*2009 International Symposium on Computing, Communication, and Control (ISCCC 2009)*
*Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore*

# Color FIC by Adaptive Zero-Mean Method

Dr. Loay E. George [1] and Dr. Eman A. Al-Hilo [2]

[1] Baghdad University/College of Science/Iraq
[2] Kufa University/College of Medicine/Iraq

**Abstract.** In this work, a fractal image compression method based on zero-mean block matching method is adopted. In this method the mean of the range blocks is used instead of traditional offset parameter. This reduces and simplifies the computations of the affine parameters during the encoding process. To increase the compression gain the indexes of range mean bits have been coded using DPCM followed by shift coding. Also the scale indexes are shift coded. This method is tested on 24 bits/pixel color image. The data of the color image (R,G,B) are transformed to (Y,U,V) components (to take the advantage of the existing spectral correlation). Also the low spatial resolution of the human vision systems to the chromatic components (U, V) is utilized to increase the compression ratio without making significant subjective distortion. The test results conducted on Lena image indicated encoding time (34.64) sec, compression ratio (10.33) and PSNR (33.85). These results showed reduction in the encoding time about 54.51%, and increase in the compression ratio around 5.24%, and the image quality was nearly preserved in comparison with traditional method.

## 1. Introduction

Recently fractal compression of digital images has attracted much attention. Mathematically, It is based on theory of iterated function systems (IFS) developed by Hutchinson and Barnsly [1,2]. The first use of IFS for image compression was by Jacquin and Barnsly [3,4]. In IFS coding scheme, the image is partitioned into non-overlapped range blocks. For every block, a similar domain block is found using IFS mapping. The data of blocks of the compressed image are represented using the IFS mapping coefficients. Decoding proceed as follows. Provided the IFS transformations are contracting, the image converges to a stable image, resembling the original picture.

Recently some improvements and modifications have been studied which made the fractal technique a challenging candidate its encoding time needs to be reduced. One of the introduced improvements is based on zero-mean block matching method, which utilizes unconventional affine mapping [5,6,7]. Many researchers studied the implementation of FIC on color images using different methods [8,9,10]. In this work, an enhanced zero-mean fractal coding scheme for color images is introduced, the modified scheme implies the additional coding stages which are differential pluses coding modules (DPCM) and shift coding (SC). The reason behind using DPCM is the existence of spatial correlation between the mean values of range block. The output of DPCM of range blocks and the scaling indexes are highly peaked around zero, and in such case the use of shift coding would increase the compression gain.

In this project the loaded RGB color image is transformed to YUV color space, where Y is the luminance component, (U and V) are the chromatic components. In order to get an effective compression the U, V component have been down sampled [11]. Each component (Y,U and V) was coded individually using FIC method, as it is applied on gray image. In decoding stage the reconstructed image is obtained by establishing the YUV color bands using the inverse IFS mapping, and then they transformed to RGB components after up sampling the U and V bands [12].

## 2. IFS Coding of Zero-Mean Block

The offset factor is determined using traditional affain mapping, described by equation (1), has wide dynamic range (i.e., [-255,510]), this may cause large errors in some image regions (or points) especially those which belong to high contrast area. The results of the analysis conducted in this research indicate that the traditional offset factors require an additional bit (i.e., sign-bit). Also, the analysis results indicate that the correlation between the offset coefficients of the adjacent blocks is week and not similar to that found between the mean values of the adjacent blocks. So, to get over this disadvantage a change in IFS mapping equation was performed.

For a range block with pixel values ($r_o$, $r_1,...,r_{n-1}$), and the domain block ($d_o,d_1,...,d_{n-1}$) the contractive affine approximation is [13]:

$r_i' = sd_i + o$ , ……..…….....…….(1)

Where, $r_i'$ is the optimally approximated i[th] pixel value in the range block. $d_i$ is the corresponding pixel value in the domain block. The symbols $s,o$ represent the scaling and offset coefficients, respectively.

