Express Letters

A New Efficient Fuzzy Algorithm for Color Correction

Jer Min Jou, Shiann Rong Kuang, and Ren Der Chen

***Abstract*—Color correction, which nonlinearly converts the color coordinates of a scanner into that of a printer, is important and difficult for multimedia applications. An efficient tree-based fuzzy logic approach [1] has been proposed to solve the problem. However, the algorithm design and the implementation in [1] are slow, costly, and of poor color quality, due to the use of maximum criteria for defuzzification. In this paper, a new fuzzy tree inference algorithm for color correction is proposed. It is very simple and efficient and is suitable for adopting the center-average method for defuzzification to obtain better color quality. Therefore, a fast and cost-efficient implementation with good correction effects for tree-based fuzzy color correction is achieved.**

***Index Terms*—Color correction, defuzzification, fuzzy.**

I. Introduction

Because of the popularity of multimedia, more and more electronic documents contain color material. The subject raised by this wide us of color is how to reproduce the original color image faithfully between different color peripheral devices. The most common and important example is the color document transferred between a scanner and a printer. However, in this color reproducing example since the nonlinear color mapping problem exists between them, the coordinates obtained by the color scanner cannot be used directly by the color printer. Therefore, a mechanism to efficiently transform the scanner color coordinates into that of the printer is necessary and critical for a color document environment.

In the past, Takeuchi [2] proposed a direct conversion method and Hung [3] proposed a look-up table and interpolation method for color correction. However, the costs for the computation and storage are extremely high. On the other hand, Kang [4] and Chang [5] used the feed-forward neural network and multilayer back-propagation neural network to solve the problem, respectively. However, they have slow learning speed and large memory-required problems. Recently, an efficient fuzzy-tree approach was proposed by Liu [1]. Fig. 1 depicts the block diagram of the system. In it, the original color image is first scanned in by the scanner to generate the red, green, and blue (RGB) scanner color coordinates. Then, the RGB scanner colors are converted to the cyan, magenta, and yellow (CMY) printer color coordinates through the fuzzy-tree color correction system. The advantages of the fuzzy tree approach lie in its simplicity, adaptability, and good correction effect. However, the algorithm design and the implementation of it in [1] are slow, costly, and of poor color quality due to the use of maximum criteria for defuzzification.

In this paper, we propose a new fuzzy tree inference algorithm for color correction. It is simple, efficient, and suitable for using the center-average method for defuzzification, by which a better color correction effect is obtained, while keeping low computational complexity. Moreover, an efficient hardware architecture of the proposed algorithm is also presented, which uses less hardware due to the low complexity of the algorithm.

Manuscript received November 4, 1997; revised May 8, 1998. This work was supported in part by the National Science Council under Contract NSC-86-2221-E-006-022. This paper was recommended by Associate Editor J. Zurada.

The authors are with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C.

Publisher Item Identifier S 1057-7122(99)04745-5.

II. New Algorithm and Its Hardware Architecture

A. New Fuzzy Color Correction Algorithm

In [1], the color correction process is modeled as a three-level fuzzy tree (see Fig. 2) inference process. The function of each fuzzy subtree is to do one color conversion, which is performed by finding a decision path in it. At the beginning, the R-tree is employed to determine the mapping of the red color according to the matching degree of the input, which is processed with eight fuzzy sets $(s_1, s_2, \cdots, s_8)$ for that color [see Fig. 3(a)]. The eight fuzzy sets are used to represent the different intensities of each color. After determining the decision path of the red color, the subtrees and decision paths of green and blue colors can be found sequentially on the analogy of the red color inference process. The fuzzy tree inference process is simple, adaptive, and has good correction effects. However, the algorithm of it in [1] is inefficient and its hardware implementation is costly and slow. Additionally, in order to save the processing time and cost, the maximum criteria method is used for defuzzification in the implementation. It has been shown in [1] that the maximum criteria defuzzification method usually results in poorer color quality for the correction than the center-average defuzzification method.

To overcome these problems, a new efficient fuzzy tree inference algorithm suitable for the center-average defuzzification method is developed. By carefully inspecting the fuzzy tree inference approach in [1], we find the following facts for color correction: 1) the triangle membership functions of each fuzzy set shown in Fig. 3(a) are with the same shape; 2) the distance d between arbitrary two neighbor fuzzy sets of a subtree is fixed, thus $s^j = s^1 + (j-1)^* d$ for $j = 2, 3, \cdots, 8$ where s^j denotes the supported value of fuzzy set s_j; and 3) the overlap factor of the membership functions is two: that is, at most two membership functions are overlapped so that at most two matching degrees are nonzero, as shown in Fig. 3(a). With these facts we can design a new efficient fuzzy tree color correction algorithm as follows.

