Reducing Color from Data Image File Using the Modification Thresholding Algorithm

Dr. Riyad Mubarak Abdullah

Address For Correspondence:
Dr. Riyad Mubarak Abdullah, Computer Sciences Department/ College of Education/ Hamdaniyah University

ABSTRACT
A digital image could be manipulated with many effects. One effect that can be done to an image is by decreasing the number of colors. Thresholding algorithm is one of the image processing techniques that can be used to decrease the amount of colors with similar value and grouping their intensity. In this paper, modification of thresholding algorithm was implemented on BMP (bitmap) image; the programming language Matlab version 11 was used; an image with the color component RGB (Red-Green-blue) is processed and altered from 1 up to 10. The yielded color resolution is between 1 – 1000 color. The intensity distance can be valued 0 up to 255. In general, the results can be obtained if the component of RGB equals 1, hence yielding a one color image. However, the thresholding process at cartoon image between the original image and the result would not be seen so different because images differ from one another.

INTRODUCTION
The world digital image is very closely related to the computer. With the help of our computer digital images could be processed in such a way so that the pictures are more beautiful and interesting.

Pictures taken with a digital camera can easily be transferred to a computer. If the image is already in digital form, we can manipulate the image so that the image can be changed as cross stitch. There are several techniques for image processing, filtering, smearing, thresholding, among many others. With these techniques an image can be processed to produce the image as desired. One of the image processing techniques that can be used is thresholding. This technique, allows one to manipulate the image so that the resulting image has a less colors when compared to the original image (Prosise, J., 1994).

2. Related Work:
Nowadays more researchers have touched on this field and some attempts have been presented.

Nikos, P. present an approach for a new algorithm for the reduction of the number of colors in an image. The proposed adaptive color reduction (ACR) technique achieves color reduction using a tree clustering procedure. In each node of the tree, a self-organized neural network classifier (NNC) is used which is fed by image color values and additional local spatial features. The NNC consists of a principal component analyzer (PCA) and a Kohonen self-organized feature map (SOFM) neural network (NN). The output neurons of the NNC define the color classes for each node. The final image not only has the dominant image colors, but its texture also approaches the image local characteristics used. Using the adaptive procedure and different local features for each level of the tree, the initial color classes can be split even more. For better classification, split and merging conditions are used in order to define if color classes must be split or merged. To speed up the entire algorithm and reduce memory requirements, a fractal scanning subsampling technique is used. The method is independent of the color scheme, it is applicable to any type of color images, and it can be easily
modified to accommodate any type of spatial features and any type of tree structure. Several experimental and comparative results, exhibiting the performance of the proposed technique, are presented. Index Terms—Color quantization, color segmentation, neural networks (NN), principal component analyzer (PCA), self-organized feature map (SOFM) (Nikos, P., 2002).

Papamarkos et al. present an approach for one of the most frequently used methods in image processing is thresholding. This can be a highly efficient means of aiding the interpretation of images. A new technique suitable for segmenting both gray-level and color images is presented in this paper. The proposed approach is a multithresholding technique implemented by a Principal Component Analyzer (PCA) and a Kohonen Self-Organized Feature Map (SOFM) neural network. To speed up the entire multithresholding algorithm and reduce the memory requirements, a sub-sampling technique can be used. Several experimental and comparative results exhibiting the performance of the proposed technique are presented (Papamarkos, N., 1999).

Multithresholding is one of the most powerful techniques for image segmentation. The application of multithresholding techniques is based on the assumption that object and background pixels in a digital image can be distinguished by their gray-level or color values. The proposed multithresholding technique can be applied to gray-level images as well as to color images (Papamarkos, N., 1999).