Taking the average of both sides in equation (1) the following equation is obtained:

$\bar{r} = s\bar{d} + o$ , ……..……....…….…..(2)

The subtraction of equation (2) from equation (1) leads to:

$r_i' - \bar{r} = s(d_i - \bar{d})$ ,…….....……..……...…(3)

So, this contractive affine transform could be rewritten to become in the form:

$r_i' = s\left(d_i - \bar{d}\right) + \bar{r}$ ,…….....(4)

Where, $\bar{r} = \frac{1}{m}\sum_{i=0}^{m-1} r_i$ ,…….....(5), $\bar{d} = \frac{1}{m}\sum_{i=0}^{m-1} d_i$ ,…….....……..(6)

From equation (4), the fractal parameters become $\{s, \bar{r}\}$ instead of the of the conventional $\{s,o\}$ coefficients in traditional IFS mapping equation (1).

The scale (s) parameter could be determined by applying the least mean square difference ($\chi^2$) criteria between the approximated ($r_i'$) and actual ($r_i$) values [7], that is:

$\chi^2 = \frac{1}{m}\sum_{i=0}^{m-1}(r_i' - r_i)^2$ ,…….....……..……..…….....(7)

$\frac{\partial \chi^2}{\partial s} = 0$ ,…….....…….....(8)

The straightforward manipulation for equations (7), (8) leads to:

$$s = \begin{cases} \dfrac{\frac{1}{m}\sum_{i=0}^{m-1} d_i r_i - \bar{r}\bar{d}}{\sigma_d^2} & if\ \sigma_d^2 > 0 \\ 0 & if\ \sigma_d^2 = 0 \end{cases}$$ ,…….....(9)

$$\chi^2 = \sigma_r^2 + s\left[ s\sigma_d^2 + 2\bar{d}\bar{r} - \frac{2}{m}\sum_{i=0}^{m-1} d_i r_i \right]$$ ……..(10)

Where, $\sigma_d{}^2 = \frac{1}{m}\sum_{i=0}^{m-1} d_i{}^2 - \bar{d}^2$ ,…….....(11) $\sigma_r{}^2 = \frac{1}{m}\sum_{i=0}^{m-1} r_i{}^2$ ,……..…..(12)

The range average parameter ($\bar{r}$) has a smaller dynamic range [0,255] than ($o$) parameter [-255, 511]. So, in case of implementing quantization it is more effective (in terms of minimizing the quantization error) to quantize the value of ($\bar{r}$) than to encode the ($o$) parameter.

Before the determination of ($\chi^2$) values, the values of the scale ($s$) and range average ($\bar{r}$) values coefficients should be bounded and quantized as following:

1. The value of range average ($\bar{r}$) should be bounded within the interval $\bar{r}_{min} \le \bar{r} \le \bar{r}_{max}$, where $\bar{r}_{min}$ and $\bar{r}_{max}$ are the lower and upper boundaries of the permissible mean values respectively. Normally they should always be within the range [0, 255].

2. Also the values of scale ($s$) parameters should be bounded within the interval $s_{min} \le s \le s_{max}$, where $s_{max}$ and $s_{min}$ are the lower and upper boundaries of the permissible values of the scale respectively.
3. The coefficient $\bar{r}$ should be quantized using the following equations:

$$Q_r = \frac{(255)}{2^{b_r}-1}, \ldots\ldots\ldots\ldots\ldots(13) \qquad i_r = round\left(\frac{\bar{r}}{Q_r}\right), \ldots\ldots\ldots\ldots\ldots(14) \quad \bar{r}_q = i_r Q_r, \ldots\ldots\ldots\ldots\ldots\ldots\ldots(15)$$

Where, $b_r$ is the number of bits used to represent the quantization index of the block mean coefficient ($\bar{r}$); $Q_r$ is the quantization step of the mean coefficient ($\bar{r}$); $i_r$ is the quantization

4. The scale value should be quantized using the following equation:

$$Q_s = \begin{cases} \frac{s_{Max}}{2^{b_S-1}-1} & if\ s_{max} = -s_{min} \\ \frac{s_{Max}-s_{Min}}{2^{b_S}-1} & if\ s_{max} \neq -s_{min} \end{cases}, \ldots\ldots(16) \quad i_S = round\left(\frac{s}{Q_s}\right), \ldots\ldots(17) \quad s_q = i_s Q_s, \ldots\ldots\ldots\ldots\ldots\ldots\ldots(18)$$

Where $s_{max}$ is the greatest permissible value of scale coefficient; $s_{min}$ is the lowest permissible value of scale coefficient; $b_s$ is the number of bits used to represent the quantization index of the scale coefficient; $Q_s$ is the quantization step of the scale coefficient; $i_s$ is the quantization index of the scale coefficient and $s_q$ is the quantized value of the scale coefficient.

