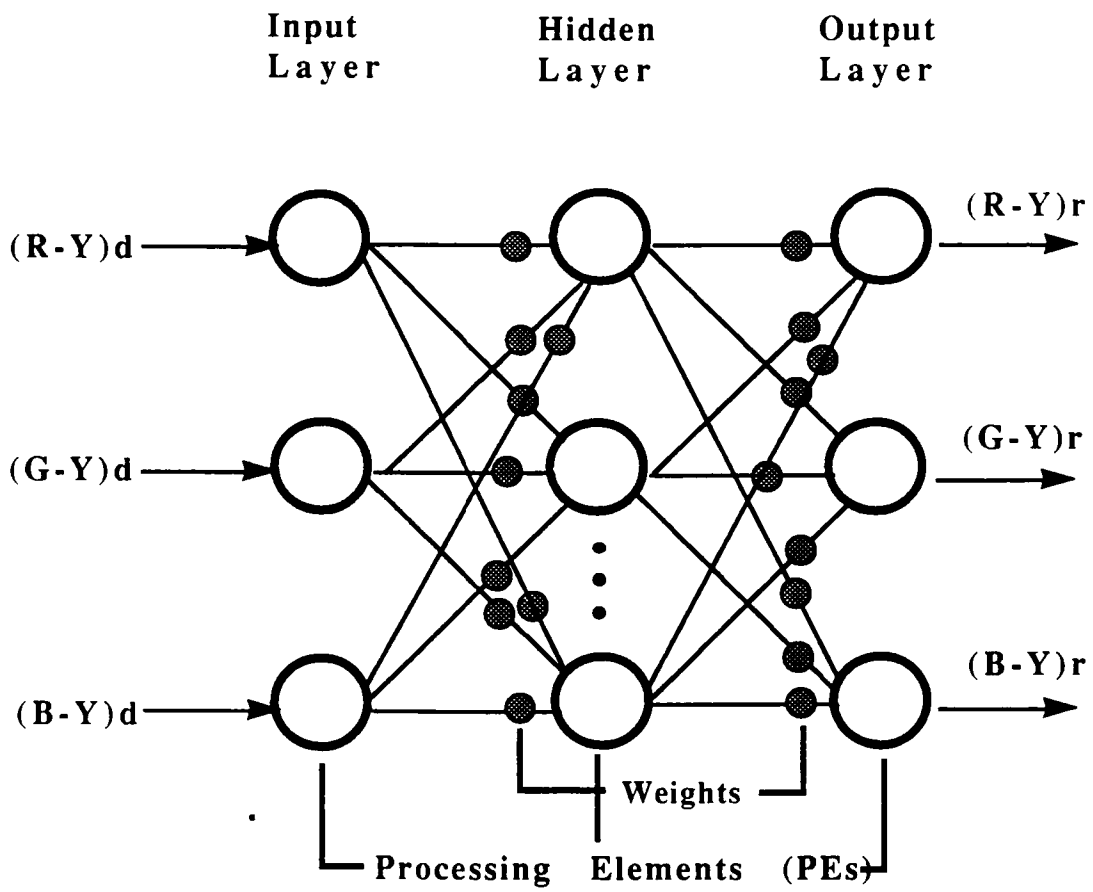


Figure 4.2 Neural Network for Color Difference Signal Processing.

Training and testing were made to those networks. It turned out that optimum results were obtained with one hidden layer of seven PEs.

Neural Network Training

The neural network (RGB_Y) with 7 PEs in the hidden layer was chosen for training with the color difference data produced from CHIPS8F2 program. These data include 360 color difference data (180 for demodulator output and 180 for desired data) for six colors with 10% saturation step size. These data were normalized to the range of 0 to 1 for the network training to use the sigmoid transfer function. The training presentations used were 1,000, 5,000, 10,000, 15,000, 30,000, 100,000 and 500,000 times. These took about 3 minutes, 15 minutes, 30 minutes, 45 minutes, 1.5 hours, 4.5 hours and 20 hours respectively. Usually the accuracy improved with additional training presentations, but with diminishing returns. Accuracy improved little after about 50,000 presentations.

Improvement of Neural Networks

After training, the neural network was ready to process input data. The same data as those for training were inputted. When the outputs were compared with the desired data, it was found that

about half of the outputs were closer to the desired data than the input data, while the others had little change or got worse. Therefore, this network needed some improvement. One way was to train the neural network with more uniform data. To get more color difference data, CHIPS8P2 program was created, which produced data for twelve colors. The total data number was 720, doubling the previous one.

The RGB_Y neural network was trained with the new data. Varieties of training presentations, 1,000, 5,000, 10,000, 15,000, 30,000 and 100,000, were applied to this network. After training, the network was presented with the same data to predict the desired data. The network trained with 10,000 times got the better results than the other training presentations. Comparing this result with that of the previous network which was trained with the data of six colors, it was found that the neural network was improved with more training data.

Neural Network Testing

A trained neural network needs to be tested with new data that it has never seen. For this purpose, the twelve color's data were inputted to the network trained previously with six color's data, which means that half of the input data had never been seen by that network. The output data displayed satisfactory agreement with the desired data.

Neural Network Results

Since there are a large number of input and output data, it is difficult to compare them manually. Some computer programs were needed and created. The input, output and desired data for the twelve colors were compared using one of these programs, called RGBERR3.BAS. Approximately 70% of the output data were closer to the desired data than the input data. We use RGood (%) as a measurement for good result data. The relationship between RGood and training presentation is illustrated in Figure 4.3. These good results mean that the color error was reduced. 30% showed no improvement. Fortunately, after examining the data more carefully, we found that a large majority of the data with large errors, for example, the (B-Y) data, had been improved. Table 4.1 gives an example of the data comparisons, from which it can be seen that for the (B-Y) signal of the blue color, all the data for the ten different levels of saturation were improved. (|D-I|: difference between desired data and input data, |D-R|: difference between desired data and network result data). The result is easily seen in Figure 4.4.

If observing the data at a single saturation level, say, 70%, but using all 12 colors, we see that all the network output data are closer

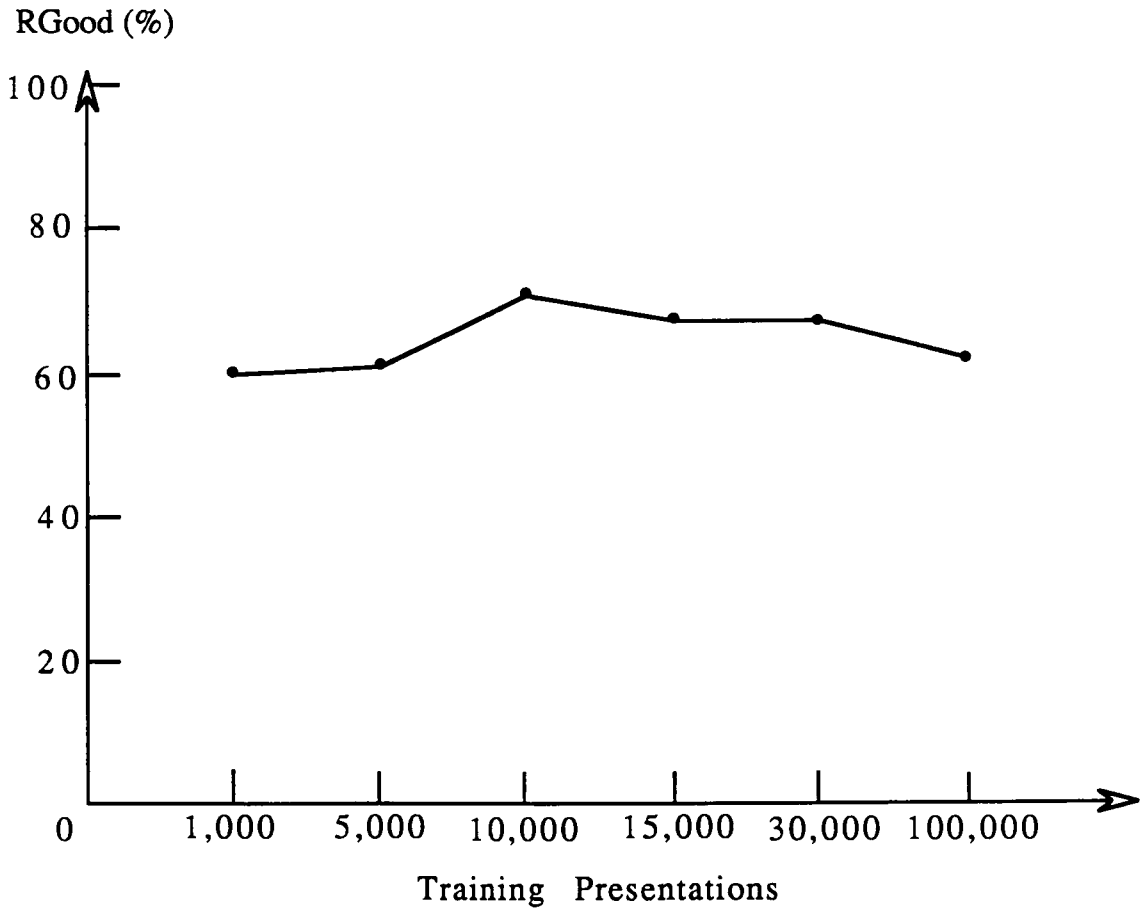


Figure 4.3 Relationship Between Training Presentations and RGood.

Table 4.1 Data Comparison for (B-Y) of Blue color

Saturation (%)	|D-I|	|D-R|
10	0.0406	0.0136
20	0.0417	0.0121
30	0.0431	0.0100
40	0.0446	0.0071
50	0.0463	0.0033
60	0.0483	0.0013
70	0.0504	0.0073
80	0.0525	0.0155
90	0.0428	0.0398
100	0.1229	0.0054

|D-I|: Difference between desired data and input data

|D-R|: Difference between desired data and network output data

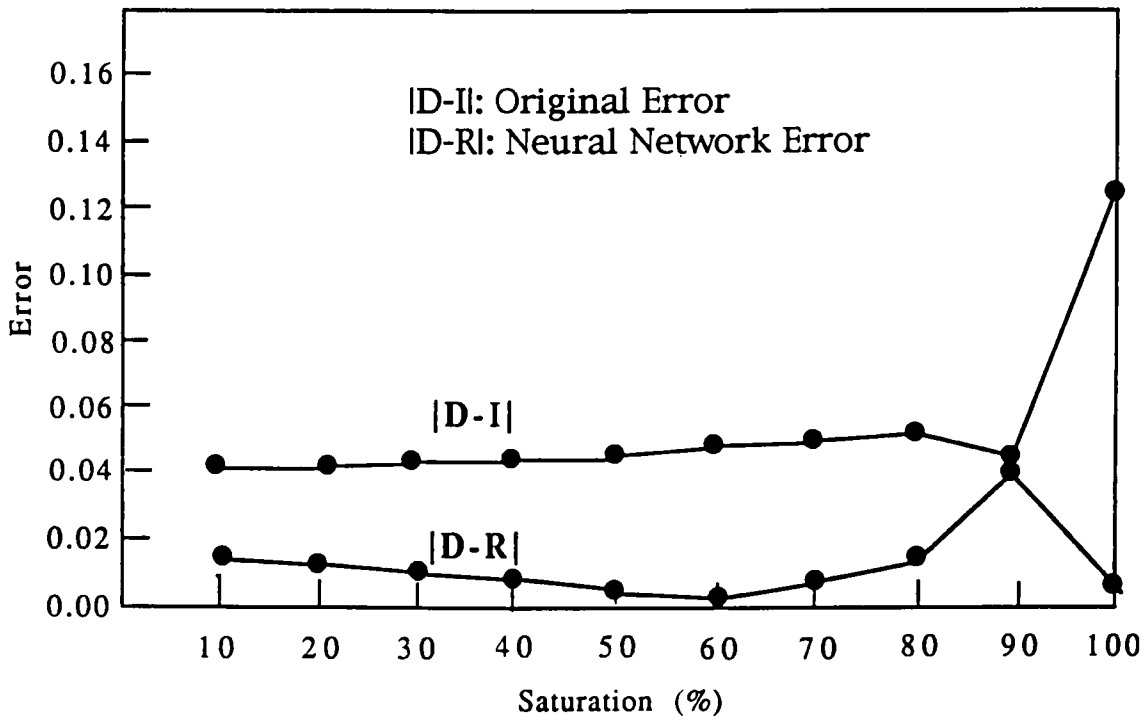


Figure 4.4 Error Comparison for (B-Y) of Blue Color.

to the desired data than the input data are (See Table 4.2). A bar chart in Figure 4.5 clearly indicates the data improvement.

