A Reinforcement-Learning Approach to Color Quantization

Chien-Hsing Chou[1]*, Mu-Chun Su[2], Yu-Xiang Zhao[3] and Fu-Hau Hsu[2]

[1]*Department of Electrical Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.*
[2]*Department of Computer Science & Information Engineering, National Central University, Taoyuan, Taiwan 320, R.O.C.*
[3]*Department of Computer Science & Information Engineering, National Quemoy University, Kinmen, Taiwan 892, R.O.C.*

Abstract

Color quantization is a process of sampling three-dimensional color space (e.g. RGB) to reduce the number of colors in a color image. By reducing to a discrete subset of colors known as a color codebook or palette, each pixel in the original image is mapped to an entry according to these palette colors. In this paper, a reinforcement-learning approach to color image quantization is proposed. Fuzzy rules, which are used to select appropriate parameters for the adaptive clustering algorithm applied to color quantization, are built through reinforcement learning. By comparing this new method with the original adaptive clustering algorithm on 30 color images, our method shows an improvement of 3.3% to 5.8% in peak signal to noise ratio (PSNR) values on average and results in savings of about 10% in computation time. Moreover, we demonstrate that reinforcement learning is an efficacious as well as efficient way to provide a solution of the learning problem where there is a lack of knowledge regarding the input-output relationship.

Key Words: Color Quantization, Color Reduction, Classifier Systems, Pattern Recognition, Reinforcement Learning, Neuro-Fuzzy Systems, Machine Learning

1. Introduction

True type color images can consist of more than 10 million (2^{24}) possible colors in a 24 bit full RGB color space. Consequently, it is very important in games and mobile devices to reduce the number of image colors for presentation, transmission and compression of color images. The most popular techniques for color reduction in digital images are the multithresholding and color quantization approaches. In the multithresholding approach, proper thresholds are determined from color histograms defining the limits of the image color classes [1−5]. Since the approach is based on the assumption that object and background pixels in an image can be well distinguished by their colors, the multithresholding approach does not give satisfactory results in the case of complex images such as natural or textured images.

In color quantization, a true color image is irreversibly transformed into a color-mapped image consisting of K carefully selected representative colors. The ultimate goal of color quantization is to select K representative colors to ensure minimization of a specific distortion measure between the original and the quantized images [6−17]. In general, these color quantization algorithms can be divided into two main categories: (a) splitting algorithms [7−10] and (b) clustering-based algorithms [6, 11−17]. In this paper, we aim at improving the clustering-based algorithms. The performance of these clustering-algorithm-based techniques varies greatly depending on how the K representative colors are chosen. These

*Corresponding author. E-mail: chchou@mail.tku.edu.tw techniques have to make a tradeoff between their computational efficiency and minimization of the distortion measure. For example, the K-means algorithm can efficiently minimize the quantization error if enough number of iterations are allowed [12,13]. Hsieh and Fang proposed an adaptive clustering (**AC**) algorithm for color quantization [14], in their algorithm the 3D RGB histogram of an input image is calculated first to generate a sorted histogram bin list. Then the K palette colors corresponding to the representative dominant colors are generated from the sorted histogram bin list. By controlling carefully for a distance threshold, the palette colors will be adaptively determined for the best situation. However, the authors did not provide guidance in how to choose an appropriate distance threshold. It is the quest for resolution of this question which motivated us to use a neuro-fuzzy system trained in a reinforcement-learning environment to choose an appropriate distance threshold for the AC algorithm in order to further improve its performance.

An overview of reinforcement learning can be found in [18]. The article surveys the field of reinforcement learning from a computer-science perspective. The approaches for solving reinforcement-learning problems can be roughly divided into two strategies. The first is to search among the range of behaviors in order to find one that performs well in the environment. This approach has been adapted by work in genetic algorithms and genetic programming [19] as well as some more novel search techniques [20]. The second strategy involves using statistical techniques and dynamic programming methods to estimate the effectiveness of taking actions in different states of the world, for example, adaptive heuristic critic [21], temporal difference ($TD(\lambda)$) [22], Q-learning [23, 24], etc. Each approach has its own relative advantages and disadvantages. For example, some reinforcement learning algorithms may suffer from the inordinate amount of time required to learn an effective strategy.

In our previous work [25], we developed an adaptive classifier-system-based neuro-fuzzy inference system called **ACSNFIS**, which used a special reinforcement learning algorithm to incrementally construct its architecture and tune the system parameters. The special reinforcement learning algorithm was motivated by the so-called *classifier system* which is a machine learning system that learns syntactically simple string rules (called classifiers) [19,26]. In this paper, we use the ACSNFIS to implement the reinforcement-learning-based color quantization algorithm.

The organization of the paper is as follows. The AC and the ACSNFIS algorithm are introduced in section 2. In section 3, we describe how to combine the ACSNFIS with the AC algorithm to solve the problem of color quantization. Experimental results of the proposed method are given in section 4. Finally, in section 5 we present the conclusions of the paper.

2. Brief Review of AC and ACSNFIS Algorithms

Before we introduce the way to construct a reinforcement-learning-based approach to color image quantization, we first briefly review the AC algorithm proposed by Hsieh and Fan [14].