First, we find the minimal j that is used to indicate which fuzzy set the input color X_i belongs to so that

$$X_i - s^1 - j * d \le 0, \quad \text{for } 0 \le j \le 8. \tag{1}$$

Let k be the minimal j found and $D = X_i - s^1 - k^* d$ then $|D|$ represents the distance between X_i and the fuzzy set s_k [see Fig. 3(a)]. When k is found the fuzzy set to which X_i belongs and the nonzero matching degrees can be determined efficiently by the following process due to facts 2) and 3):

Case 1: $k = 0$: X_i belongs to fuzzy set s_1 and only the matching degree $\mu_1 = 1$.

Case 2: $1 \le k \le 7$. If $|D| > d/2, X_i$ belongs to fuzzy set s_k, otherwise, X_i belongs to fuzzy set s_{k+1}. Moreover, μ_k and μ_{k+1} are nonzero and $\mu_k + \mu_{k+1} = 1$.

Case 3: $k = 8$: X_i belongs to fuzzy set s_8 and only the matching degree $\mu_8 = 1$.

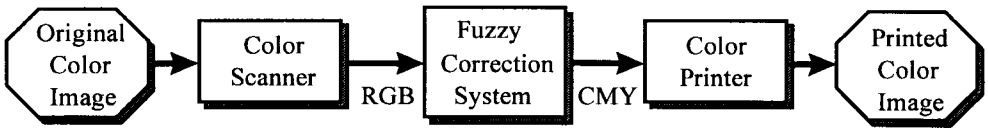

Fig. 1. Block diagram of fuzzy color reproduction system.

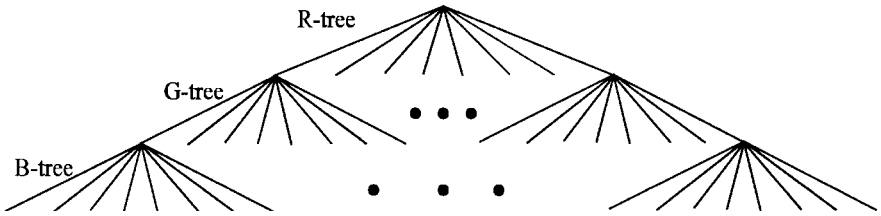

Fig. 2. The tree structure for fuzzy color correction.

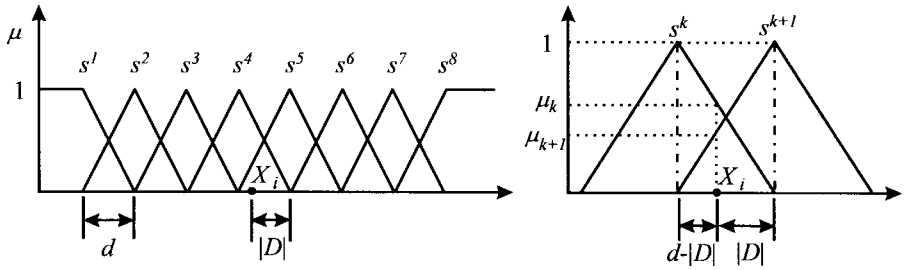

Fig. 3. (a) The set of membership functions. (b) The relationship between $\mu_k, \mu_{k+1}, |D|$, and $d - |D|$.

The decision path of the subtree is then found after the fuzzy set to which X_i belongs has been found.

Subsequently, if the center-average method, which has higher computational complexity than that of the maximum criteria method, is used the inference result is calculated as

$$X_o = \sum_{j=1}^{8} \mu_j \times w_j \Big/ \sum_{j=1}^{8} \mu_j \tag{2}$$

where $(w_1, w_2, \cdots, w_8)$ are the supported values of the fuzzy sets which represent the various intensities of color space CMY. The direct calculation of (2) is costly and slow. A new formula, based on the facts described above, is therefore derived as follows. After the k and $|D|$ are obtained, the (2) can be rewritten as

$$X_o = \begin{cases} w_1, & \text{if } k = 0 \\ \mu_k * w_k + \mu_{k+1} * w_{k+1}, & \text{if } 1 \le k \le 7 \\ w_8, & \text{if } k = 8. \end{cases} \tag{3}$$

