RGB-YMCK Color Conversion by Application of the Neural Networks

Gabriel Marcu and Kansei Iwata
Graphica Computer Corporation, Tokyo, Japan

Abstract

Printing applications require to convert RGB displayable pictures into four printing process components: yellow, magenta, cyan and black. The printing process generates black color by two mechanisms: substractive mixture of YMC components and by K component. In almost all cases, the RGB picture on the display has different colors to the YMCK converted printed picture. The colors of the YMCK printed picture can be simulated on the RGB display. Simulation is a complex process, which depends on the printer (ink, paper, printing technique, dot gain, UCR or GCR corrections), monitor (RGB phosphor components, gamma correction, brightness and saturation adjustments) as well as observation conditions (illuminant, reflections).

In the paper, a neural network is proposed as an alternative solution for RGB-YMCK color conversion, in order to obtain closer color appearance between RGB image and the corresponding YMCK printed image. The YMCK data, as inputs, and the RGB data resulted from simulation of YMCK printed colors, as outputs, are used to learn the neural net-work in order to perform the global color conversion from RGB to YMCK. The general RGB simulation process of the printed YMCK colors is not bidirectional, so that, the network finds one possible transformation with a certain probability, strongly dependent on the learning data which determines the weights of the neural network.

1. About the RGB-YMCK color conversion

The majority printing applications use four components process, yellow, magenta, cyan and black. Consequently, the RGB applications that need to print the colors using four components requires to convert RGB displayable pictures into YMCK printable format.

In almost all cases, the RGB colors on the display has different appearance to the YMCK converted printed colors. One cause of different color appearance consists of different color rendition mechanisms in the two cases. In case of RGB displays, the color is the result of additive mixture of the R, G, B phosphor sources of the CRT. The color appearance is affected by spectral phosphor emission, environment condition, screen reflections, calibration of the monitor. In case of YMCK printers, the color is the result of substractive mixture of Y, M, C, K thin ink layers deposited on the white substrate, i.e. paper. The printed color appearance is affected by the ink and paper substrate colors, illumination condition, printing technology, chemistry of wet ink combination in some cases. Another cause of differences in color appearance consist of the difference between color gamut of the displayed colors and printed colors.

The YMCK printed colors can be simulated on the RGB display. Simulation is a complex process, which depends on the printer (ink, paper, printing technique, dot gain, UCR or GCR corrections), monitor (RGB phosphor components, gamma correction, brightness and saturation adjustments) as well as observation conditions (illuminant, reflections).

The transformation from YMCK components to RGB components which represent on the CRT the same color like on the paper is referred in this article as YMCK to RGB color conversion.

The conventional color simulation uses basically the Neugebauer equations, combined, if it is necessary, with the Yule-Nielsen corrections for the ink mask effect. This color simulation enables to find the R,G,B reflectances of the YMCK printed colors based in principal on the y, m, c, k amount of ink deposited on the paper substrate. The model is calibrated according to the r,g,b reflectances of all color combinations resulted from the four ymck components. Details about the Neugebauer model are available in [1,9].

The reverse color conversion, from three RGB color components to four YMCK components is, in principle, a non-determined problem since requires to transform the tri-dimensional color space to a four dimensional color space. The printing process generates black by two mechanisms: substractive mixture of YMC components and by K component. The balance between the two mechanism of black color printing reflects the indetermination of the RGB to YMCK color transformation, and request the additional parameter to be used. In practical conditions, the fourth component, K, is determined as a function of the Y,M,C component set. Usually, the function can include few parameters such as the maximum black ink limit, maximum total ink limit, grey component replacement factor (GCR), under color removal factor (UCR), under color addition factor (UCA). Some models[2,3] have been proposed for the RGB to YMCK color conversion. Recently, different computer applications are currently available for color conversion ,and simulation, as are compared in references[4,5].

In this paper, a neural network is proposed as an alternative solution for RGB to YMCK color conversion, in order to obtain closer color appearance between RGB image and the corresponding YMCK printed image and to include in a single model all the complex parameters concerning the color rendition by both displaying and printing.

In paragraph 2 it is introduced a network architecture for color simulation. The calibration of the network weights is proposed in paragraph 3. Results and comments are available in paragraph 4 and 5.