2.1 Brief Review of AC Algorithm

The AC algorithm is briefly described as follows:

Step 1: Accumulate all the pixels with the same color in the input image to form a 3D RGB color histogram. The N_b least significant bits of each color component are set to be zero while the histogram is calculated, where the parameter N_b is determined by the user. Here we assume that the number of the histogram bins is C. For example, each of two color images shown in Figure 1 consists of 256 × 256 pixels. We first sum up the histogram bins for the two images. Figures 1(a) and (b) contain a total of 61,389 and 33,870 different colors, respectively. Then, the three least significant bits ($N_b = 3$) of each color component (R, G and B) are set to be 0. The number of reduced histogram bins, C, of Figures 1(a) and (b) become 2,860 and 992, respectively. The advantage of such a procedure is that the reduced RGB color space design can not only shorten the execution time but also easily locate the dominant colors in the image. Furthermore, Figure 1(a) contains more histogram bins than Figure 1(b) does. This means that Figure 1(a) is more colorful than Figure 1(b). In our opinion, one should also consider this information in quantizing a color image. More details are given in section 3.

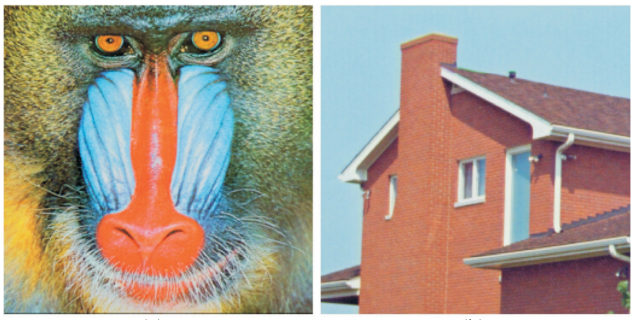
(a) (b)

Figure 1. Two color images. (a) Baboon; (b) House.

Step 2: Sort the histogram bins in a descending order according to the histogram values. Table 1 shows the list of the top 5 ordered histogram bins for Figure 1(a).

Step 3: An adaptive clustering algorithm is used to incrementally select K dominant colors from the C histogram bins $\{b_1, b_2, \ldots, b_C\}$.

(a) Set $i = 1$ and $k = 1$. Assign the histogram bin b_1 to dominant color DC_1. Set $\eta = \eta_0$, where η is a distance threshold and η_0 is its initial value pre-specified by the user.

(b) Set $i = i + 1$. Find the nearest neighbor of b_i among the already found dominant colors according to their RGB values. Let d_{im} denote the distance from b_i to its nearest neighbor. Suppose the nearest neighbor to be dominant color DC_m. Note that the distance between the histogram bin b_i and dominant color DC_m is defined as follows:

$$d_{im}(b_i, DC_m) = \sqrt{(R_i - R_m)^2 + (G_i - G_m)^2 + (B_i - B_m)^2} \tag{1}$$

(c) If $d_{im} \leq \eta$, then assign b_i to DC_m, otherwise, set $k = k + 1$ and assign b_i to be a new dominant color DC_k.

(d) If K dominant colors have been generated, then go to Step (f). Else repeat Step (b) and Step (c) till all bins have been processed.

(e) The value of the threshold η is updated as the following equation:

$$\eta = \eta - \eta_d \tag{2}$$

where η_d is the decreasing parameter pre-specified by the user. Then set $i = 1$ and go back to Step (b).

(f) Assign all the unassigned bins to their nearest dominant colors.

Step 4: Re-calculate the R, G and B values of each dominant color as the weighted average of the R, G and B values of all the bins assigned to it.

By controlling the distance threshold η carefully, the palette colors will be adaptively determined for the best situation. However, Hsieh and Fan did not provide guidance about how to choose appropriate η_0 and η_d. They empirically set the initial distance threshold η_0 to be $\sqrt[3]{255^3/K}$. And they set the decreasing parameter η_d to be 4 if the number of unfound dominant colors is larger than 64, otherwise they set η_d to be 2.

Through our experiments with setting different η_0 and η_d, we found that the performance of the AC algorithm greatly depends on these two parameters (especially η_0). Better results could be achieved if the appropriate parameters η_0 and η_d were used in the clustering algorithm. We apply an AC algorithm to quantize Figure 1(b) with Hsien and Fan's suggestion and 7 different sets of η_0 and η_d, for which the peak signal-to-noise ratio (**PSNR**) values are given in Table 2. According to the performance results as shown in Table 2, we have following observations: First, Hsien and Fan's suggestion of η_0 and η_d doesn't achieve the optimal performance. Second, the best set of parameters η_0 and η_d among the entire sets is not identical for quantizing different color numbers (color numbers of 8 and 16 in this case). Through some experiments, we also found that selecting an appropriate η_0 and η_d is re-

Table 1. The list of top 5 ordered histogram bins for Figure 1(a) ($N_b = 3$)

Order	1	2	3	4	5	…
Histogram value (number of pixels)	382	337	231	220	182	…
R component value	144	152	240	136	240	…
G component value	192	192	72	192	72	…
B component value	224	224	40	224	48	…

Table 2. The performance comparisons of Hsien and Fan's suggestion and another 7 sets of parameters (Figure 1(b))