From the printout of the comparison program, it is found that the original difference between the input data and the desired data could be as high as 19%, while the errors between the output and the desired data are reduced to less than 7%.

To summarize the procedure of color error correction, a block diagram is shown in Figure 4.6. CHIPS8P2 program was run to produce the color difference data. These data were processed with the neural network, RGB_Y. The input and output data from the network were compared with the RGBERR3 program to see the data improvement. The network outputs were processed with the HuetoSat program to remove the data normalization. Then, these data were inputted to a CHIP10P2 program to plot chromaticity diagrams. Finally, the CIE chromaticity diagram, Figure 4.8, and U*V* diagram [3], Figure 4.10, were compared with those diagrams from CHIPS7P2 before color error corrections, Figure 4.7 and Figure 4.9. The comparison verified that most of the color errors were greatly reduced.

Table 4.2 Data Comparison for (B-Y) of saturation 70%

Color	|D-I|	|D-R|
Red	0.1006	0.0335
Green	0.0856	0.0296
Blue	0.0504	0.0073
Yellow	0.0294	0.0061
Cyan	0.0554	0.0426
Magenta	0.0253	0.0074
Orange	0.0244	0.0039
Light Green	0.0664	0.0035
Forest Green	0.0662	0.0425
Turquoise	0.0666	0.0087
Violet	0.0141	0.0051
Purple	0.0681	0.0101

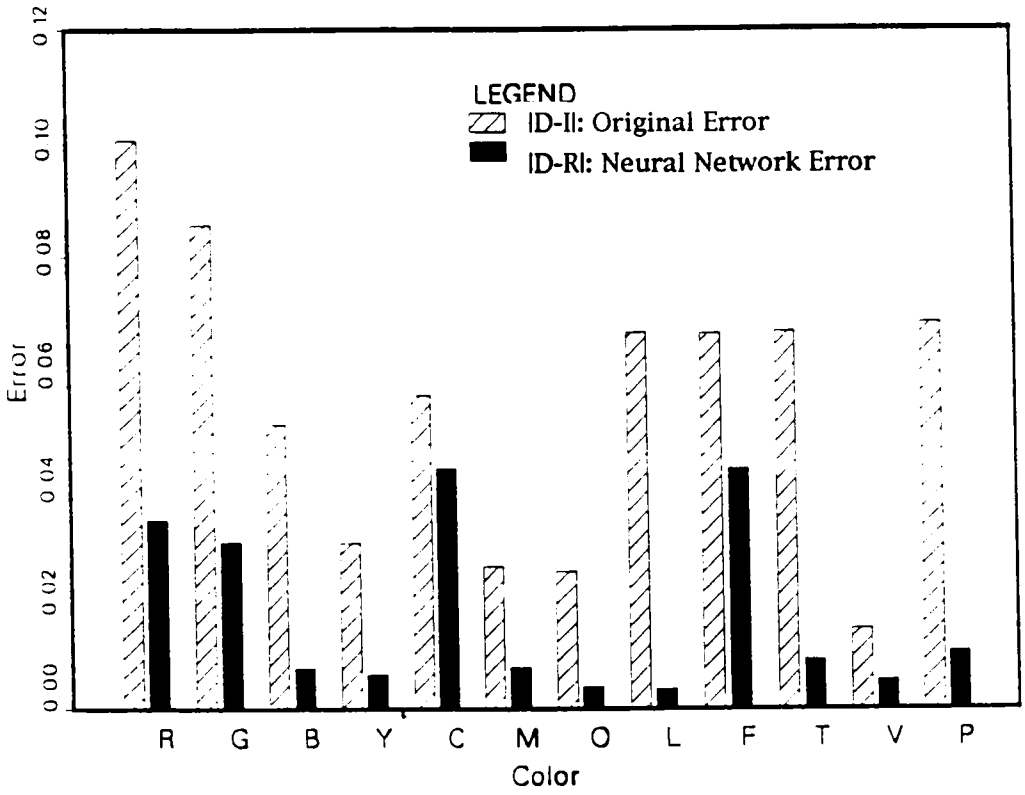


Figure 4.5 Error Comparison for (B-y) of Different Colors. (70% Saturation)

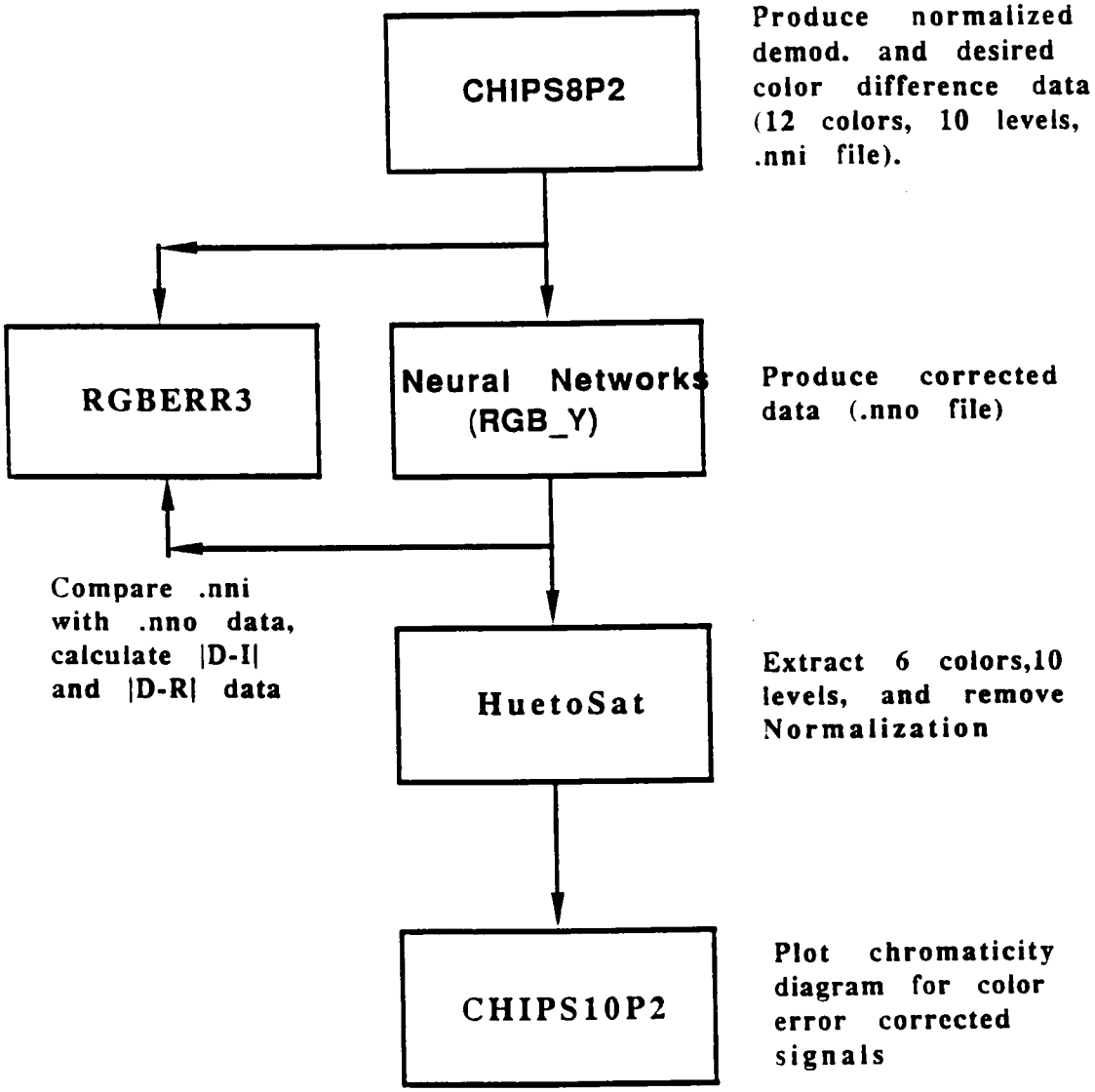


Figure 4.6 Block Diagram for Procedure of Color Error Correction Simulation.