The general color conversion is described in figure 1.

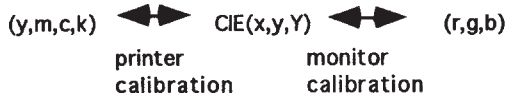


Figure 1. The diagram of RGB to YMCK color conversion

A (y,m,c,k) numerical set corresponds to a certain color, represented inside the CIE (x,y,Y) or in CIE (L*,a*,b*) color spaces by a point, if the printing conditions are decided. The point corresponds to a certain (r,g,b) numeric set if the displaying conditions are decided. Similar, the displaying and the printing conditions determine the reverse transformation from (r,g,b) to (y,m,c,k) numeric sets. One delicate problem of color conversion is to map the gamut of the displayable colors into the gamut of printable colors. Each (printer or display) device has its own color gamut and each color conversion must take it into account in order to be accurate. The color conversion presented in this paper avoids the problem of differences between displayable and printable gamuts, by a special procedure of data generation for neural network training

2. The artificial neural network for color conversion

The neural network solution for the RGB to YMCK color conversion is attractive from few points of view.

First, the direct and exact solution of the Neugebauer equations is difficult to be obtained. The nonlinearity of the transformation from YMCK densities to RGB reflectances[1] is difficult to be explicitelly reversed. One solution takes advantages of a color transformation table (LUT) which contains all the values of the YMCK to RGB color transformation derived using Neugebauer equations. This table can be easily reverted and the transformation from RGB to YMCK is obtained. The main disadvantage of this solution consists of the large transformation table required. For 8 bits/r,g,b color component, and four 8 bits y,m,c,k, output components, the table size is 2^{24} = 8M . 32bits = 32 Mbytes. The actual high speed memories are prohibited for such large LUTs, in common printing applications.

The neural network, by the ability of learning the logic functions that describes the input/output transformation, can solve more easily the problem of reversing the Neugebauer equations.

Second, the neural network includes globally all the parameters of the color conversion, and can mimic the be-havior of the complex model of color conversion. Examples of other color problems successfully solved using neural networks offer a stimulative start point in selection of this method.[6,7]

A neural network consists of a layer structure of inter-connected processing elements, referred as cells, figure 2.

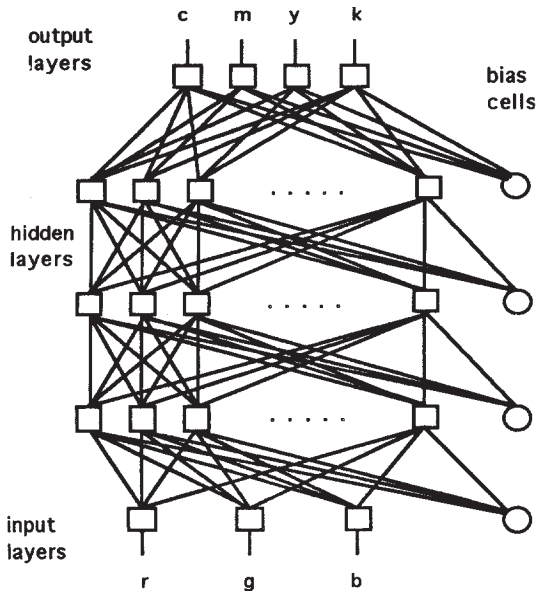


Figure 2. The structure of 3-12-12-12-4 neural network

Each cell computes a non-linear transfer function **f** of the sum of its weighted inputs from the previous layer, figure 3.

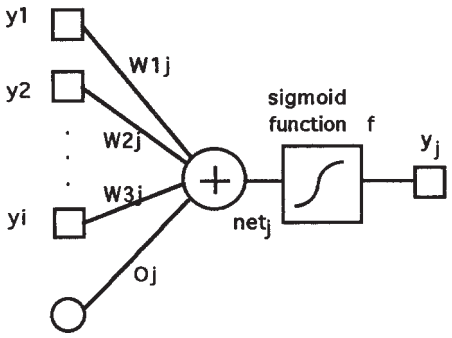


Figure 3. Computation performed in a single cell

The transfer function **f** acts as an activation function which is specifically to the network, rather to the application to be solved. The network weights adjustment is called learning process, and it is specific to the application to be solved. If the learning process uses an explicit set of input/output pairs the process is called supervised learning[8]. This type of learning process was selected for color conversion application.