Parameter Setting	PSNR (db)	
	$K = 8$	$K = 16$
$\eta_0 = \sqrt[3]{255^3/K}$, $\eta_d = 2$ (Hsien and Fan's suggestion)	24.82	28.33
$\eta_0 = 32$, $\eta_d = 2$	25.78	**29.14**
$\eta_0 = 32$, $\eta_d = 8$	25.78	**29.14**
$\eta_0 = 64$, $\eta_d = 2$	**26.41**	28.89
$\eta_0 = 64$, $\eta_d = 8$	**26.41**	28.15
$\eta_0 = 128$, $\eta_d = 8$	25.39	28.51
$\eta_0 = 160$, $\eta_d = 2$	24.52	28.03
$\eta_0 = 160$, $\eta_d = 8$	24.82	29.06

lated to the desired number of quantization level K, and the number of histogram bins C of the input image. This means that we need an expert or a prediction system to provide appropriate η_0 and η_d to deal with varied quantizing conditions. These findings motivated us to use a learning system to predict η_0 and η_d under different conditions of K and C. However, we are unable to construct the training data set to train a learning system through supervised learning since we lack knowledge of the *input-output* relationship between the input parameters K and C and the desired output η_0 and η_d. Hence, one possible way to solve this problem is to apply reinforcement learning to construct a learning system. In our previous work [25], the ACSNFIS uses a special reinforcement learning algorithm to incrementally construct its architecture and tune the system parameters. Therefore, the ACSNFIS can be one of the possible solutions to construct a learning system to predict appropriate η_0 and η_d.

2.2 Brief Review of ACSNFIS

In our previous work [25], a new approach to fuzzy classifier systems was proposed and a class of adaptive classifier-system-based neuro-fuzzy inference system referred to as ACSNFIS was developed to implement such new fuzzy classifier systems. The proposed ACSNFIS can not only tune its parameters but also incrementally construct its architecture in a reinforcement-learning environment. Through injecting new rules into the system, an ACSNFIS can explore the possible solution space. The major difference between the proposed ACSNFIS and other neuro-fuzzy systems is that each rule in an ACSNFIS is associated with a strength parameter updated by a credit assignment method such as the fuzzy bucket brigade algorithm.

An ACSNFIS is a five-layer feedforward network, as shown in Figure 2. The proposed ACSNFIS system based on the reinforcement-learning model is schematically shown in Figure 3. More Deditals of ACSNFIS is given in [25].

3. The Proposed Modified Adaptive Clustering Algorithm

In this section, we proposed the modified adaptive clustering (MAC) algorithm combined with ACSNFIS and AC algorithm. Figure 4 shows the architecture of the MAC algorithm. In our conception, choosing the suitable values of η_0 and η_d should be related to the desired number of quantization levels K. Additionally, the number of histogram bins C of the input image may also influence the performance of the AC algorithm. These two variables, K and C, are selected to be the inputs for the ACSNFIS as shown in Figure 4. Instead of using K and C directly, we use $K' = \log_2(K)$ and $C' = \dfrac{C}{\sqrt{W \cdot H}}$ as the inputs for the ACSNFIS. The use of the logarithm function is to compress the difference between two large numbers (e.g. since $256 - 128 = 128$, but $\log_2 256 - \log_2 128 = 1$). For large values of K, a small difference in the values of the two parameters, η_0 and η_d, will not result in large different visual effects. However, for small values of K (e.g. $K = 8$) different combinations of dominant colors will greatly influence visual effects. Hence, we consider that using $K' = \log_2(K)$ should be more suitable than using K. Moreover, we use $C' = \dfrac{C}{\sqrt{W \cdot H}}$ to be the second input variable, where $W \cdot H$ denotes the size of an input image. If an image has a larger size, it usually contains greater amounts of histogram bins C. Consequently it is reasonable that we simply normalize the number of histogram bins C to be C' with respect to its image size, and use C' to be another input for the ACSNFIS.

After we enter two inputs, K' and C', the outputs, η_0 and η_d, are predicted by the ACSNFIS according to the

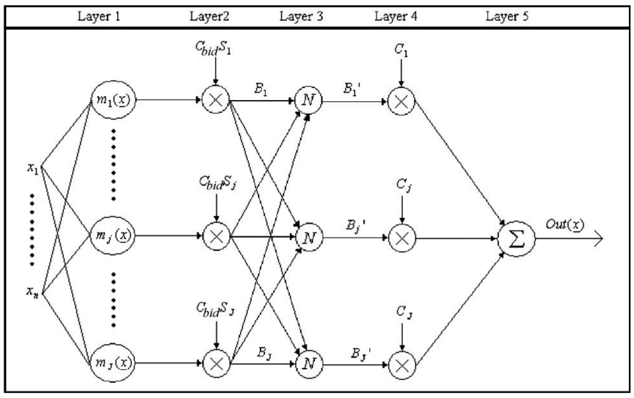

Figure 2. The architecture of the ACSNFIS.

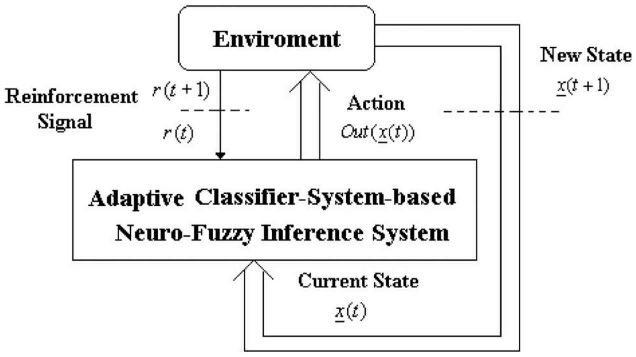

Figure 3. The proposed ACSNFIS system in the reinforcement-learning environment.