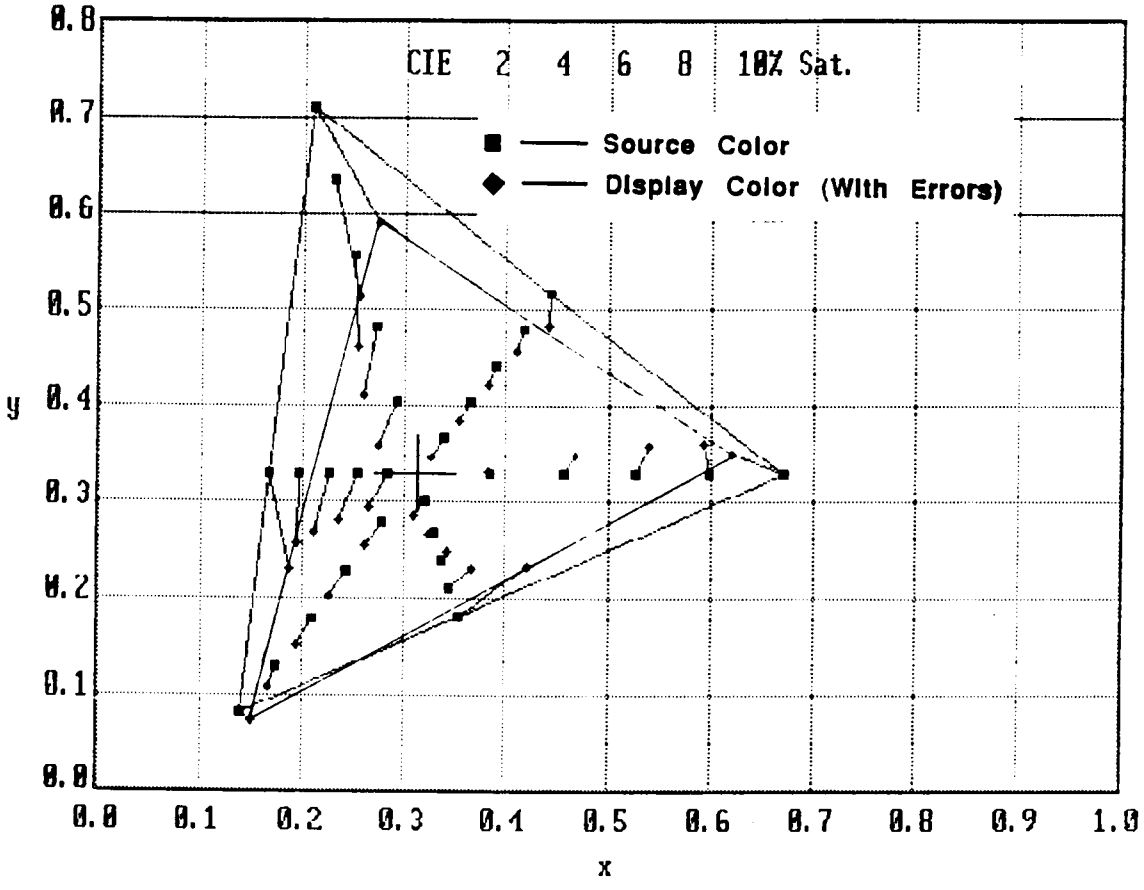


Figure 4.7 Chromaticity Diagram before Error Correction.

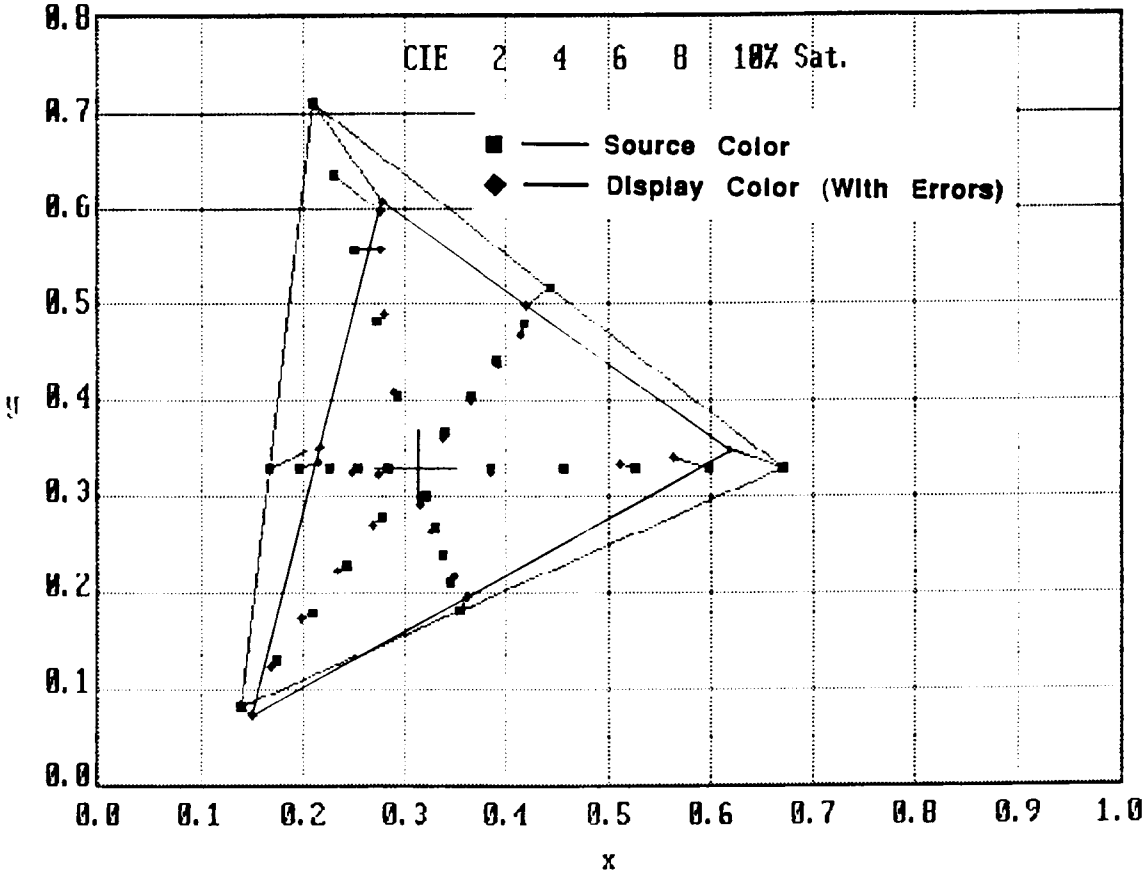


Figure 4.8 Chromaticity Diagram after Error Correction.

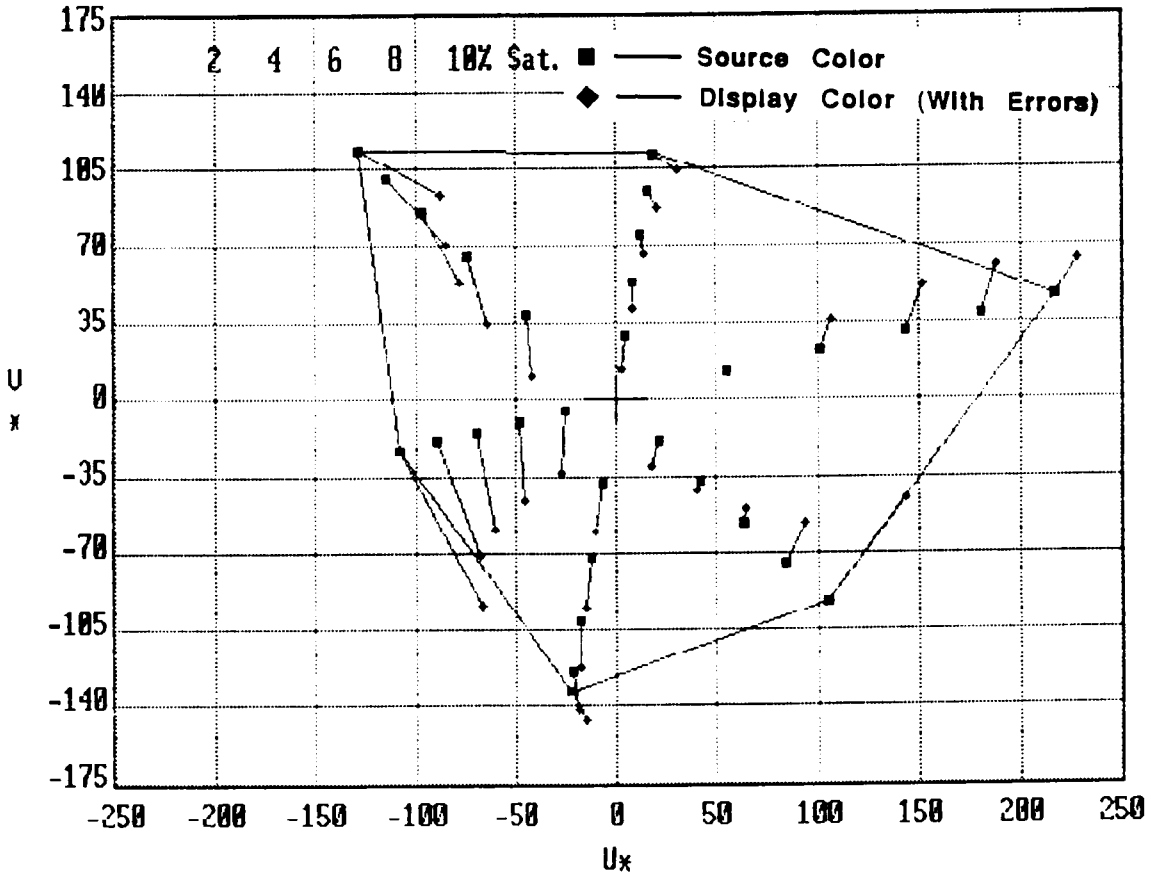


Figure 4.9 U*V* Diagram before Error Correction.

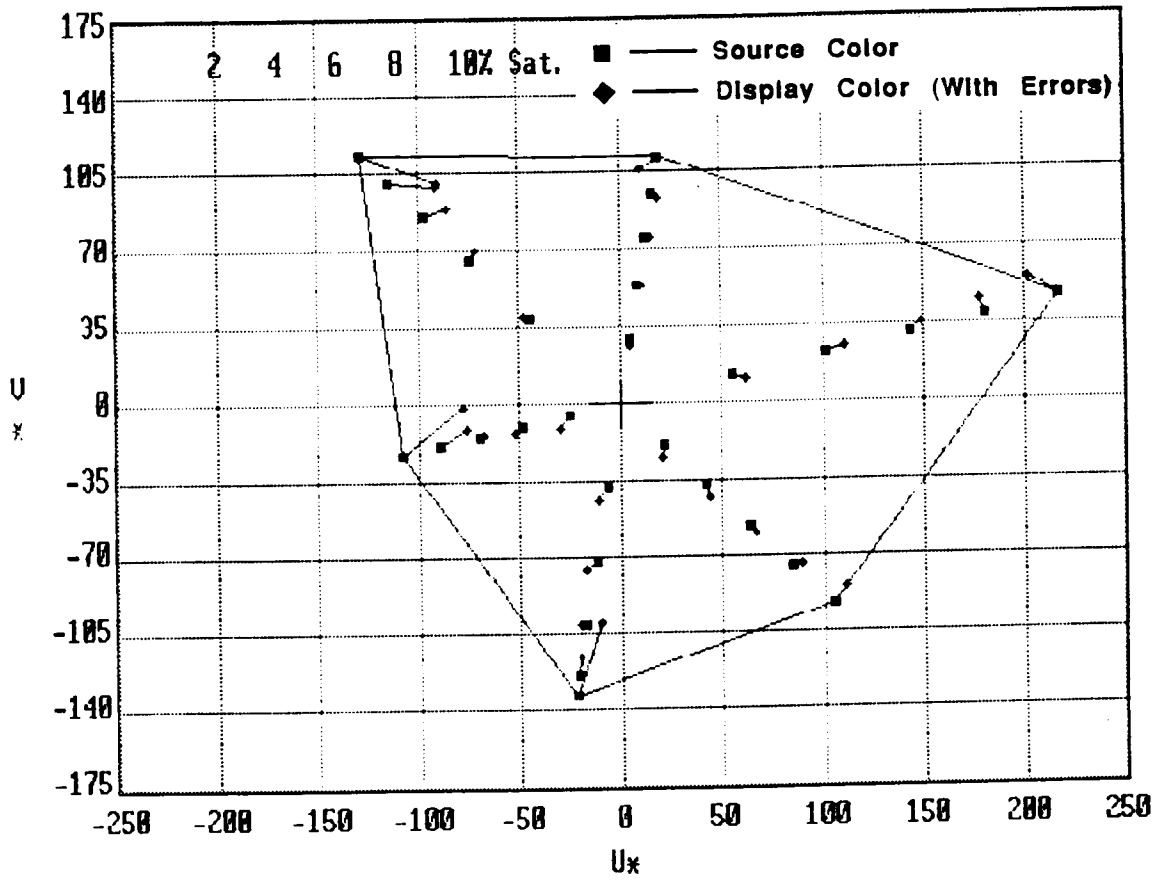


Figure 4.10 U*V* Diagram after Error Correction.