## 3. Encoding Process

The encoding method could be summarized by the following steps:

1. Load bitmap image and put its data in (R,G,B) arrays (i.e., three 2D arrays).
2. Convert (R,G,B) arrays to (Y,U,V) arrays.
3. Down sample U and V to quarter of its original size.
4. For each component (i.e., the original Y, and the down sampled U and V) do:

Establish the range image (array).

Down sample the range image to produce the domain array.

Create range and domain pool by partitioning:

(1) The range array into non-overlapping fixed blocks, to generate the range blocks ($r_1, \ldots, r_n$).

(2) The domain must be partitioned into overlapping blocks, using specific step size, to generate the domain blocks ($d_1, \ldots, d_n$). They should have the same size of range blocks.

For each range block do the following:

(1) Calculate the range average ($\bar{r}$) block.

(2) Quantize range average ($\bar{r}$) according to equations (13-15).

(3) Pick up a domain block from the domain pool.

(4) Perform one of the isometric mappings.

(5) Calculate the scale ($s$) coefficient using equation (9).

(6) Apply the following condition to bound the value of ($s$) coefficient:

*If* $s < s_{min}$ *then* $s = s_{min}$

*Else if* $s > s_{max}$ *then* $s = s_{max}$

(7) Quantize the value ($s$) using equations (16-18).

(8) Compute the approximation error ($\chi^2$) using equation (21).

(9) After the computation of the IFS code and the sum of error ($\chi^2$) of the matching between the range and the tested domain block, the ($\chi^2$) is compared with registered minimum error ($\chi^2_{min}$); such that:

*If* $\chi^2 < (\chi^2_{min})$ *then*

$s_{opt} = i_s$; $\chi^2_{min} = \chi^2$

*PosI=domain block index*

*Sym=symmetry index*

*End if*

(10) If $\chi^2_{min} < \varepsilon$ then the search across the domain blocks is stopped, and the registered domain block is considered as the best matched block.

(11) Repeat steps (4) to (10) for all symmetry states of the tested domain block.

(12) Repeat steps (3) to (11) for all the domain blocks listed in the domain pool.

(13) The output is the set of IFS parameters (i.e., $i_s, i_r, posI, sym$) which should be registered as a set of fractal coding parameters for the tested range block.

(14) Repeat steps (1) to (12) for all range blocks listed in the range pool.

(15) Encode the range average index parameters ($i_r$) using DPCM and shift coding.
(16) Encode the scale index parameters ($i_s$) using shift coding.
(17) Store all IFS mapping parameters as an array of record. The length of this array is equal to the number of range blocks in the range pool.

## 4. Decoding Process

The decoding process is summarized by the following steps:

1. Generate the first reconstructed domain pool, as a blank image or as a piece of any available image.
2. The values of the indices of ($i_r$) for the range blocks should be shift and DPCM decoded and dequantized (using equations 13-15) to reconstruct the values of ($\bar{r}$).
3. The values of the indices ($i_s$) for the range blocks should be shift decoded and then dequantized using equations (16-18) to reconstruct the values of ($s$).
4. Choose the value of the maximum number of possible iterations, and the value of the threshold of mean square error (TMSE). Then, at each iteration do the following steps:
   a. For each range block determine from its corresponding IFS parameter (*PosI*) the coordinates ($x_d$, $y_d$) of the best matched domain block in order to extract the domain block ($d$) from the arbitrary domain image.
   b. Calculate $\bar{d}$ for the corresponding domain block.
   c. For each range block, its approximation ($r_i'$) is obtained according to equation (4).
   d. The generated ($r_i'$) block is transformed (either rotated, reflected or both) according to its corresponding IFS symmetry parameter value (*Sym*).
   e. Put the generated $r_i'$ block in its position in the decoded image plane (range image).
   f. Check whether there is other range block need to be reconstructed, if yes then repeat steps (c,d,e).
   g. Downsample the reconstructed image (range array) in order to produce a new domain array using the averaging (or integer) resampling.
   h. Calculate the mean square error (MSE) between the newly reconstructed range and the previous reconstructed range image. If the MSE is greater than (TMSE) value then the iteration continued, and the steps (a-g) should repeated; till reaching the attractor state (i.e., the reconstructed range image is very similar to the previous reconstructed image). Sometimes, the iteration continues till reaching the predefined maximum number of iterations (it is set 20) without the fulfilment of MSE stopping condition.
5. The above steps (4a-4h) should be applied upon the three components (Y,U,V) to produce the attractor of each component separately.
6. Convert (YUV) color components to RGB components using inverse (YUV) transform.
7. For evaluation purpose, calculate the fidelity criteria (PSNR and MSE) for each RGB component, then determine the overall value of the fidelity criteria for the RGB reconstructed image.