By Fig. 3(b) and the analog of triangles of the fuzzy sets from fact 1) we get

$$\frac{|D|}{\mu_k} = \frac{d}{1} = \frac{d - |D|}{\mu_{k+1}}. \tag{4}$$

From (4), μ_k and μ_{k+1} in (3) can be obtained as

$$\mu_k = \frac{|D|}{d} \quad \text{and} \quad \mu_{k+1} = \frac{d - |D|}{d}. \tag{5}$$

Therefore, the computation of X_o now becomes

$$X_o = \begin{cases} w_1, & \text{if } k = 0 \\ \dfrac{|D| * w_k + (d - |D|) * w_{k+1}}{d}, & \text{if } 1 \le k \le 7 \\ w_8, & \text{if } k = 8. \end{cases} \tag{6}$$

Based on the above deduction, a C-like code of the new fuzzy tree color correction algorithm is given in Fig. 4. In it, L denotes the current level of the three-level fuzzy tree and $1 \le L \le 3$. If $L = 1$ then X_i is red, and if $L = 2$ then X_i is green. Otherwise, X_i is blue. In addition, $Path_L$ denotes the decision path of the current subtree and $0 \le Path_L \le 7$. Its complexity is analyzed in the next section.

```
1:    L=1;
2:    while (input pattern Xi ≠ NULL) {
3:          Calculate the address of rule memory (ROM);
4:          s^l=ROM[address++];   d=ROM[address];
5:          k=0;    Path_L=0;    D=Xi − s^l;
6:          while (k<8 && D>0) {
7:                D=D−d;   Path_L=k;   k++;
            }
8:          if(1≤k≤7 && |D|≤d/2) Path_L=k;
9:          Calculate Xo using Eq. (6);
10:         if(++L==4) L=1;
      }
```

Fig. 4. The modified fuzzy color correction algorithm.

B. Time Complexity Analysis

To do one color conversion, the proposed algorithm shown in Fig. 4 requires one addition operation, nine subtraction operations, two multiplication operations, one division operation, and nine comparison operations at most. The algorithm of [1] requires 16 subtraction operations, seven comparison operations, and one decoding operation fixedly. By using the 0.8-μm cell library [7], the 8-bit multiplier and divider require about triple computational time as compared with other 8-bit simple operations, such as addition, subtraction, or comparison. Thus, the algorithm of [1] needs 24 fixed units of computational time to do one conversion. On the other hand, the proposed algorithm needs four units of computational time in the best case and 28 units in the worst case to do one conversion because the defuzzification may, at times, not require any computation and the iteration number I of the inner loop in Fig. 4 is changed according

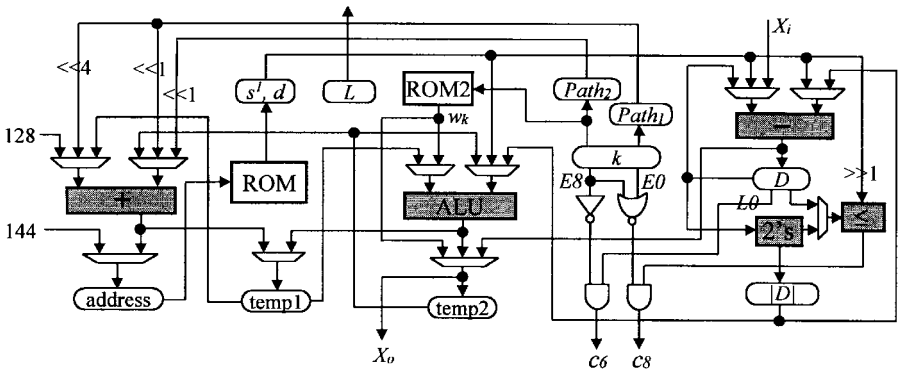

Fig. 5. The architecture of the proposed fuzzy color correction.

TABLE I
COMPARISON OF COMPUTATIONAL COMPLEXITY BETWEEN THE PROPOSED ALGORITHM AND THAT OF [1]

pictures	file size (bytes)	algorithm of [1]	proposed algorithm		speedup
		c-time	c-time	$\overline{I}$	
ex1	148416	3556800	2576595	4.25	1.38
ex2	230604	5068800	4168651	5.39	1.22
ex3	974916	23392800	16764288	3.35	1.40
ex4	1137198	27287568	19138743	4.11	1.43

to the input data $(0 \leq I \leq 8)$. If we allow I to be four fixedly, the average computational time of the proposed algorithm for one conversion is about 16 units, which is 1.5 times faster than that of [1].