For each input set, the output set is computed, successively from layer to layer, from input to the output, by the transfer function of each node.

The Back Propagation of the Generalized Delta Rule (GDL)[6,8] was used to compute the weight set of network. The GDL propagates the output error backward through the network from the output layers to the input layers, in

such way that the overall error of the net is minimized. The weights are updated using the following relationship:

$$\Delta \mathbf{W}_{ij} = \eta\,(t_j - o_j) i_i = \eta.\,\delta_j\, i_i,$$

where $\mathbf{I} = (i_1, i_2, ..., i_{1n[0]})$ and $\mathbf{O} = (o_1, o_2, ..., o_{1n[L]})$ are the input and output vectors, **T** is the target vector, **W** is the weight matrix. The amount η is referred as the learning rate, and is an important parameter of the learning process. The ln[p] represents the number of cells in layer p.

The linking weight from the i-th cell to the j-th cell is the generic element of the matrix **W**, and the $\Delta \mathbf{W}$ represents the matrix of changes that are to be made to minimize the error corresponding to the input output vectors **I** and **O**.

The δ_j quantity is referred as the error signal of the node j. For output layer the error signal is.

$$\delta_j = (t_j - o_j).\ \mathbf{f'}_j(net_j),$$

and for hidden layer:

$$\delta_j = \left(\sum_k (\delta_k . W_{kj}) \right) , \mathbf{f'}_j (net_j),$$

The function f'(netj) is the derivative of the nonlinear sigmoid function:

$$\mathbf{f}_j(net_j) = 1 / (1 + e^{-net_j}).$$

which maps the weighted and summarized inputs, described by net_j value, into an output value acting the j-th cell. The net_j value is derived as:

$$net_j = \sum_i (W_{ji} . o_i) + O_j .$$

The term O_j is named bias term and is considered as a weight linking a cell that is continuously on. The term O_j is learnt like other weights of the network.

Consequently, the error signal of the node j is for output layer:

$$\delta_j = (t_j - o_j) .\mathbf{f}_j (net_j).(1 - \mathbf{f}_j(net_j)) = (t_j - o_j) .y_j(1 - y_j).$$

and for the hidden layers:

$$\delta_j = \left(\sum_k (\delta_k . W_{kj}) \right) \mathbf{f}_j(net_j).(1 - \mathbf{f}_j(net_j)) = \left(\sum_k (\delta_k . W_{kj}) \right) . y_j.(1 - y_j),$$

The learning rate determines the stability of the neural network learning process. As the learning rate is large, the network learns faster but is susceptible to be oscilant and unstable. Increasing the learning rate based on the momentum term:

$$\Delta W_{ij} (n+1)= \eta(\delta_j . o_i) + \alpha . \Delta W_{ij} (n),$$

with α a constant, is one method of preserving the instability of the neural network. The learning algorithm of the neural network is briefly presented below:

1. select the learning rate, η, and constant α;
2. while the global network error, E, is greater than the threshold error limit, E_{lim},
   for each input i_i numeric set,
   compute the output set o_i ;
   for output cells, compute the differences :
   $\delta_j = (t_j - o_j).y_j.(1-y_j)$;
   for hidden cells, compute the differences :
   $\delta_j = \left(\sum_k (\delta_k . W_{kj}) \right) . y_j.(1-y_j)$;
   for all the weights, compute the differences :

   $\Delta W_{ij} = -\eta . \delta_j . y_i . y_j.(1 - y_j).$
   modify the weights with weight differences:
   $\Delta W_{ij} (n + 1)= \eta (\delta_j . o_i) + \alpha . \Delta W_{ij}(n);$
   $W_{ij} = W_{ij} + \Delta W_{ij};$

3. Calibration of the network learning process

The calibration of the network learning process consists of finding the suitable parameters of the network structure: number of layers, number of cells on each layer, learning rate.