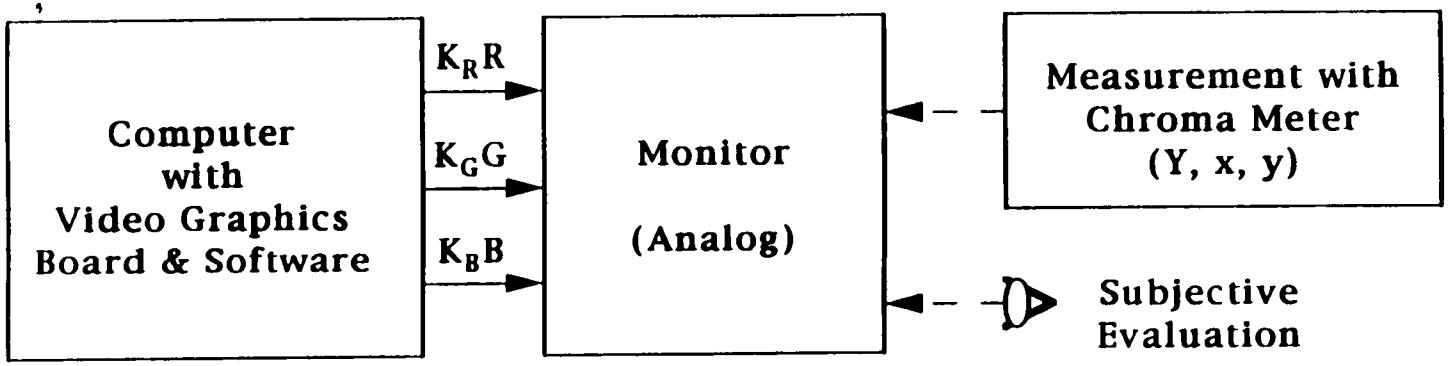
CHAPTER V

COLOR TEST AND EVALUATION

Test System Set-up

In the previous chapter, color error reduction was shown with the chromaticity diagram, which was a computer simulation. The next step is to show a more visual color error reduction. It is desired to see on a real television receiver or a monitor how the color errors are reduced. For this purpose, a color test and evaluation system was designed and assembled, which is seen in Figure 5.1. The system consists of a computer, 386 PC, which includes a video processing board (Truevision's ATVista™ Videographics Adapter [20]) and associated video graphics software (STAGE and OnSTAGE [21]), an analog RGB monitor (must be an analog monitor to work with the ATVista board), and a chroma meter (Minolta CS-100).

The CHIPS and neural network programs are run on the computer to give simulation results. The video graphics software is run to generate color bar patterns and to display frames of video. The video board outputs RGB analog signals to the monitor. Color bar patterns and selected frames of video are displayed on the monitor. The chroma meter is used for taking the color measurements of the color bars. Various processing of the video data can be displayed



CIIIPS7
CIIIPS8
Neural Network
CIIIPS10
Video Graphics Data Generation

Figure 5.1 Color Test System Set-up.

simultaneously on the monitor for direct subjective viewing and comparison. A printer is also included to record tabular data as well as graphical results.

Improvement of Computer Simulations

Although good simulation results have been obtained in the previous chapter, it is necessary to further improve the computer simulations for some reasons. Figure 5.2 shows the reasons and procedure of this improvement.

Since a RGB monitor is used for this test system, R, G, and B signals are needed to drive the monitor instead of R-Y, G-Y, and B-Y, which were used previously for the neural network training. Thus the CHIPS8P2 program was modified as CHIPS8PC to produce R, G, B data. In addition to the previous 12 colors, some special color data such as white, flesh tone and grass were also produced with this new program. White and flesh tone are the colors which most concern people. With the new R, G, B data from the CHIPS8PC program, the neural network was trained again. The output data from the neural network were run through the Hue2Sapc and CHIP10PC programs. A similar CIE chromaticity diagram like Figure 4.7 was produced from the CHIP10PC. In addition to the color error reduction (i.e., most of the colors pulled away from Blue), the White was also pulled away from Blue. That is to say, instead of staying at 8200^0 K, where people

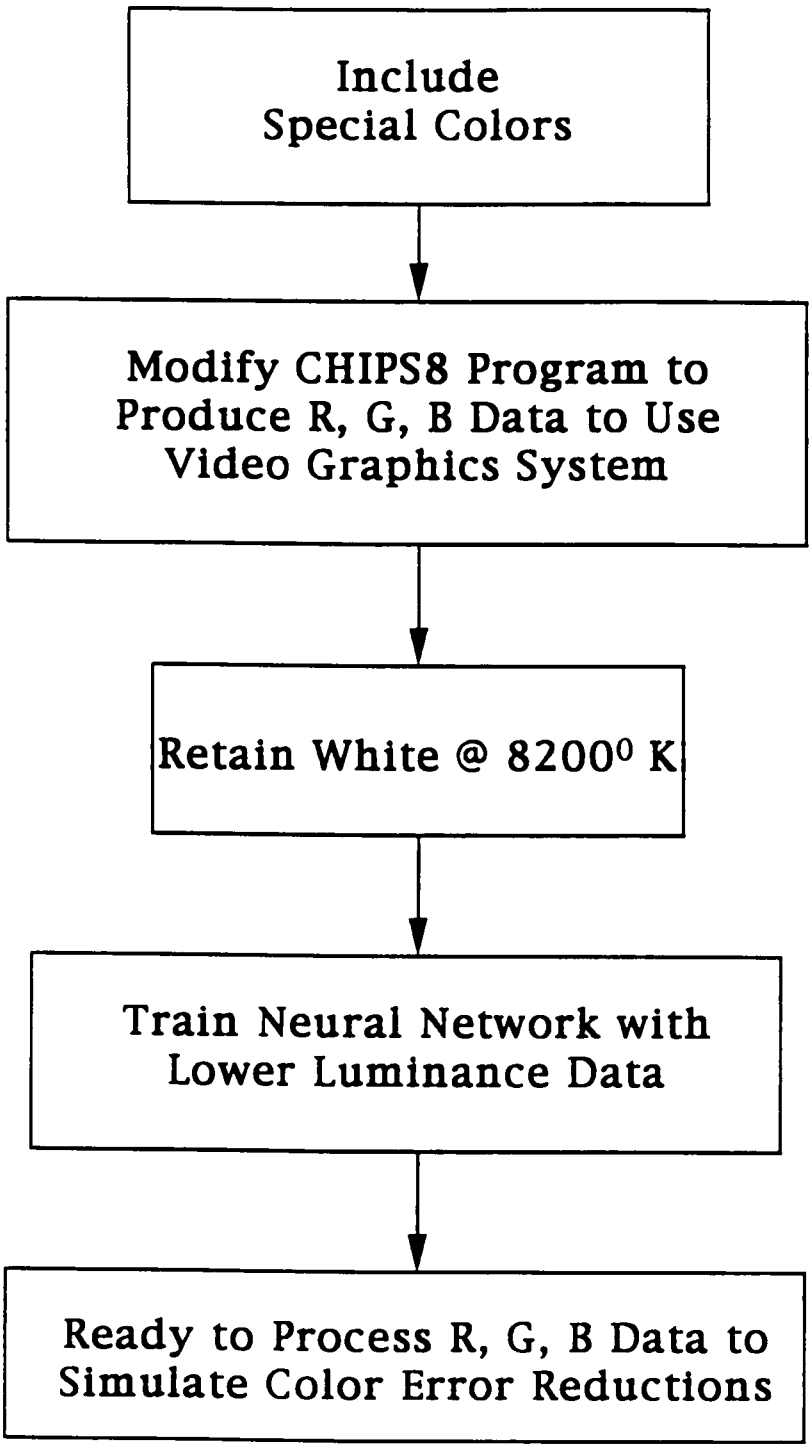


Figure 5.2 Block Diagram for Improvement of Software Simulation.

prefer, the White was shifted back towards 6500^0 K. That was really a big problem. After analysis, the reason for that problem was figured out. When writing the CHIPS8PC program, the phenomenon that colors were shifted towards Blue was considered as an error. It is true for most of the colors, but not for White! The shift of White towards Blue (from 6500^0 K to 8200^0 K) is what people want. So after the neural network was trained with the output data from CHIPS8PC program, there was a tendency for the network to pull most colors, including White, away from Blue. In order to retain White at 8200^0 K, the training data for the neural network must be modified. Some compromise has to be made. The modified data included the desired R, G, B data for White, which were adjusted to that of higher than 8200^0 K, and those for the colors with low saturation, i.e., close to White, which were adjusted to the same as the input data. That is to say, try to keep White at 8200^0 K and allow some errors for the colors close to White to exist. After these various modifications, the neural network was trained with the new data, and the result was obviously, but not totally, improved. The White was kept around 8000^0 K.

So far only the full luminance case was considered. The neural network was only trained with full luminance R, G, B data. What happens if low luminance data are encountered? Experimental run of the simulation programs showed that larger errors occurred. From the previous experience with neural networks, it was desired to train the network with lower luminance data. The CHIPS8PC program was

modified variously for producing data of different luminance levels, say, 20%, 40%, 50%, 60%, 80% and 100% of full luminance. Then, the neural network was trained properly (proper training presentation) with the new data, and gave better results for various luminance inputs.

A conclusion was drawn from the experience above. To have good results from neural networks, one must: a) Select good training data, b) Use proper training presentations.

After all the above modifications and improvement, the neural network was ready to process various R, G, B data, and the Color Test and Evaluation System was ready to operate. To view color image on a monitor for various conditions, two steps were performed, i.e., color bar pattern generations, and display of selected frames of video.

Color Bar Pattern Generations, Measurement and Evaluation

With the STAGE and OnSTAGE software, a color bar can be generated manually on the monitor screen. When a command file is created with the R, G, B data for White, Yellow, Cyan, Green, Magenta, Red, Blue, Black and Flesh indicated and position for display specified, a color bar pattern is generated. An example of the command file is shown in Appendix C.

From the NTSC standard, the R, G, B data for a color bar pattern should have the value as in Table 5.1.

With the value 1 in above table scaled to 65535, based on STAGE's R, G, B data range, a color bar pattern was displayed on the monitor. The x, y coordinates of these colors were measured with the chroma meter. The coordinates of the monitor primaries were determined as in Table 5.2.