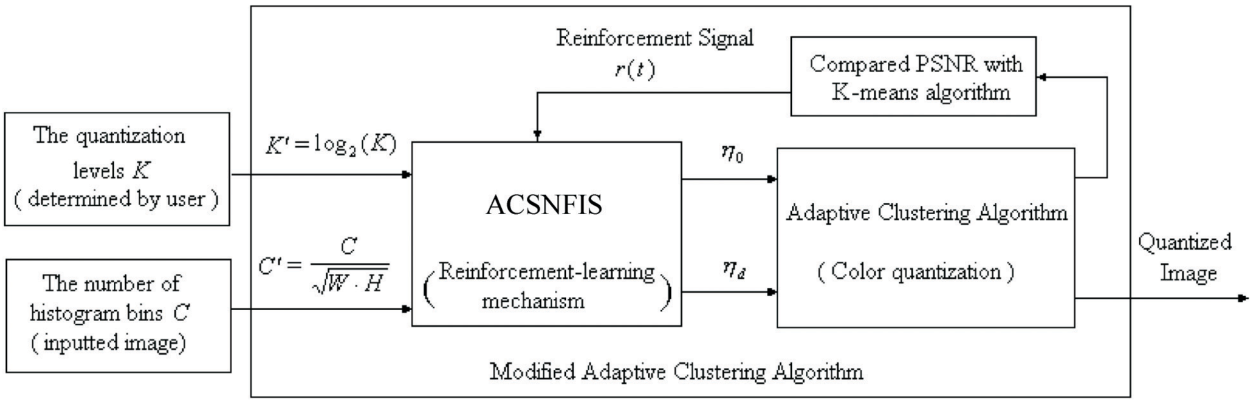
Reinforcement Signal
r(t)
Compared PSNR with K-means algorithm
The quantization levels K (determined by user)
K' = log₂(K)
The number of histogram bins C (inputted image)
C' = C/√(W·H)
ACSNFIS
(Reinforcement-learning mechanism)
η₀
η_d
Adaptive Clustering Algorithm
(Color quantization)
Quantized Image
Modified Adaptive Clustering Algorithm

Figure 4. The architecture of the modified adaptive clustering (MAC) algorithm combining the ACSNFIS with the AC algorithm.

initial parameters applied in the AC algorithm. Then, the quantized image is reproduced by the AC algorithm. In the training process, we calculate PSNR values to evaluate the quantized image performance and provide further comparison with another excellent color quantization algorithm (K-means algorithm) to generate the reinforcement signal. If the quantized image of the AC algorithm performs a comparable effect to the K-means algorithm, the ACSNFIS will then receive a reinforcement signal with a larger value. Otherwise, it will receive a reinforcement signal with a lower value. The temporal difference (*TD*) of the reinforcement signal, *r*(*t* + 1) − *r*(*t*), will be used to adjust the parameter in ACSNFIS by rewarding the fuzzy rules by raising their strength or punishing the fuzzy rules by lowering their strength. After irrelatively training the ACSNFIS through the above process, the MAC algorithm, which combines the ACSNFIS with the AC algorithm, is shown to provide better performance than the original AC algorithm. The comparisons are given in section 4.

In the ACSNFIS, the crucial problem is how to determine a reinforcement signal. The success of training the ACSNFIS through reinforcement learning is always related to providing an appropriate reinforcement signal. Before we introduce our chosen method to determine the reinforcement signal, we first define how to evaluate the performance between an original image and its quantized image. The PSNR is usually adopted to evaluate this performance. Consequently, it can be used as the basis of the reinforcement signal. The PSNR is defined as follows:

$$PSNR = 10\log_{10}\left(\frac{255^2}{\frac{1}{3\cdot W\cdot H}\sum_{i=1}^{W}\sum_{j=1}^{H}\left[\sum_{k=1}^{3}(A(i,j,k)-B(i.j.k))^2\right]}\right) \tag{3}$$

where $A(i, j, k)$ and $B(i, j, k)$ are the RGB values of pixel (i, j) taken from the original image and its corresponding quantized image, respectively.

Here, we use twenty color images for training, and some of them are shown in Figure 5. For all training images, we apply the K-means algorithm [27] to quantize the training images with $K = 2^n$, for $n = 1, 2, \ldots, 9$. For each run we continued the K-means algorithm until 50 iterations were reached. Table 3 tabulates the resulting PSNR values achieved for Figure 5(a).