The neural network proposed for this application consists of 3 cells input layer, 3 hidden layers with 12, 12 and 12 cells respectively, and 4 cells output layer. The network structure is presented in figure 2. The total number of weights is 412.

The YMCK data, as inputs, and the RGB data resulted from conversion of YMCK data, as outputs, are used to learn the neural network in order to perform the global color conversion from RGB to YMCK sets.

Three experiments was performed according to the data sets that are provided for the learning process of neural network. Each set of data strongly influences the neural learning process, and therefore, they are considered different experiments. In all experiments, the data sets are generated by the following procedure: a set of YMCK data is converted to RGB by the explicit Neugebauer equations (including correction for ink mask, monitor and printer calibration).[9] This special data generation procedure avoids the problem of difference between the displayable and printable color gamuts, since all RGB sets are the result of transformation of at least one YMCK set. Original input/output data are represented with 8 bits/color component. These data are normalized in order to be used for network learning process. Figure 4 shows the procedure of data generation.

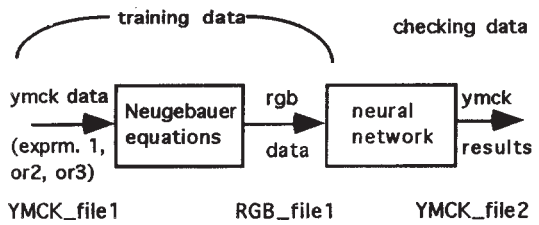


Figure 4. The procedure of generation of data used in neural network training

The RGB and YMCK numeric pairs are presented randomly at the input and output of the neural network,

and the mean squared error of the output layer is recorded. The back propagation learning procedure is used to adjust the weights of the network and the training process is stopped when the mean squared error decreases under a threshold limit or if the number of iterations (epoch) exceeds an upper limit.

First experiment uses the YMCK data uniformly generated, as for printing of the printer test chart. The total number of samples generated by this procedure is theoretically 4096Mega samples, considering a representation with 8 bits/each YMCK color component. The number of different resulted colors considerably smaller, 16Mega samples and corresponds to a representation with 8 bits/RGB color component. This set of data is selected for the learning process. The K component is generated independently of YMC components.

The second experiment uses the same procedure of generation of the YMCK data, but the K component is generated as a function of YMC component, using a GCR factor GCR=0.6.

The third experiment uses the YMCK Standard Color Image Data represented by 6 YMCK files, 2048 by 2560 pixels, summarizing about 31 Mega samples. The six files represent natural scene, but with histogram uniform distributed, representing almost equally all color combination. For these data, the K factor is not generated with an explicit function of YMC components, as it is illustrated in table I (pixels which differ only by K component), but a vague dependency of K from YMC components it can be observed (in column of the table with YMCK components).

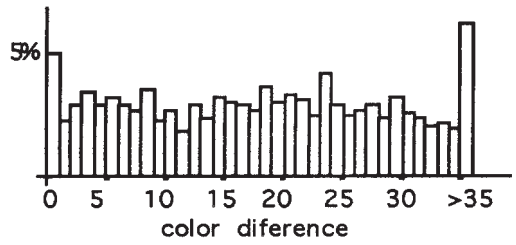
Table 1.
Some samples of YMCK sets from one file (portrait) of Standard Color Image Data. Column 1 shows the relative independency of K with respect to YMC components. All YMCK samples in the table corresponds by Neugebauer equations to the same color described by RGB components, illustrating the variable balance between K and YMC black printing.

Y	M	C	K	R	G	B
33	33	33	47,48,...,54			
34	34	34	39,40,...,46			
35	35	35	31,32,...,38			
36	**36**	**36**	24,25,...,30	215	215	215
37	37	37	16,17,...,23			
38	38	38	8,9,.....,15			
39	39	39	1,2,.....,7			

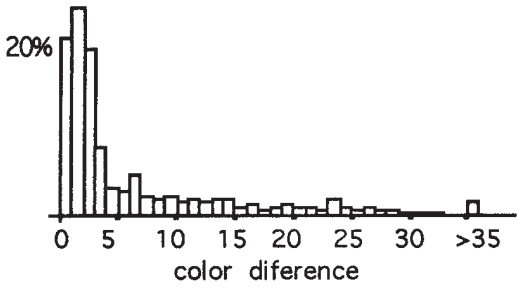
* bold values are selected by RGB to YMCK conversion network trained with second experiment data set.