Additionally, when different color bars of white for various luminances were created and the associated luminances measured, the gamma of the monitor was calculated as

$$\gamma = 2.34$$

In order to see the color errors and the effect of neural networks, two cases of color bar pattern comparison were performed.

1. Comparison of 6500^0 K White set-up and 8200^0 K White set-up color bar patterns, Case 1.
2. Comparison of 8200^0 K White set-up color bar patterns with and without neural network processing, Case 2.

For Case 1, the block diagram seen in Figure 5.3 indicates the procedure of the comparison. First of all, CHIPS7P2 was run with the measured primaries and gamma data of the monitor, and the PCEC (Philips Consumer Electronics Corporation) demodulator data, with

Table 5.1 R, G, B Data for NTSC Color Bars

	R	G	B
White	1	1	1
Yellow	1	1	0
Cyan	0	1	1
Green	0	1	0
Magenta	1	0	1
Red	1	0	0
Blue	0	0	1
Black	0	0	0

Table 5.2 Coordinates of Monitor Primaries

	x	y
Red	0.605	0.36
Green	0.313	0.581
Blue	0.144	0.075

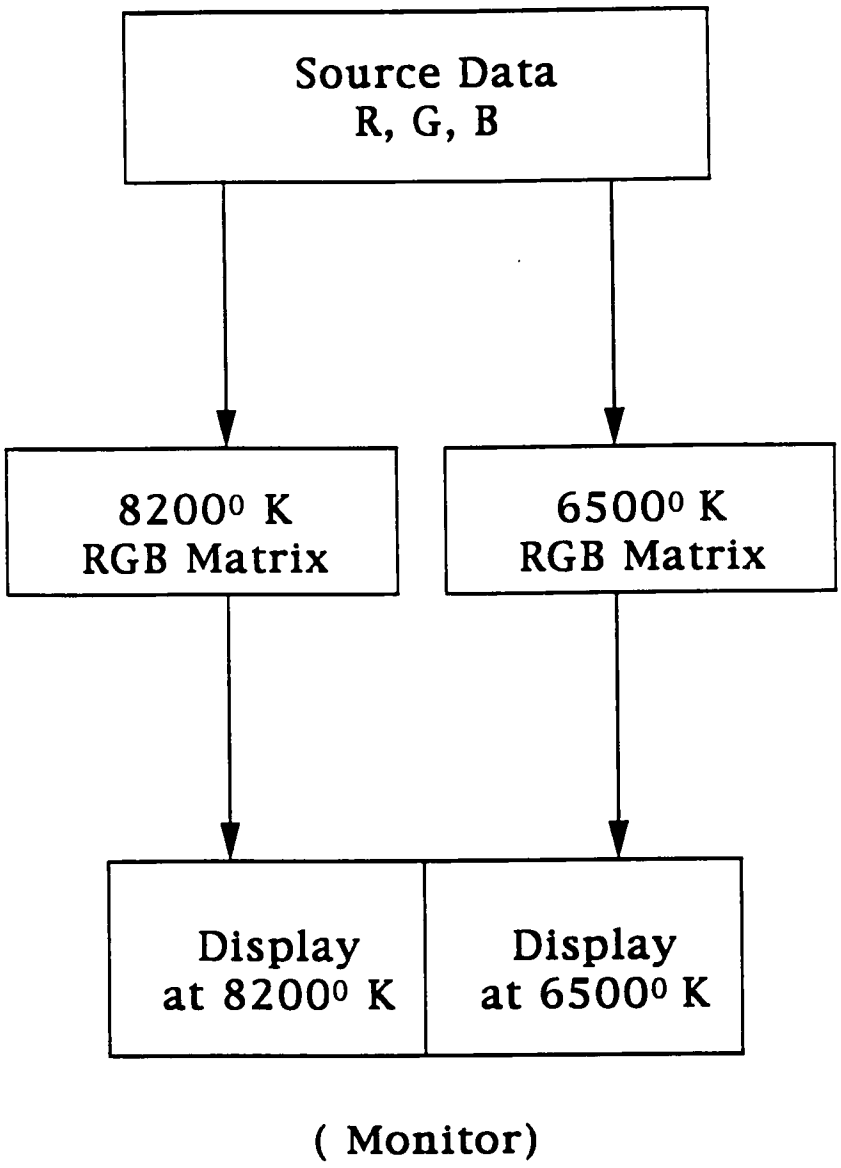


Figure 5.3 Block Diagram for 8200^0 K & 6500^0 K White Set-up Comparison Video Graphics Display.

display White set-up at 6500^0 K, and with chroma gain and phase adjusted for flesh tone error free. A tabular output R, G, B data for various colors were produced. Secondly, a RGB matrix for 6500^0 K White set-up was determined with the aid of the chroma meter measurement. Then, the output R, G, B data from the CHIPS program were processed with the RGB matrix, which produced a color bar pattern data. Finally, with these data, a color bar pattern with White set-up at 6500^0 K was displayed on the monitor. The command file in OnSTAGE language which generated this color bar pattern is seen in Appendix C.

Similarly, CHIPS7P2 program was run again for 8200^0 K White set-up, and a new RGB matrix was determined. Also, a color bar pattern with White set-up at 8200^0 K was displayed on the monitor.

With two color bar patterns displayed side by side, the color differences (errors) were easily seen by subjective observation. Also, a measurement with the chroma meter for the coordinates x, y and luminance Y of the color bars was made. With the measurement data, a CIE chromaticity diagram was made in Figure 5.4. This diagram looks like that from CHIPS7 program, seen in Figure 4.6, which verifies the computer simulation.

For Case 2, the comparison procedure is indicated in Figure 5.5. The color bar pattern for 8200^0 K White set-up without neural network processing is the same as that one in previous case. But the other one is a pattern which was created with the data processed with neural network. That is to say, the source R, G, B data were

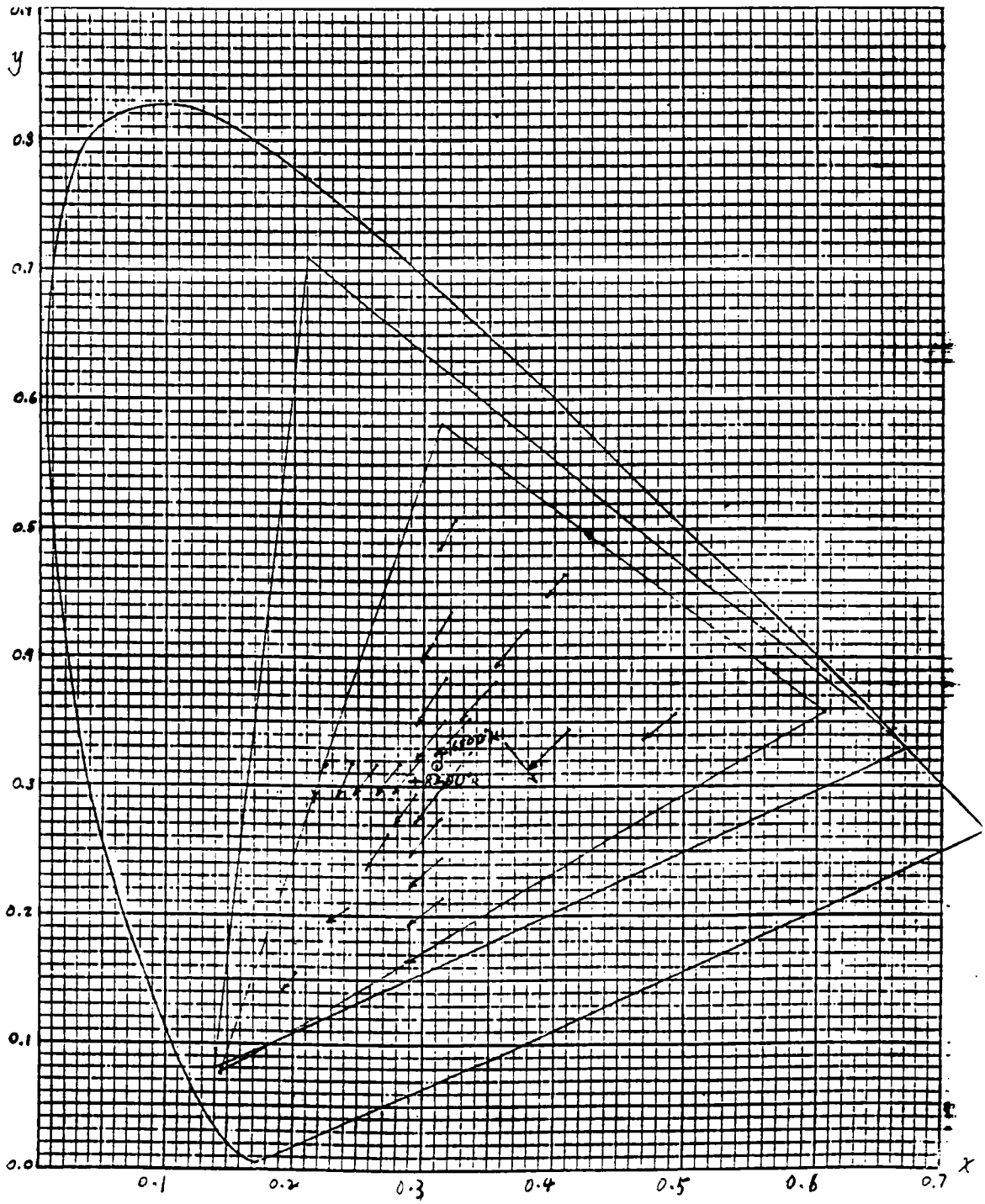


Figure 5.4 CIE Chromaticity Diagram with Color Measurement Data.

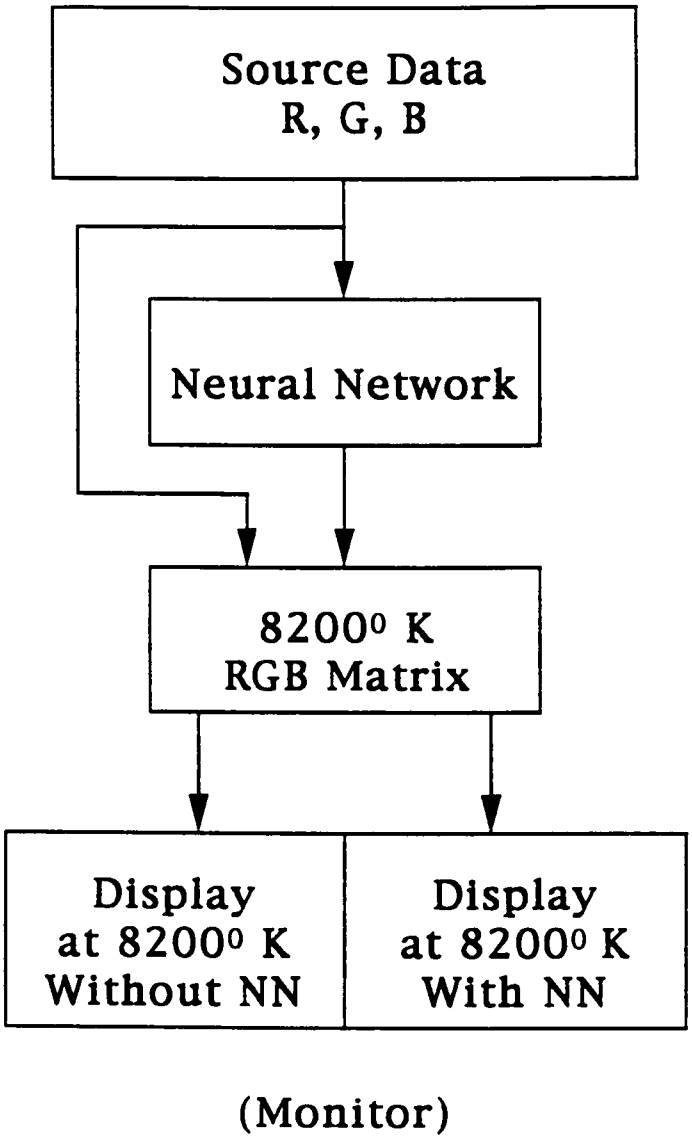


Figure 5.5 Block Diagram for 8200^0 K White Set-up with & without Neural Network Processing Comparison Video Graphics Display.

processed with the neural network first. The processed data went through the 8200^0 K RGB matrix before displayed on the monitor screen. Then these two color bar patterns could be compared side by side.