After we obtain the PSNR values of all training images, the reinforcement signal, $r(t)$, of the ACSNFIS with different K is further determined to be:

IF $PSNR_{MAC}(K) > PSNR_{KM}(K)$
  THEN $r(t) = 1$;
ELSE IF $PSNR_{KM}(K) \geq PSNR_{MAC}(K) \geq (PSNR_{KM}(K) - 10)$
  THEN $$r(t) = \frac{PSNR_{MAC}(K) - (PSNR_{KM}(K) - 10)}{10} \quad (4)$$
ELSE $r(t) = 0$;

where $PSNR_{MAC}(K)$ and $PSNR_{KM}(K)$ represent the PSNR values resultant from the MAC algorithm and the K-means algorithm, respectively. The idea of using the reinforcement signal given in Eq. (4) is very straightforward. If the MAC algorithm achieves comparable performance to the K-means algorithm, we can obtain a larger value of $r(t)$. Otherwise, $r(t)$ will have a lower value when the MAC algorithm performs worse than the K-means algorithm. If $K \neq 2^n$, for $n = 1, 2, \ldots, 9$, we can simply use the linear interpolation to approximate the values of $PSNR_{KM}(K)$. One thing should be noticed here. In the training process of the MAC algorithm, the K-means algorithm is applied to compare the quantized image of MAC, and generate the reinforcement signal $r(t)$ by Eq. (4). Then the reinforcement signal $r(t)$ is used to update the parameter of the MAC algorithm. However, the K-means algorithm is not executed in the testing process, because we don't need to generate the reinforcement signal and update the parameter of the MAC algorithm.

Besides, in the training process, the final goal is to generate a well-trained MAC algorithm that may outperform K-means with 50 times iteration for the entire training images. In fact such situation is never met in the training process, however, the Eq. (4) should still consider this situation if MAC outperform K-means. Finally the temporal difference of the reinforcement signal will be used to reward or punish the strength of each rule by using the Reward Mechanism [25].The training task was decomposed into three sub-tasks such as are shown in

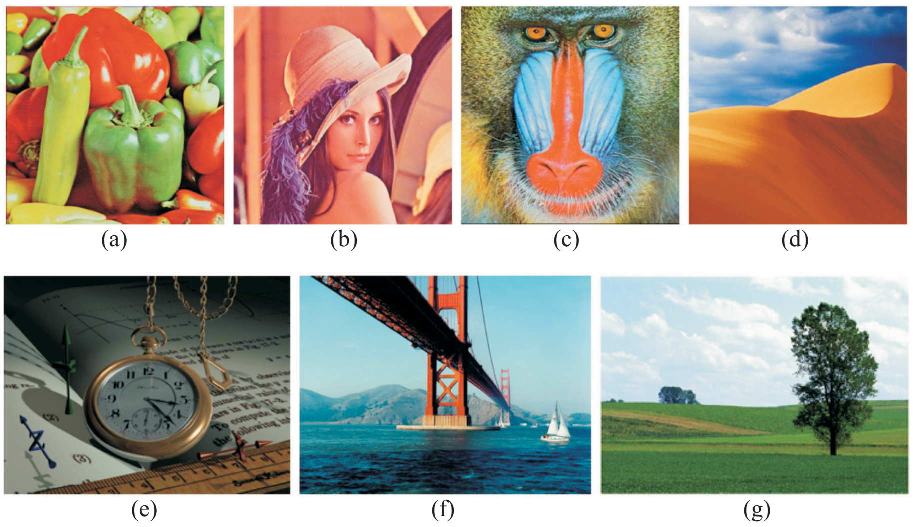

Figure 5. Some examples of the training images.

Table 3. The PSNR values achieved for the Peppers image (Figure 5(a)) by running the K-means algorithm

Number of Colors (K)	2	4	8	16	32	64	128	256	512
PSNR (db)	17.262	20.803	24.428	27.255	29.626	32.023	34.238	36.262	38.130

Table 4. At each sub-task, we trained the ACSNFIS for 500 trials. After 1500 trials, the ACSNFIS established 89 fuzzy rules.

4. Experimental Results

The experimental results were implemented in Borland C++ Builder using an Intel Pentium 4 CPU 2 GHz with 1 GB RAM under Microsoft Window 2000 environments. The parameters of the ACSNFIS, θ_m, θ_s, C_{bid}, S_r, $S_{\max}$, and r_t, were chosen to be 0.3, 0.1, 0.05, 2, 2, and 0.05, respectively. The learning rate δ was selected to be 0.05. The parameter N_b was chosen to be 3 for the AC algorithm and the MAC algorithm. For comparison purposes, the AC algorithm and the K-means algorithm were also applied to quantize the same set of test images. The number of iterations was set at 10 for training the K-means algorithm. For K-means algorithm, we have tried different initializations of K dominant colors during the training procedures. The PSNR value and computation time of the K-means algorithm were evaluated finally as the averages from these different initializations.

There were two test image sets used for testing, the first test set contained four images as shown in Figure 6(a) to 9(a) and the second test set contained 30 color images downloaded from the USC-SIPI image database [28]. The number of histogram bins C for Figures 6(a), 7(a), 8(a), and 9(a) were 765, 592, 638, and 1134, respectively. According to their different K' and C', two parameters, η_0 and η_d, were predicted by the trained ACSNFIS. The quantized results of four test images are shown in Figure 6 to 9. Table 5 tabulates the comparison results. In Table 5, the highlighted (bold and shaded) entries correspond to the best performance among the three methods. Two kinds of comparisons are listed. One is the PSNR value of all reproduced images, while the other is the computation time. In the first comparison, the K-means algorithm achieves the highest PSNR for several cases, especially for the Tiffany image (Figure 8). But it also pays the price of having the longest computational time. The proposed MAC algorithm achieves the highest PSNR on the other cases, especially for the Science image (Figure 9). Moreover, we observe that the MAC algorithm requires the least processing time of all the cases. This means that the MAC algorithm achieves comparable performance (PSNR) to K-means algorithm but only spends about 20% of the required execution time on average. Here we would like to emphasize that if the KM algorithm is processed more than 30 to 50 iterations, the performance of the KM algorithm will outperform the MAC algorithm in all cases. However, this level of performance necessitates a very long computational time. We also observe that the computation time of the MAC algorithm is shorter than that of the AC algorithm in all cases (computational time is reduced by 10.4% on average). According to the experimental results, we show that our proposed MAC algorithm outperform the AC algorithm in execution time and PSNR value.