The presentation to the network of all patterns selected for learning process is called an epoch. In practical condition, an epoch used 2Mega samples. The samples are uniformly selected from available data, for each one of the three experiments. During an epoch, the patterns are randomly selected for network learning process.

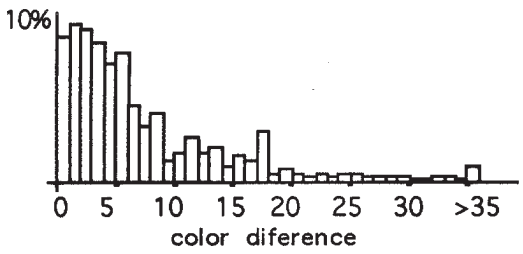
The general RGB- YMCK color conversion is not bidirectional conversion process, if the K component is not a function of the YMC components. Same (r,g,b) numeric set defining a precise color on the display, may be the results of YMCK to RGB transformation of many differ-ent (y,m,c,k) numeric sets, which corresponds to a single printed color. Table I offers an example of few YMCK numeric values that conduct to same RGB numeric set by the Neugebauer transformation. This phenomena occurs when a variable balance between the dual black printing mechanisms (by black ink or by overlapping the Y,M,C components) is used and it is more evident for dark colors, as effect of an increased value of K component. Therefore, in these cases, the network finds one possible transformation with a certain probability, strongly dependent on the learning data which determines the weights of the neural network. The learning process must be carefully designed, by selecting the input output pairs which corresponds as much as possible to a constant balance factor of black printing.



(5a) histogram of color differences for experiment 1, with K independent from YMC components.



(5b) histogram of color differences for experiment 2, with K a function of YMC components (GCR = 0.6).



(5c) histogram of color differences for experiment 3, with Standard Color Image Data (natural images, K dependent of YMC).

Figure 5. Histogram of color differences for 3 experiments of network training

4. Results

Testing for conversion of the neural network uses the same strategy as in [6],[7] which consists of learning the network using one part of the available data, and next, checking the correctness of the color conversion using the remain part of data.

The conversion accuracy is evaluated in few ways.

The first evaluation method of conversion accuracy is performed in the CIE L*a*b* color space. The numeric set is transformed to XYZ tristimulus set by the matrix transformation specific to the monitor parameters[10]. Then the XYZ tristimulus set is converted to CIE L*a*b* color set by the transformation which defines the CIE L*a*b* space[11]. The normalized histogram of color differences is presented in figure 5a,b,c for each experiment.

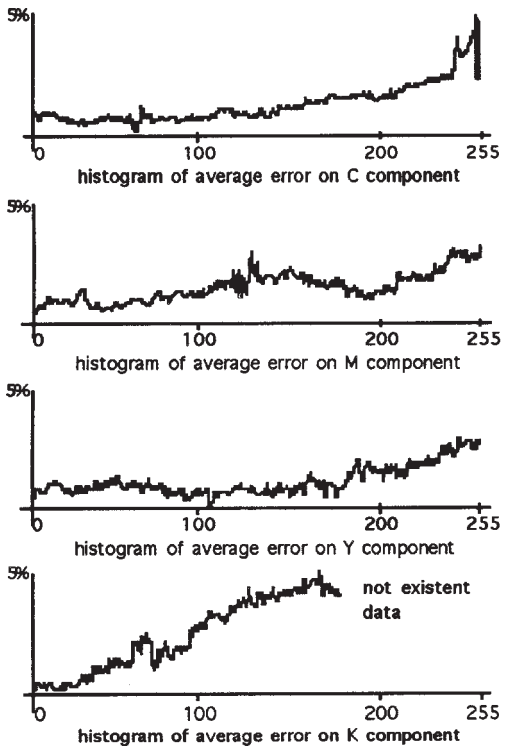


Figure 6. The error histogram on each YMCK component for the second experiment.