## 5. Tests Results

The proposed system was implemented using Visual Basic (Ver. 6.0). It is tested on laptop core 2 duo, centrino, 2 GHz processor.

The proposed system had been tested using Lena image (256x256 pixel, 24bits) as test image. The value of the parameters MaxAvg ($\bar{r}_{max}$) and MinAvg ($\bar{r}_{min}$) were fixed in all these tests at (0) and (256) respectively. To test the effect of one parameter the values of other parameters have been fixed. The test results show that best result set include PSNR (33.85) with appropriate CR (10.33) and encoding time (34.64) sec. Table (1) illustrates the effect of ScaleBits and AvgBits on PSNR and CR.

Table (1) Effects of ScaleBits and AvgBits on the compression performance parameters

| ScaleBits | | | AvgBits | | | PSNR | CR | ET (sec) |
|---|---|---|---|---|---|---|---|---|
| Y | U | V | Y | U | V | | | |
| 3 | 3 | 3 | 5 | 5 | 5 | 27.57 | 12.40 | 43.39 |
| 3 | 3 | 3 | 6 | 6 | 6 | 30.83 | 11.84 | 42.44 |
| 3 | 3 | 3 | 7 | 7 | 7 | 32.80 | 11.28 | 37.03 |
| 3 | 3 | 3 | 8 | 8 | 8 | 33.32 | 10.80 | 32.29 |
| 3 | 3 | 3 | 9 | 9 | 9 | 33.39 | 10.35 | 32.11 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 10 | 10 | 10 | 33.38 | 9.93 | 31.50 |
| 4 | 4 | 4 | 5 | 5 | 5 | 29.11 | 11.79 | 45.08 |
| 4 | 4 | 4 | 6 | 6 | 6 | 32.08 | 11.28 | 40.39 |
| 4 | 4 | 4 | 7 | 7 | 7 | 33.53 | 10.77 | 35.45 |
| 4 | 4 | 4 | 8 | 8 | 8 | 33.85 | 10.36 | 29.88 |
| 4 | 4 | 4 | 9 | 9 | 9 | 33.91 | 9.94 | 30.98 |
| 4 | 4 | 4 | 10 | 10 | 10 | 33.94 | 9.55 | 30.25 |
| 5 | 5 | 5 | 5 | 5 | 5 | 29.68 | 11.26 | 44.44 |
| 5 | 5 | 5 | 6 | 6 | 6 | 32.57 | 10.79 | 40.56 |
| 5 | 5 | 5 | 7 | 7 | 7 | 33.85 | 10.33 | 34.64 |
| 5 | 5 | 5 | 8 | 8 | 8 | 34.17 | 9.95 | 29.83 |
| 5 | 5 | 5 | 9 | 9 | 9 | 34.25 | 9.56 | 31.22 |
| 5 | 5 | 5 | 10 | 10 | 10 | 34.27 | 9.20 | 29.05 |
| 6 | 6 | 6 | 5 | 5 | 5 | 29.96 | 10.76 | 44.20 |
| 6 | 6 | 6 | 6 | 6 | 6 | 32.77 | 10.34 | 40.38 |
| 6 | 6 | 6 | 7 | 7 | 7 | 33.99 | 9.90 | 35.30 |
| 6 | 6 | 6 | 8 | 8 | 8 | 34.28 | 9.56 | 29.61 |
| 6 | 6 | 6 | 9 | 9 | 9 | 34.37 | 9.20 | 30.52 |
| 6 | 6 | 6 | 10 | 10 | 10 | 34.38 | 8.86 | 28.84 |
| 7 | 7 | 7 | 5 | 5 | 5 | 30.12 | 10.29 | 45.78 |
| 7 | 7 | 7 | 6 | 6 | 6 | 32.88 | 9.89 | 40.75 |
| 7 | 7 | 7 | 7 | 7 | 7 | 34.06 | 9.50 | 34.13 |
| 7 | 7 | 7 | 8 | 8 | 8 | 34.33 | 9.19 | 31.11 |
| 7 | 7 | 7 | 9 | 9 | 9 | 34.40 | 8.85 | 28.69 |
| 7 | 7 | 7 | 10 | 10 | 10 | 34.42 | 8.54 | 29.03 |