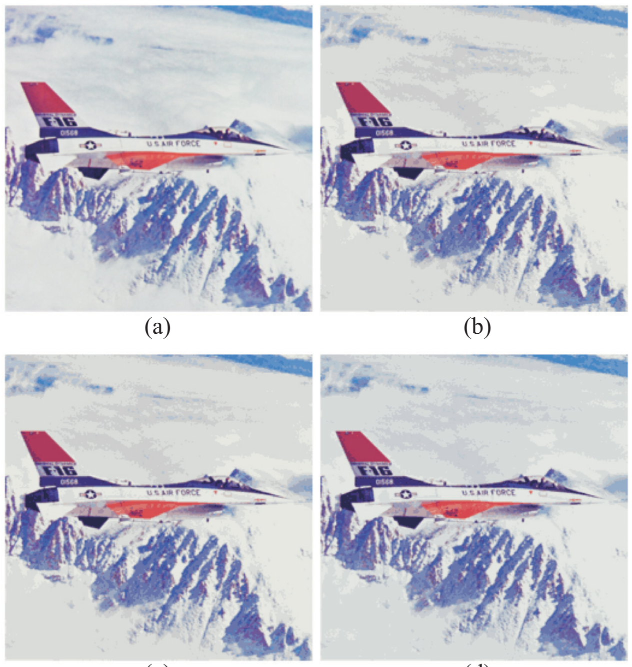

Figure 6. Test results for Jet image. (a) The original image. The quantized images with 32 colors by using (b) AC; (c) K-means; (d) MAC algorithm.

Table 4. The individuated sub-tasks in the problem training process for color quantization

Sub-task	Training conditions of the application to color quantization
(1)	Using three images among the training images, and $K = 2^n$, for $n = 1, 2, \ldots, 9$
(2)	Using all the training images, and $K = 2^n$, for $n = 1, 2, \ldots, 9$
(3)	Using all the training images, and $K = 2, 3, 4, \ldots, 512$

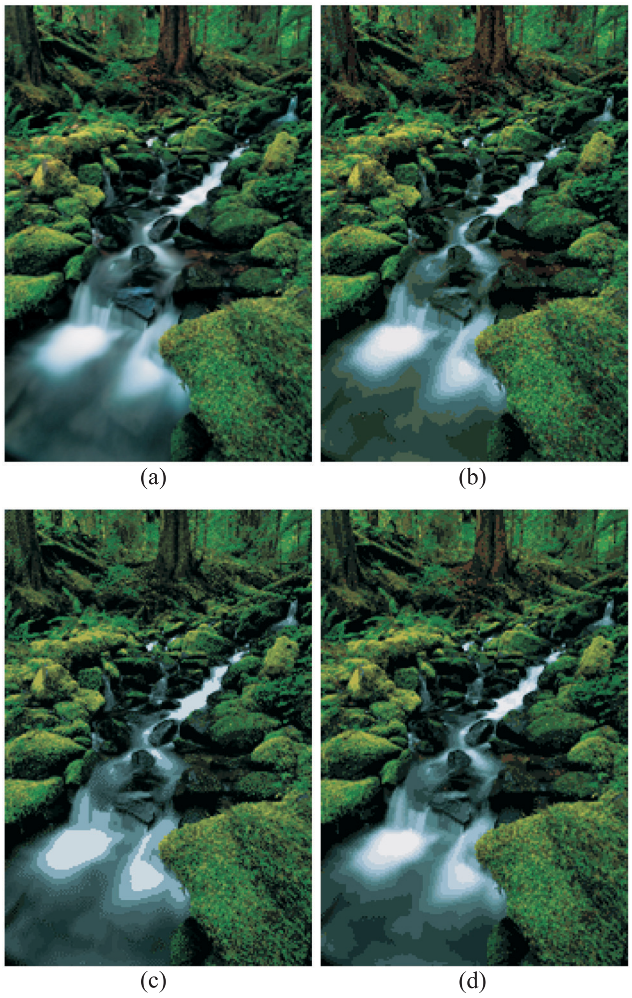

Figure 7. Test results for Waterfall image. (a) The original image. The quantized images with 32 colors by using (b) AC; (c) K-means; (d) MAC algorithm.