After the two cases were separately evaluated, a combination of comparison was made. The color bar patterns in two cases were displayed on a single screen. This is shown in Figure 5.6. The color bar pattern for 6500^0 K White set-up was displayed on the top. The pattern for 8200^0 K was on the center, with the other one with neural network processing on the bottom. When the three color bar patterns were compared carefully, one could see clearly how the color bars in the center pattern differed from those on the top when the White was set at 8200^0 K. The White shifted to bluish, but most of the other colors also shifted. That is the color error. When the bottom color bar pattern was examined, it was seen that the neural network really brought some color bars back towards the original ones, especially for Red, Blue and Cyan. This is the color error reduction. However, for certain colors, the luminance values were not corrected.

A picture was taken with a camera for these three color bar patterns. This picture is included in Figure 5.7. Although this picture can not represent the real one on the monitor very well, it can still be seen that how the color errors occur and how they are reduced.

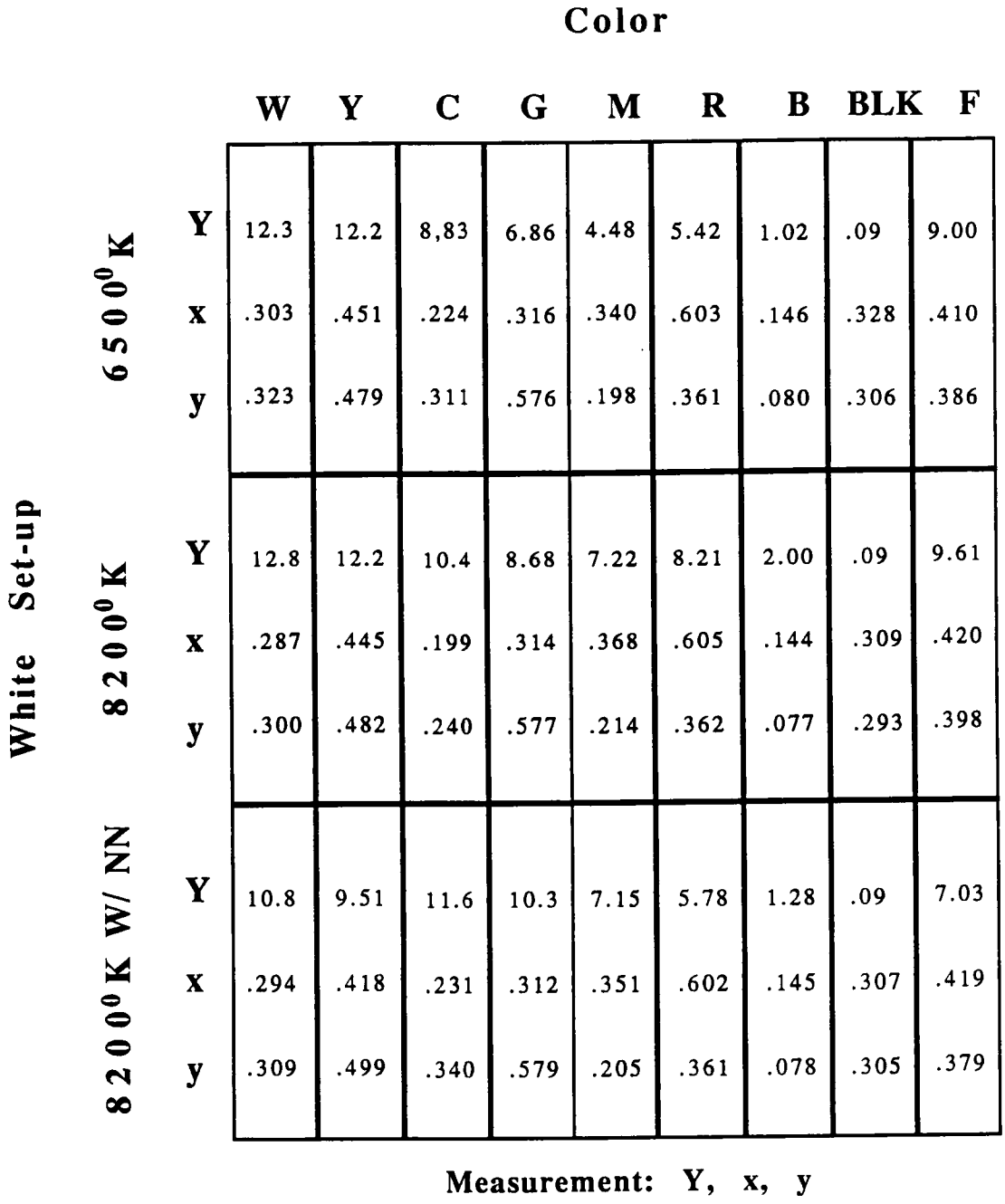


Figure 5.6 Color Bar Comparison.

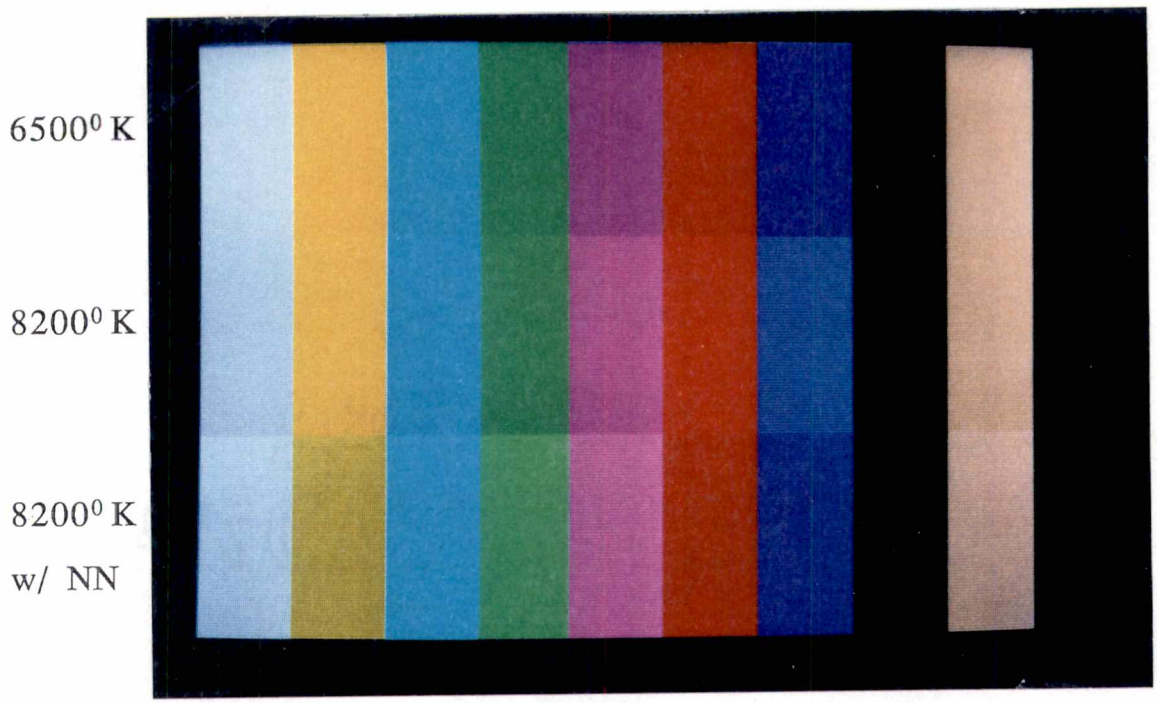


Figure 5.7 Color Bar Photograph.

Display of Frames of Video

After the color bar pattern generation and processing, it is desirable to deal with frames of video, which is more interesting to people. First of all, the frames of video should be able to display on the monitor directly. Secondly, the video source data should go through some procedure of processing before being displayed.

The first job was trivial. Some selected frames of video were obtained. With the aid of the OnSTAGE software, these frames can be displayed on the screen simultaneously at various positions with different sizes.

The second task was tedious. To see how the neural networks work, the source video data should be processed with the networks. However, the specific neural network software requires specific format for input data. Unfortunately, the video source data have their own format. Moreover, they are written in binary form. It is hard to read and process. A program called TRUEIN4.C in C++ was written for this purpose. This program reads in the video source data, selects the R, G, B data, converts them to the desired format, and writes them into an output file in ASCII form. Then these new data were processed with the neural network. Another program called TRUEOUT2.C was written to read in the processed ASCII data, convert them into the original source file format, and write into a binary file.

Finally, the processed video data were displayed on the monitor side by side with the original video scene. With a comparison of these two video displays, it could be seen that the neural network really changed colors. For example, the color close to Cyan in the original video scene was shifted towards Green, which is the case in the CIE diagram from the computer simulation. Also, White color was shifted towards bluish.

CHAPTER VI

CONCLUSIONS

Neural networks for minimization of color errors in color TV receivers were designed, built and tested. The results of the networks were examined with computer programs. The effect of color error reduction was shown with one of those computer programs. The CIE Chromaticity Diagram produced from that program clearly indicated the reduction of color errors. The color error reduction was also verified with a color test and evaluation system, which displayed color bars and selected frames of video with and without neural network processing for the purpose of comparison. A conclusion is drawn that neural networks see promising application on minimization of color errors in TV receivers.

However, the neural network solution is not perfect. Some remaining problems include: a) Some color errors are over-corrected. b) The luminance of some colors are not processed properly. c) To retain White around 8200^0 K, some color errors have to be allowed.

In fact, the first two problems are not a matter of the neural network itself. It is a matter of the training data. The future work includes further improvement of the software simulation by adjusting the training data.