The results show that when ScaleBits and AvgBits is increased the PSNR increases, but compression ratio decreases. When, the values of ScaleBits and AvgBits are set 5 and 7, respectively, for all components (Y,U,V) a good PSNR and proper CR are attained. Table (2) illustrates the effect of MaxScale and MinScale on PSNR, CR and encoding time.

Table (2) Effects of MaxScale and MinScale on compression performance

| Max Scale | Min Scale | PSNR | CR | ET(sec) |
|---|---|---|---|---|
| 1 | -1 | 33.70 | 10.22 | 39.59 |
| 1.5 | -1.5 | 33.95 | 10.22 | 38.23 |
| 2 | -2 | 33.98 | 10.23 | 38.89 |
| 2.5 | -2.5 | 33.89 | 10.27 | 44.44 |
| 3 | -3 | 33.85 | 10.33 | 34.64 |

The results show that at MinScale=-3 and MaxScale=3 good PSNR and CR values are obtained While the encoding time is not significally affected by the variations of MaxScale and MinScale. Table (3) illustrates the effects of BlockSize parameter on PSNR, CR and encoding time.

Table (3) Effect of BlockSize on the compression performance parameters

| BlkSiz | PSNR | CR | ET(sec)) |
|---|---|---|---|
| **4x4** | **33.85** | **10.33** | **34.64** |
| **8x8** | **28.15** | **41.38** | **33.20** |
| **16x16** | **24.05** | **162.62** | **27.84** |

The results show that when the BlockSize increase the CR increases, but PSNR and ET are decreased. When the BlockSize is taken (4x4) the value of PSNR and CR is more appropriate than other sizes. Table (4) illustrates the effects of StepSize parameter on PSNR, CR and encoding time.

Table (4) Effect of StepSize values on the compression performance parameters

| StepSize | PSNR | CR | ET(sec) |
|---|---|---|---|
| 1 | 34.49 | 9.54 | 133.61 |
| 2 | 33.85 | 10.33 | 34.64 |
| 3 | 33.39 | 10.76 | 15.92 |
| 4 | 32.96 | 11.19 | 9.41 |

| | | | |
|---|---|---|---|
| 6 | 32.55 | 11.81 | 4.61 |
| 8 | 32.00 | 12.26 | 3.13 |
| 10 | 31.74 | 12.34 | 2.03 |
| 12 | 31.45 | 12.84 | 1.95 |
| 14 | 31.06 | 13.02 | 1.41 |
| 16 | 30.80 | 13.57 | 1.14 |
| 18 | 30.52 | 13.62 | 0.67 |
| 20 | 30.53 | 13.60 | 0.75 |

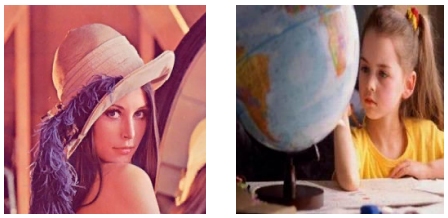
The results show that PSNR and ET are inversely affected with StepSize but CR is directly affected. The value StepSize=2 leads to an appropriate compromise between CR and PSNR.

The use of stopping search condition when monitoring the minimum matching error ($\varepsilon$) is efficient in reducing the required long fractal coding time. Table (5) illustrates the effects of ($\varepsilon$) on PSNR, CR and encoding time. The results show that PSNR and ET are inversely proportional with permissible error level ($\varepsilon$) but CR is directly proportional. The results show that there is a large reduction in ET (more than 7 times) without causes significant degradation in image quality.