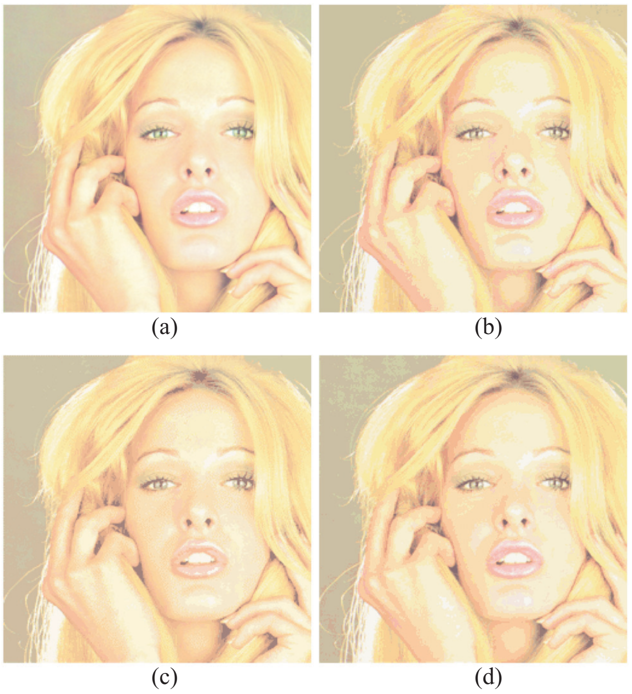

Figure 8. Test results for Tiffany image. (a) The original image. The quantized images with 32 colors by using (b) AC; (c) K-means; (d) MAC algorithm.

5. Conclusion

In this paper, we have shown a way to apply a reinforcement learning algorithm to solve the problem of color quantization where there is lack of knowledge about the input-output relationship. Reinforcement learning is used to generate the fuzzy rules for choosing more appropriate parameters that are able to improve the performance of the original AC algorithm. Compared with the original AC algorithm in testing 30 color images, the MAC algorithm improves average PSNR values by 3.3% to 5.8%, and with less computational time. Compared with the K-means algorithm, the MAC algorithm provides comparable quantized image performance but requires only 20% of the computational time.

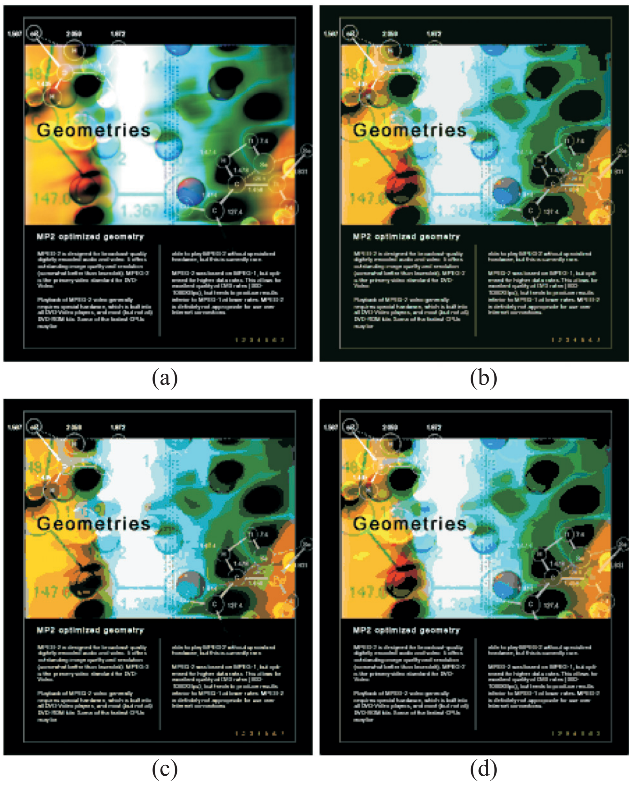

Figure 9. Test results for Science image. (a) The original image. The quantized images with 32 colors by using (b) AC; (c) K-means; (d) MAC algorithm.

Table 5. The performance comparisons of four tested images

Testing Image	No. of Colors (*K*)	PSNR (db)			Time (s)		
		AC	K-means	MAC	AC	K-means	MAC
Jet	16	30.052	**31.491**	30.966	0.733	1.608	**0.655**
256 × 256	32	32.237	33.226	**33.431**	0.889	2.781	**0.861**
C = 765	64	34.121	35.361	**35.897**	1.256	4.893	**1.111**
	128	35.609	**37.600**	37.213	1.938	9.514	**1.687**
	256	38.131	**40.045**	38.579	3.171	18.767	**2.686**
Waterfall	16	28.525	29.376	**29.403**	0.502	1.032	**0.486**
172 × 256	32	31.791	31.926	**32.611**	0.626	1.721	**0.579**
C = 592	64	34.452	35.269	**35.322**	0.891	3.438	**0.766**
	128	36.922	**37.569**	37.466	1.296	6.517	**1.187**
	256	38.604	**39.893**	38.798	3.252	12.313	**2.889**
Tiffany	16	28.417	**30.883**	29.724	0.704	1.516	**0.638**
256 × 256	32	29.850	**33.986**	31.432	0.875	2.562	**0.779**
C = 638	64	31.608	**35.976**	33.908	1.188	5.064	**1.048**
	128	33.643	**38.019**	34.667	1.812	9.702	**1.593**
	256	35.626	**40.079**	36.567	3.108	16.968	**2.673**
Science	16	22.363	24.052	**24.185**	0.721	1.421	**0.704**
222 × 256	32	23.942	26.466	**26.656**	0.921	2.189	**0.811**
C = 1134	64	26.905	28.569	**29.177**	1.234	4.360	**1.077**
	128	29.740	30.598	**31.618**	1.845	8.375	**1.627**
	256	31.915	32.722	**33.313**	2.985	14.812	**2.610**

Acknowledgement

This work was partly supported by the National Science Council, Taiwan, R.O.C, under the NSC 98-2218-E-238-001 and the NSC 99-2221-E-238-017.

References

[1] Kittler, J. and lllingworth, J., "Minimum Error Thresholding," *Pattern Recognition*, Vol. 19, pp. 41–47 (1986).