Additionally, higher-order neural networks can be considered. A study made by Max B. Reid [22] demonstrated that a higher-order

neural network is vastly superior to a two layer, first-order network trained by backpropagation in terms of learning speed and recognition accuracy. The main principle of higher-order neural network is that the weighted sum includes cross product of inputs, in addition to individual input terms. The output of nodes in a general higher order network is given by:

$$y_i = f(\sum_j w_{ij}x_j + \sum_j \sum_k w_{ijk}x_j x_k + \sum_j \sum_k \sum_i w_{ijkl}x_j x_k x_l + \ldots) \qquad (6\text{-}1)$$

As this project progressed, a totally different idea was conceived. Since the color errors occur mainly because of the White set-up at 8200^0 K, an algorithm, called Algorithm B, can be set-up as following:

Test color signals for White or near White:
If true, apply 8200^0 K White set-up;
Otherwise, leave the data alone.

In this way, when the color are not White or near White, there exists no color error at all. This idea was verified with the color test and evaluation system. The algorithm was applied to the color bars and selected video scene data. A program, called MODWHT2.C, was written for this purpose. The monitor display proves that Algorithm B (Case 3) works. Only white color shifts towards bluish, and all the other colors remain unchanged. It appears as a simpler solution.

There is also the problem, wherein, one can do nothing about the color errors due to the non-NTSC display primaries.

A side by side comparison of four scene displays on the monitor was made, which is represented in Figure 6.1. From this comparison, we can see how color errors occur, how the neural network and Algorithm B work differently.

Both neural network and Algorithm B deserve further study. Hardware of either one may be implemented. And finally, experiment on a TV receiver may be conducted.

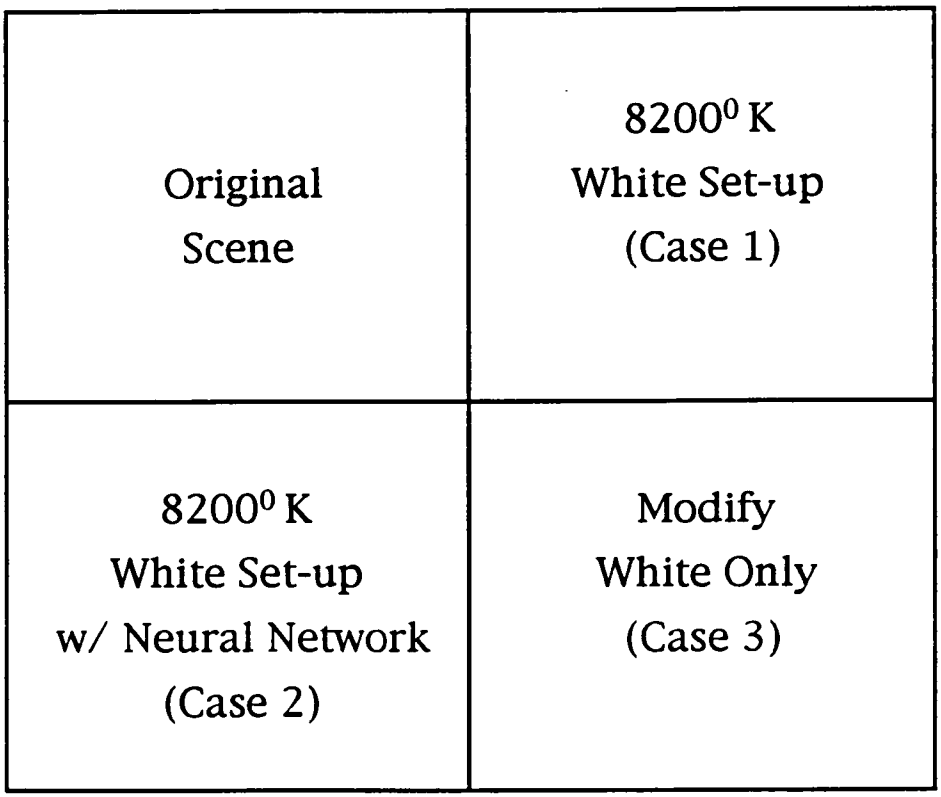


Figure 6.1 Four Scene Displays on Monitor

LIST OF REFERENCES

LIST OF REFERENCES

[1] John T. Cahill, Robert L. Werner, Petition of Radio Corporation of America and National Broadcasting Company, Inc. for Approval of Color Standards for the RCA Color Television System, June 25, 1953.

[2] Philips Laboratories, North American Philips corporation, "Documentation for the CHIPS7 program", Proprietary Information, 1991.

[3] C. Bailey Neal, "Television Colorimetry for Receiver Engineers", IEEE Transactions of BTR, Vol. BTR-19, No. 3, August, 1973, pp. 149-162.

[4] N. W. Parker, "An Analysis of the Necessary Decoder Corrections for Color Receiver Operation with Non-standard Receiver Primaries", Transactions of BTR, April, 1966, pp. 23-32.

[5] C. Bailey Neal, "Computing Colorimetric Errors of a Color Television Display System", NEC, December, 1974, pp. 1-10.

[6] Maureen Caudill, "Neural Networks PRIMER Part I", AI Expert, December 1987, pp. 46-52.

[7] F. J. Bingley, "Colorimetry in Color Television", Proceedings of the I-R-E, Vol. 41, No. 7, July, 1953, pp. 838-851.

[8] F. J. Bingley, "Colorimetry in Color Television--Part II", Proceedings of the I-R-E, Vol. 42, No. 1, January, 1954, pp. 48-51.

[9] F. J. Bingley, "Colorimetry in Color Television--Part III", Proceedings of the I-R-E, Vol. 42, No. 1, January 1954, pp. 51-57.

[10] D. H. Pritchard, "US Color Television Fundamentals - A Review", IEEE Transactions on Consumer Electronics, Vol. CE-23, No.4, November 1977, pp.467-478.

[11] F. J. Bingley, "Transfer Characteristics in NTSC Color Television", Proceedings of the I-R-E, Vol. 42, No. 1, January, 1954, pp. 71-78.

[12] Clyde N. Herrick, Color Television: Theory and Servicing, Reston Publishing Company, 1973.

[13] NeuroDynamX, Inc., DynaMind User's Guide, 1991.

[14] Richard P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April, 1987, pp. 4-22.

[15] Neural Ware Inc., NeuralWorks, 1988.

[16] Maureen Caudill, Charles Butler, Naturally Intelligent Systems, A Bradford Book, The MIT Press, 1990.

[17] California Scientific Software, BrainMaker, User's Guide and Reference Manual, 6th Edition, April, 1992.

[18] HNC Inc., ExploreNet, 1990.

[19] R. Hecht-Nielsen, "Theory of the Back propagation Neural Network," Intl. Joint Conf. On Neural Networks (Washington), Vol. I, pp. 593-605, 1989.

[20] Truevision, Inc., ATVista User's Guide, Release 1.0, October, 1988.

[21] Truevision, Inc., OnSTAGE Help File, July, 1989.

[22] Max B. Reid, Lilly Spirkovska, and Eilen Ochoa, "Rapid Training of Higher-order Neural Networks for Invariant Pattern Recognition", proc. International Joing Conference on Neural Networks, pp. 689-692, 1989.

APPENDIXES

APPENDIX A. LIST OF PROGRAMS

1. RGBERR1.BAS
2. B65E.ONS
3. B82EC.ONS
4. B82N5Y.ONS
5. TRUEIN4.C
6. TRUEOUT2.C
7. MODWHT2.C
8. CHIPS7P2.BAS (Proprietary)
9. CHIPS8PC.BAS (Proprietary)
10. CHIP10PC.BAS (Proprietary)
11. HUE2SAPC.BAS (Proprietary)
12. RGB5F.NND (Neural Network Design. Binary File)

APPENDIX B. RGBERR1.BAS PROGRAM

```
' RGBERR1.BAS WRITTEN IN QUICKBASIC
' THIS PROGRAM READS .NNI AND .NNO FILES AND
'      CALCULATES THE ERROR BETWEEN THE DESIRED RGB
'      VALUES AND THE NEURAL NETWORK OUTPUT VALUES.
' PROGRAMMER: SHAOFAN XU
'           ELECTRICAL & COMPUTER ENGINEERING DEPT.
'           UNIVERSITY OF TENNESSEE
'           KNOXVILLE, TN 37996
' FIRST WRITTEN: MAY 1992
' FINAL VERSION: JULY 1992

DIM I(800, 3), D(800, 3), R(800, 3), I$(8), D$(8), R$(8)
FE$ = "EOF"
PRINT "RGB ERROR CALCULATION PROGRAM"
PRINT "ENTER NAME OF .NNI FILE:"
INPUT FI$
PRINT "ENTER NAME OF .NNO FILE:"
INPUT FO$
PRINT "ENTER NUMBER OF DATA:"
INPUT K

OPEN FI$ + ".NNI" FOR INPUT AS #1
OPEN FO$ + ".NNO" FOR INPUT AS #2

' READ DISPLAY DATA
FOR I = 1 TO K
100  I$(1) = INPUT$(1, #1)
  IF I$(1) = "i" THEN
    INPUT #1, I(I, 1), I(I, 2), I(I, 3)
    D$(1) = INPUT$(1, #1)
    INPUT #1, D(I, 1), D(I, 2), D(I, 3)
   ELSE
    GOTO 100
  END IF
200  R$(1) = INPUT$(1, #2)
```

```
  IF R$(1) <> "r" THEN
  LINE INPUT #2, XX$
  GOTO 200
  ELSEIF R$(1) = "r" THEN
  INPUT #2, R(I, 1), R(I, 2), R(I, 3)
  END IF

' CALCULATE ERRORS
  RER = D(I, 1) - R(I, 1)
  GER = D(I, 2) - R(I, 2)
  BER = D(I, 3) - R(I, 3)
  REI = D(I, 1) - I(I, 1)
  GEI = D(I, 2) - I(I, 2)
  BEI = D(I, 3) - I(I, 3)
NEXT I
300 CLOSE #1
 CLOSE #2
 IGOOD = 0
 RGOOD = 0
 FOR I = 1 TO K
  FOR J = 1 TO 3
   IERR = ABS(I(I, J) - D(I, J))
   RERR = ABS(R(I, J) - D(I, J))
   CMP = RERR - IERR
   IF CMP > 0 THEN
    IGOOD = IGOOD + 1
   ELSE
    RGOOD = RGOOD + 1
   END IF
  NEXT J
 NEXT I
 PRINT "IGOOD= ", IGOOD
 PRINT "RGOOD= ", RGOOD
 PRINT "PER= ", RGOOD / (IGOOD + RGOOD)
 END
```

APPENDIX C. B65E.ONS PROGRAM

(Color bar generation with 6500^0 K White set-up)

```
setrgbb 42927 46583 46112
eraserect 0 0 80 140
setrgbb 52972 45186 2721
eraserect 80 0 160 140
setrgbb 0 45511 45236
eraserect 160 0 240 140
setrgbb 0 44393 1798
eraserect 240 0 320 140
setrgbb 48937 2236 44314
eraserect 320 0 400 140
setrgbb 59325 1071 876
eraserect 400 0 480 140
setrgbb 0 1118 43484
eraserect 480 0 560 140
setrgbb 0 0 0
eraserect 560 0 640 140
setrgbb 48121 37126 29143
eraserect 640 0 720 140
```