Table (3) Effects of permissible error ( $\varepsilon$ ) value on the compression performance parameters

| $\varepsilon_o$ | PSNR | CR | ET(sec) |
|---|---|---|---|
| 0.1 | 33.94 | 10.33 | 47.95 |
| 0.2 | 33.94 | 10.33 | 49.94 |
| 0.3 | 33.95 | 10.33 | 52.55 |
| 0.4 | 33.96 | 10.34 | 46.58 |
| 0.5 | 33.94 | 10.33 | 45.44 |
| 0.6 | 33.93 | 10.33 | 48.31 |
| 0.7 | 33.93 | 10.33 | 41.55 |
| 0.8 | 33.93 | 10.33 | 3 8.91 |
| 0.9 | 33.89 | 10.33 | 36.89 |
| 1 | 33.85 | 10.33 | 34.64 |
| 2 | 33.61 | 10.69 | 20.34 |
| 3 | 32.28 | 10.80 | 14.95 |
| 4 | 31.72 | 10.97 | 11.88 |
| 5 | 31.16 | 11.61 | 10.22 |
| 6 | 30.67 | 11.89 | 7.50 |
| 7 | 30.25 | 12.34 | 6.25 |

Figure (1) shows samples of reconstructed images with best PSNR. They obtained when the values of the parameters ScaleBits, AvgBits, MinScale, MaxScale, BlockSize, StepSize and permissible error value are set (5, 7, -3, 3, 4x4, 2, 1), respectively.



**PSNR =33.85** **PSNR =36.43**

**CR =10.33** **CR =10.53**

**ET=34.64** **ET=33.13**

Fig (1) A samples of reconstructed images with best PSNR

## 6. Conclusions

Fractal color image compression using adaptive zero-mean block matching method is more effective than the classical IFS mapping method [7] in compressing color images, where:

a. There is increase in PSNR (around 5.3 %)
b. There is increase in the values of CR around (3%)
c. There is decrease in the encoding time (around 66%), this is due to the use of DPCM and Shift coding.
d. The proper value for the AvgBit lay within the range [5-10], while in the classical method the proper range is [5-7].

Further work is needed to enhance the IFS matching method. Different color space for example YIQ, YUV, and Lab…etc may be applied to find the best color space which can lead to high compression ratio as well as good image quality.

The proposed method could be combined with quadtree, or HV partition scheme to increase the compression performance.

## 7. References


[1] Hutchinson, J. E, "Fractal and Self Similarity", Indiana Univ. Math Journal, Vol. 35, p. 5, 1981.

[2] Bransly, M. F. "Fractal Every Where", Academic Press, San Diego, CA, 1988.

[3] Jacquin, A, "A Fractal Theory of Iterated Markov Operations, with Applications to Digital Image Coding", Ph.D., Thesis, Department of Mathematics, Georgia Institute of Technology, 1989.

[4] Jacquin, A, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations", IEEE Transactions on Image processing, Vol. 1. No.1, 1992.

[5] Tong, C. S.,"Fast Fractal Image Encoding Based on Adaptive Search", IEEE transaction on image processing, Vol.10, No. 9, 2001.

[6] George, L.E., "IFS Coding for Zero-Mean Image Blocks", Iraqi Journal of Science, Vol.47, No.1, 2006.

[7] Al-Hilo, E. A., George, L.E. "Speeding- up Fractal Color Image Compression Using Moments Features", Digital Image Computing: Techniques and Applications, 1-3 December 2008_Canberra, Australia. /DICTA.2008, Page(s):486-490.

[8] H¨urtgen, B., Mols, P. and Simon, S., "Fractal Transforms Coding of Color Images", Visual Communications and Image Processing, SPIE 94, Chicago, 1994.

[9] Zhaohuili, Zhao, L., Soma, y., "Fractal Color Image Compression", IEEE, 0-7695-0878-2/00, 2000.

[10] Al-Hilo, E. A., George, L.E., Al-Zuky, A. A. "Fractal Color Image Compression", Proceedings of the 4th International Conference on 17th–19th November 2008 Information Technology and Multimedia at UNITEN (ICIMU' 2008), pages(637-642) Malaysia.

[11] Ning, L., "Fractal Imaging", book, Academic Press, 1997.

[12] Sangwine, S. J., Horne, R., "The Color Image processing Handbook", Champan & Hall, 1998.