Four pictures have been used to test the computational complexity of the proposed algorithm. The compared results with [1] are reported in Table I. In it, columns file size, c-time, $\overline{I}$, and speedup represents the sizes of these pictures, the total units of computational time, the average number of iterations of the inner loop in the proposed algorithm, and the speedup ratio of our algorithm comparing with that of [1], respectively. The results show that about 1.4 times speedup can be obtained.

C. Hardware Architecture

To design an efficient architecture of the proposed algorithm, the high-level synthesis tool [6] is applied. The resulting architecture is shown in Fig. 5. It only requires one comparator, one two's complement converter, one adder, one subtractor, and one arithmetic logic unit (ALU) which can perform multiplication and division operations. Additionally, a 146-byte ROM is used as the rule memory. The s^1 and d of the blue color subtrees are stored in the first 128 bytes of the ROM, and the s^1 and d of the green and red color subtrees are stored in the following locations. Therefore, the address for retrieving the s^1 and d from ROM can be calculated as

$$\text{address} = \begin{cases} 144, & \text{if } L = 1 \\ 128 + Path_{L-1} * 2, & \text{if } L = 2 \\ Path_{L-2} * 16 + Path_{L-1} * 2, & \text{otherwise.} \end{cases} \quad (7)$$

The storing order shown above significantly simplifies the hardware design and reduces the hardware cost and time for calculating the address, since the multiplication operation in it can be reduced by the shift-left operation.

When the input X_i is transmitted to the architecture, first it calculates the rule address according to the level number L and (7), and then it reads s^1 and d from the ROM. The notation $\ll n (\gg n)$ in it denotes that the value is shifted left (right) n bits. Next, the architecture finds out the values of k and $|D|$ to determine $Path_1$ and $Path_2$. The register used to store k is a 4-bit up-count counter. The condition $E8(E0)$ is one if $k = 8$ $(k = 0)$. Another condition $L0$ is one if $D > 0$. Finally, the center-average method is used for defuzzification and the inferential result X_o is obtained by (6). In Fig. 5, the $(w_1, w_2, \cdots, w_8)$ are stored in a 8-byte ROM (ROM2) and k is used as its address. The architecture in Fig. 5 has been simulated in Verilog HDL with an 0.8-μm CMOS cell library [7] within the Cadence System. The simulation results show that its working frequency can be 50 MHz, which is sufficiently fast for speed-critical applications.

On the other hand, the architecture in [1] requires at least one subtractor and one two's complement converter for fuzzification, and one comparator and one decoder for defuzzification. Comparing both architectures, the architecture in [1] is one adder and one ALU less, but one decoder more, and it uses a 2-kbyte ROM while our architecture only uses a 154-byte ROM (two ROM's). Moreover, a data manager, which is not required in our architecture, is used in [1] to control the desired data flow for processing the coordinates red, green, and blue. Therefore, the proposed architecture uses less hardware than that of [1].

III. CONCLUSIONS

In this paper, we have proposed a new efficient algorithm and its hardware architecture for fuzzy tree color correction. Experimental results show that not only is the correction effect better and the processing speed faster, but also the hardware cost is lower.

ACKNOWLEDGMENT

The authors would like to thank Dr. C. Y. Huang for his constructive discussions.

REFERENCES

[1] B. D. Liu and C. Y. Huang, "Design and implementation of the tree-based fuzzy logic controller," *IEEE Trans. Syst., Man, Cybern. B*, vol. 27, pp. 475–487, June 1997.
[2] R. Takeuchi, M. Tsumura, M. Tadauchi, and H. Shio, "Color image scanner with an RGB linear image sensor," *J. Image Technol.*, vol. 14, pp. 68–72, 1988.
[3] P. C. Hung, "Colorimetric calibration in electronic imaging devices using a look-up-table model and interpolations," *J. Electron. Imag.*, vol. 2, pp. 53–61, 1993.
[4] H. R. Kang and P. G. Anderson, "Neural network application to the color scanner and printer calibrations," *J. Electron. Imag.*, vol. 1, pp. 125–135, 1992.
[5] C. W. Chang and P. R. Chang, "Neural plant inverse control approach to color error reduction for scanner and printer," in *Proc. Int. Conf. Neural Networks*, 1993, pp. 1979–1983.
[6] J. M. Jou, C. C. Chin, and Y. R. Li, "PASS: A package for automatic scheduling and sharing pipelined data paths," in *Proc. IEEE Int. Symp. Circuits Systems*, 1991, pp. 1769–1772.
[7] *0.8 μm SPDM Technology Manual*, Computer & Communication Lab., Industry Technol. Research Inst., Taiwan, R.O.C., 1993.