Another evaluation method conversion accuracy takes advantage of the explicit color conversion from YMCK to RGB, described by the Neugebauer equations. The coefficients of color transformation are calibrated for the RGB monitor Hitachi CM2185M and to the IRIS 3024 ink jet printer. The original YMCK data file, YMCK_file1, is converted to RGB by the Neugebauer equations and the resulted RGB data, stored in file RGB_file1, are converted back to YMCK by the neural network, conducting to the file YMCK_file2. Original and transformed YMCK files are compared and differences are evaluated. The histograms of average errors on each color component are presented in figure 6 for second experiment.

The errors between YMCK numeric files are sensible high but does not reflects the errors between printed colors corresponding to the these files. The YMCK to RGB color conversion can be performed once again for the network processed file, YMCK_file2, by Neugebauer equations conducting to a new set of RGB data, stored in RGB_file2, and the errors between RGB files can be evaluated. The errors in case of RGB files are sensible smaller than in case of YMCK files. This demonstrates that error in appearance of printed colors are smaller than the YMCK file errors. The YMCK file errors are sensible higher in the dark part of the picture. This fact illustrates the variable balance of black printing presented in the input YMCK data, which seems to be not learned by the network.

The data used to learn the network are provided by the YMCK to RGB conversion, and consequently corresponds to the printing color gamut. This avoids the critical problem of the mapping of the RGB displayable gamut to YMCK printable gamut. All the RGB colors provided by the YMCK to RGB are convertible back to YMCK without any problem of color gamut, since these (r,g,b) numeric sets corresponds to the YMCK printable colors.

A similar network structure can be used to perform the color conversion from YMCK to RGB color components. The data base remains unchanged and the method of testing the convergence of the learning process can be used in the same manner as for RGB to YMCK color conversion.

5. Conclusions

Few advantages of the neural network approach to the RGB to YMCK color conversion can be derived:

a. The neural network enables to obtain a convenient procedure of color conversion from RGB to YMCK color components, including all the parameters of color transformation which take into account the calibration of the monitor and printer.

b. The color conversion is fast, since no complex computations are required.

c. The network can continue to learn after the initial training process, and offers potential adaptation to more complex phenomena concerning the printing process (i.e. the effect of combination of wet ink layers, the effect of replacing the paper substrate with another substrate, such as textile material, etc.).

This paper represents an initial study that can be continued and developed in order to obtain suitable input/output sets and proper structure of the neural network that conducts to better results in color conversion. The derivation of the network structure concerning the number of the input, output and hidden layers is a subject of future development. The study offers an optimistic perspective to the use of the neural network for the color conversion problems.

References

1. G. G. Field, *Color and Its Reproduction*, Graphic Arts Technical Foundation, GATF, 1988.

2. B. J. Lindbloom, *Accurate Color Reproduction for Computer Graphics Applications*, Computer Graphics, V23,N3, July 1989.
3. K. D.Gennetten, RGB to CMYK conversion using 3-D baricentric interpolation, *Proceedings of IS&T/SPIE Conference* no.**1909** on Device-Independent Color Imaging and Imaging Systems Integration, San Jose, CA, Feb. 1993.
4. S. Roth, All about the Color, *Mac World*, January 1992.
5. S. Roth, Managing Color, *Mac World*, January 1993.
6. J. M. Bishop, M. J. Bushnell, S. Westland, Application of the Neural Networks to Computer Recipe Prediction, *Color Research and Applications*, V**16**, N1, February 1993.
7. S.Tominaga, Color Notation Conversion between Munsell and CIE color Systems Using Neural Networks, *Journal of Color Science Association of Japan*, V**17**, N1, 1993.
8. D. R. Hush, B. G. Horne, Progress in Supervised Neural Networks, *IEEE Signal Processing Magazine*, January 1993.
9. * * *, *Manual of Standard Color Image Data (SCID)*, Japan Association for Standardization, Image Processing Standardization Committee, 1990.
10. G. W. Meyer, D. P. Greenberg, *Perceptual Color Spaces for Computer Graphics*, in "Color and the Computer", edited by H. J. Durrett, Academic Press,1987.
11. G. Wyszecki, W. S.Stiles, *Color Science*, 2nd ed., Wiley, New York,1982.