APPENDIX D. B82EC.ONS PROGRAM

(Color bar generation with 8200^0 K White set-up)

```
setrgbb 42476 46370 49544
eraserect 0 140 80 280
setrgbb 50929 46787 0
eraserect 80 140 160 280
setrgbb 0 45628 57471
eraserect 160 140 240 280
setrgbb 0 46370 4063
eraserect 240 140 320 280
setrgbb 56408 46 45481
eraserect 320 140 400 280
setrgbb 65201 742 0
eraserect 400 140 480 280
setrgbb 0 0 53359
eraserect 480 140 560 280
setrgbb 0 0 0
eraserect 560 140 640 280
setrgbb 48677 37235 28339
eraserect 640 140 720 280
```

APPENDIX E. B82N5Y.ONS PROGRAM

(Color bar generation with neural network output data)

```
setrgbb 45828 50535 52805
eraserect 0 280 80 420
setrgbb 44540 48521 0
eraserect 80 280 160 420
setrgbb 0 57164 49413
eraserect 160 280 240 420
setrgbb 0 55143 0
eraserect 240 280 320 420
setrgbb 61207 0 51681
eraserect 320 280 400 420
setrgbb 61068 0 0
eraserect 400 280 480 420
setrgbb 0 1819 47901
eraserect 480 280 560 420
setrgbb 0 0 0
eraserect 560 280 640 420
setrgbb 49772 38868 30674
eraserect 640 280 720 420
```

APPENDIX F. TRUEIN4.C PROGRAM

```
/* truein4.c written in Borland C++ */
/* Read a binary image file in truevision TGA and write to
      ASCII file (*.nni) in format of R, G, B data. */
/* Programmer: Shaofan Xu */
/*          Electrical & Computer Engineering Dept. */
/*          University of Tennessee */
/*          Knoxville, TN 37996 */
/* First written: August 1992 */
/* Final version: October 1992 */

#include <stdio.h>
int main()
{
   FILE *ifile, *ofile;
   unsigned red[1], blue[1] ,green[1], alpha[1];
   char dont[13], c, d, sfile[20], dfile[20];
   int i,j;
   short width[1], height[1], pixeldep[1];
   float red0, blue0, green0;

   pixeldep[0] = 0;

  /* Supply file names */
11: printf("Enter source file name:\n");
   gets(sfile);
   if ((ifile=fopen(sfile, "rb")) == NULL)
    { printf("Cannot find %s.\n", sfile); goto 11; }
   printf("Enter destination file name:\n");
   gets (dfile);
   ofile=fopen(dfile, "w");

  /* Read header data */
   fread(dont, sizeof(char), 12,ifile);
   fread(width, sizeof(short), 1, ifile);
   fread(height, sizeof(short), 1, ifile);
```

```
  fread(pixeldep, sizeof(char), 1, ifile);
  fread(dont, sizeof(char), 1, ifile);
  printf ("width = %d  Height = %d  pixeldep = %d\n",
        width[0], height[0], pixeldep[0]);
  printf ("x,y image sizes are = %d %d\n", width[0],
        height[0]);

 /* Read R, G, B, Alpha data */
  for (i=0; i<=height[0]; i++)
   {
    for (j=0; j<=width[0]; j++)
     {
      red[0]=green[0]=blue[0]=0.0;
      fread (blue, sizeof (char), 1, ifile);
      fread (green, sizeof (char), 1, ifile);
      fread (red, sizeof (char), 1, ifile);
      fread (alpha, sizeof (char), 1, ifile);
      red0=red[0]/256.0; green0=green[0]/256.0;
           blue0=blue[0]/256.0;
      fprintf (ofile, "i %f,%f,%f\n",  red0, green0, blue0);
     }
   }

  fclose(ifile);
  fclose(ofile);
  return 0;
}
```

APPENDIX G. TRUEOUT2.C PROGRAM

```
/* trueout2.c written in Borland C++ */
/* Read an ASCII file (*.nno) and write to binary image
      file in truevision TGA. */
/* 8200 degree K White set-up */
/* Programmer: Shaofan Xu */
/*          Electrical & Computer Engineering Dept. */
/*          University of Tennessee */
/*          Knoxville, TN 37996 */
/* First written: August 1992 */
/* Final version: October 1992 */

#include <stdio.h>
int main()
{
   FILE *ifile1, *ifile2, *ofile;
   unsigned red[1], blue[1] ,green[1], alpha[1];
   char rgb[4], dont[13], ii, pixeldep[1], sfile[20],
      nfile[20], dfile[20], *line;
   int i,j;
   short width[1], height[1];
   float red0[1], blue0[1], green0[1];
   unsigned char x[4];

   pixeldep[0] = 0;

  /* Supply file mames */
  11: printf("Enter source file name:\n");
   gets(sfile);
   if ((ifile1=fopen(sfile, "rb")) == NULL)
    { printf("Cannot find %s.\n", sfile); goto 11; }
  12: printf("Enter .nno file name:\n");
   gets(nfile);
   if ((ifile2=fopen(nfile, "r")) == NULL)
    { printf("Cannot find %s.\n", nfile); goto 12; }
   printf("Enter destination file name:\n");
   gets(dfile);
```

```
 ofile=fopen(dfile, "wb");

/* Read and write header data */
 fread(dont, sizeof(char), 12,ifile1);
 fread(width, sizeof(short), 1, ifile1);
 fread(height, sizeof(short), 1, ifile1);
 fread(pixeldep, sizeof(char), 1, ifile1);
 fread(dont, sizeof(char), 1, ifile1);
 printf ("width = %d  Height = %d  pixeldep = %d\n",
     width[0], height[0], pixeldep[0]);
 printf ("x,y image sizes are = %d %d\n", width[0],
     height[0]);

 fwrite (dont, sizeof(char), 12, ofile);
 fwrite (width, sizeof(short), 1, ofile);
 fwrite (height, sizeof(short), 1, ofile);
 fwrite (pixeldep, sizeof(char), 1, ofile);
 fwrite (dont, sizeof(char), 1, ofile);

/* Take off 3 header lines from .nno file */
 fgets(line, 82, ifile2);
 fgets(line, 82, ifile2);
 fgets(line, 82, ifile2);

/* Read R, G, B data and write R, G, B and Alpha */
 for (i=0; i<=height[0]; i++)
 {
  for (j=0; j<=width[0]; j++)
   {
     red0[0]=green0[0]=blue0[0]=0.0;
     fscanf (ifile2, " %c %f,%f,%f", &ii,&red0[0],
           &green0[0],&blue0[0]);
     alpha[0]=0;
     x[0]=blue0[0]*256.0*1.068; x[1]=green0[0]*256.0;
     x[2]=red0[0]*256.0*.916; x[3]=alpha[0];
     fwrite (x, sizeof(unsigned char), 4, ofile);
   }
  }

 fclose(ifile1);
```

```
    fclose(ifile2);
    fclose(ofile);
    return 0;
}
```

APPENDIX H. MODWHT2.C PROGRAM

```
/* modwht2.c written in Borland C++ */
/* Modify the R, G, B data for White to 8200 degree K in
      truevision TGA file */
/* Programmer: Shaofan Xu */
/*          Electrical & Computer Engineering Dept. */
/*          University of Tennessee */
/*          Knoxville, TN 37996 */
/* First written: September 1992 */
/* Final version: October 1992 */

#include <stdio.h>
int main()
{
   FILE *ifile, *ofile;
   unsigned red[1], blue[1] ,green[1], alpha[1];
   char dont[13], pixeldep[1], sfile[20], dfile[20], c;
   int i,j;
   short width[1], height[1];
   unsigned char x[4];
   float wc, kr, kg, kb;

   pixeldep[0] = 0;

   /* Supply file names */
 11: printf("Enter source file name:\n");
   gets (sfile);
   if ((ifile=fopen(sfile, "rb")) == NULL)
    { printf("Cannot find %s.\n", sfile); goto 11; }
   printf("Enter destination file name:\n");
   gets (dfile);
   ofile=fopen(dfile, "wb");
   printf("Etern White test factor:(5-10)\n");
   scanf("%f", &wc);
   printf("Enter kr,kg,kb:\n");
   scanf("%f,%f,%f", &kr,&kg,&kb);
```

```
/* Read and write header data */
fread(dont, sizeof(char), 12,ifile);
fread(width, sizeof(short), 1, ifile);
fread(height, sizeof(short), 1, ifile);
fread(pixeldep, sizeof(char), 1, ifile);
fread(dont, sizeof(char), 1, ifile);
printf ("width = %d  Height = %d  pixeldep = %d\n",
    width[0], height[0], pixeldep[0]);
printf ("x,y image sizes are = %d %d\n", width[0],
    height[0]);

fwrite (dont, sizeof(char), 12, ofile);
fwrite (width, sizeof(short), 1, ofile);
fwrite (height, sizeof(short), 1, ofile);
fwrite (pixeldep, sizeof(char), 1, ofile);
fwrite (dont, sizeof(char), 1, ofile);

/* Read and modify pixel data */
for (i=0; i<=height[0]; i++)
 {
  for (j=0; j<=width[0]; j++)
   {
     red[0]=green[0]=blue[0]=0;
     fread (blue, sizeof(char), 1, ifile);
     fread (green, sizeof(char), 1, ifile);
     fread (red, sizeof(char), 1, ifile);
     fread (alpha, sizeof(char), 1, ifile);
     if (red[0] != 0 && green[0]!=0 && blue[0]!=0)
      if ((red[0]/green[0]) > (wc/10) &&
            (red[0]/green[0]) < (10/wc))
       if ((red[0]/blue[0]) > (wc/10) &&
            (red[0]/blue[0]) < (10/wc))
        { red[0]=red[0]*kr; green[0]=green[0]*kg;
            blue[0]=blue[0]*kb;}

     fwrite (blue, sizeof(char), 1, ofile);
     fwrite (green, sizeof(char), 1, ofile);
     fwrite (red, sizeof(char), 1, ofile);
     fwrite (alpha, sizeof(char), 1, ofile);
    }
```

```
    }

  fclose(ifile);
  fclose(ofile);
  return 0;
}
```

VITA

Shaofan Xu was born in Nanning, Guangxi, P.R. China. He Graduated from China University of Electronic Science and Technology in Chengdu with a Bachelor's degree in Physics in July 1982, when he received a National Outstanding Student Award. The following month he entered East China Institute of Technology in Nanjing and graduated with a Master's degree in Computer Science in July 1985. From August 1985 to May 1988, he worked as a Research Member in North Institute for Scientific and Technical Information in Beijing. He received a National Excellent Project Prize for database development, and authored ten publications during that period.

In May 1988, the author enter the University of Tennessee, Knoxville (UTK) as a Visiting Scholar. He began working towards his second M.S. degree in Electrical and Computer Engineering in January 1990. While pursuing his graduate studies, he worked as a Research Assistant and Teaching Assistant in Electrical and Computer Engineering Department at UTK, and worked for Philips Laboratories, North American Philips Corporation on a joint project for twelve months. He is a co-inventor for a pending patent during that project period. Upon completion of the Master's degree, the author would like to get some industrial experience.