[2] Papamarkos, N. and Gatos, B., "A New Approach for Multithreshold Selection," *Comput. Vision Graph. Image Process. – Graph. Models Image Process.*, Vol. 56, pp. 357–370 (1994).

[3] Sahoo, P. K., Soltani, S. and Wong, A. K. C., "A Survey of Thresholding Techniques," *Comput. Vision, Graph. Image Process.*, Vol. 41, pp. 233–260 (1988).

[4] Reddi, S. S., Rudin, S. F. and Keshavan, H. R., "An Optimal Multiple Threshold Scheme for Image Segmentation," *IEEE Trans. on System, Man, and Cybernetics*, Vol. SMC-14, pp. 661–665 (1984).

[5] Tsai, D. M., "A Fast Thresholding Selection Procedure for Multimodal and Unimodal Histograms," *Pattern Recognition Letters*, Vol. 16, pp. 653–666 (1995).

[6] Scheunders, P., "A Comparison of Clustering Algorithms Applied to Color Image Quantization," *Pattern Recognition Letters*, Vol. 18, pp. 1379–1384 (1997).

[7] Heckbert, P., "Color Image Quantization for Frame Buffer Display," *Comput. Graph.*, Vol. 16, pp. 297–307 (1982).

[8] Wan, S. J., Prusinkiewicz, P. and Wong, S. K. M., "Variance Based Color Image Quantization for Frame Buffer Display," *Color Res. Applicat.*, Vol. 15, pp. 52–58 (1990).

[9] Ashdown, I., "Octree Color Quantization," *Radiosity-A Programmer's Perspective*, New York: Wiley (1994).

[10] Gervautz, M. and Purgathofer, W., "A Simple Method for Color Quantization: Octree Quantization," in *Graphics Gems*, A. S. Glassner, Ed. New York: Academic, pp. 287–293 (1990).

[11] Cheng, G., Yang, J., Wang, K. and Wang, X., "Image Color Reduction Based on Self-Organizing Maps and Growing Self-Organizing Neural Networks," *Sixth International Conference on Hybrid Intelligent Systems*, pp. 24–24 (2006).

[12] Verevka, O., *The Local K-means Algorithm for Color Image Quantization*, M.Sc. dissertation, Univ. Alberta, Edmonton, AB, Canada (1995).

[13] Frackiewicz, M. and Palus, H., "Clustering with K-Harmonic Means Applied to Colour Image Quantization," *2008 IEEE International Symposium on Signal Processing and Information Technology*, pp. 52–57 (2008).

[14] Hsieh, I. S. and Fan, K. C., "An Adaptive Clustering Algorithm for Color Quantization," *Pattern Recognition Letters*, Vol. 21, pp. 337–346 (2000).

[15] Ashutosh, D., Bose, N. S. C., Kandula, P. and Kalra, P. K., "Modified Forward Only Counterpropogation Network (MFOCPN) for Improved Color Quantization by Entropy Based Sub-Clustering," *2007 International Joint Conference on Neural Networks*, pp. 1865–1870 (2007).

[16] Papamarkos, N., Atsalakis, A. E. and Strouthopoulos, P., "Adaptive Color Reduction," *IEEE Trans. on System, Man, Cybernetics-Part B*, Vol. 32, pp. 44–56 (2002).

[17] Cheng, F.-C. and Chen, Y.-K., "A New Approach of Image Segmentation Based on Gray-Level Clustering," *2009 IEEE International Symposium on Industrial Electronics*, pp. 775–778 (2009).

[18] Kaelbling, L. P., Littman, M. L. and Moore, A. W., "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, Vol. 4, pp. 237–285 (1996).

[19] Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI (1975).

[20] Schmidhuber, J., "A General Method for Multi-Agent Learning and Incremental Self-Improvement in Unrestricted Environments," In Yao, X. (Ed.), *Evolutionary Computation: Theory and Applications*, Scientific Publ. Co., Singapore (1996).

[21] Barto, A. G., Sutton, R. S. and Anderson, C. W., "Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems," *IEEE Trans. on System, Man, and Cybernetics*, Vol. 13, pp. 834–846 (1983).

[22] Sutton, R. S., "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, Vol. 3, pp. 9–44 (1988).

[23] Watkins, C. J. C. H., *Learning form Delayed Rewards*, Ph.D. thesis, King's College, Cambridge, UK (1989).

[24] Watkins, C. J. C. H. and Dayan, P., "Q-Learning," *Machine Learning*, Vol. 8, pp. 279–292 (1992).

[25] Su, M. C., Chou, C. H., Lai, E. and Lee, J., "A New Approach to Fuzzy Classifier Systems and Its Application in Self-Generating Neuro-Fuzzy Systems," *Neurocomputing*, Vol. 69, pp. 586–614 (2006).

[26] Goldberg, D., *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Welsley, MA (1989).

[27] Ball, H. and Hall, D. I., "Some Fundamental Concepts and Synthesis Procedures for Pattern Recognition Preprocessors," *Proc. of Int. Conf. Microwaves, Circuit Theory, and Information Theory*, Tokyo, Japan, pp. 281–297 (1964).

[28] The USC-SIPI Image Database. "http://sipi.usc.edu/database/".

Manuscript Received: Mar. 9, 2010
Accepted: Oct. 1, 2010