Color is one of the most important properties which humans use for object discrimination. Color is a component, which adds a new dimension to machine vision, and it has been overlooked in the past. Limitations of possible applications of color machine vision have been associated with high cost and low processing of added information. Recent progress, however, in the microelectronics industry resulted in tackling, partially, some of these difficulties, and a limited, but increasing, number of systems that utilize color information have been reported (Ohta, Y., 1980; Schettini, R., 1994). Researchers have proposed various color image segmentation techniques. Ohta et al. (1980) applied the Karhunen–Loeve transform to the color images to derive color features with large discriminating power. Connah and Fishbourne (Connah, O.M., C.A. Fishbourne, 1981) reported an algorithm based on the histogram analysis of color gray-level distribution. Huntsberger and Delxazi (1985) reported a color edge detector, which uses fuzzy set theory. Barth et al. (1986) presented a microcomputer color-based vision system to inspect integrated circuits. Tomimaga (1990) described a color classification method based on the Lab uniform color space. Representative colors are classified according to a color distance criterion. Jordan and Bovik (1988) described an investigation of the utilization of chromatic information in solving stereo vision problems. Valchos and Constantinides (1993) proposed an algorithm based on color and graph-theory, most suitable for applications such as broadcasting and pictorial information retrieval. Schettini (1994) presented a color image segmentation based on recursive 1D histogram analysis. The resulting regions are merged on the basis of a criterion, which takes into account color similarity and spatial proximity.

Segmentation by multithresholding started many years ago from simple beginnings, and in recent years has been refined into a set of mature procedures (Sahoo, P.K., 1988; Sahoo, P., 1997). The outstanding problem is how to devise an automatic procedure for determining the optimal thresholds. The various techniques can be classified into three categories. Methods based on edge matching and classification belong to the first category (Wang, S., R.M. Haralick, 1984; Hertz, L., R.W. Schaefer, 1988).

In the second category belong the histogram based multithreshold techniques (Reddi, S.S., 1984; Otsu, N., 1979). These techniques are based on different criteria, such as the minimum entropy, the interclass variance between dark and bright regions, and changes of zero-crossing and curve-fitting.

Finally, in the third category belong all the other techniques, which are usually characterized as hybrid. Extension of the aforementioned techniques to multispectral images and, in particular, to color images is not a trivial task. The increasing complexity and the structure of natural color scenes are the main reasons. Furthermore, a threshold technique must be fast and suitable for real-time applications since the basic concepts were, after all, to save computational time and memory.

Also Kevin, A. reports that the Color depth reduction is a common operation in the processing of image data, particularly for image file format conversion or for mapping images to display hardware. A problem of particular interest is the mapping of 24-bit deep RGB data to an 8-bit indexed color model i.e. how to choose 256 (or less) representative colors from a set of up to 2^32 (16.8 million) colors. Solutions to this problem range from the generation of an optimal palette, which is a time-consuming process, to the fast but approximate method of mapping to a regular grid i.e. a color cube. In this paper we develop a practical wavelet-based thresholding algorithm to compute a near optimal palette in close to real time (Kevin A.,). Konstantinos et al. reports that the color reduction in digital images is an active research area in digital image processing. In many applications such as image segmentation, analysis, compression and transmission, it is preferable to have images with a limited number of colors. In this paper, a color clustering technique which is a combination of a Kohonen Self Organized Featured Map (KSOFM) and a fuzzy clustering algorithm is proposed. Initially, we reduce the number of image’s colors by using a KSOFM. Then, using the KSOFM color clustering results as starting values, we obtain the final colors by a Gustafson-Kessel Fuzzy Classifier (GKFC).
Doing this, we lead to better color classification results because the final color classes obtained are not spherical (Konstantinos, Z., 2008).

Scheunders, P. also report that the color of the digital images is one of the most widely used information for the image processing researchers. Digital images are usually described by a set of pixels uniformly distributed in a two-dimensional grid. In color images each color is expressed by a vector containing the values of three color components. True-type color images consist of more than 16 million different colors, in a 24-bit RGB color space. However, in many applications, such as image segmentation, analysis, compression and transmission, it is preferable to have images with a limited number of colors (Scheunders, P., 1997).

3. Thresholding:

A grayscale image (also referred to as intensity image) can be converted to binary image by a process called thresholding. Thresholding also is a simple technique for denoising signals and images (Nex Robotics Pvt. Ltd.).

The purpose of the thresholding is to reduce the number of colors used in an image; but it also shifts the intensity of ribbons on the histogram; the image to another location is desired as well as change in the resolution of the color intensity. This method is possible to determine the minimum and maximum intensity value and the number of colors or the color intensity resolution. The minimum intensity is the smallest color values in an image and the maximum intensity is at its greatest value. To be noted the basis of thresholding process is carried out by means of grouping the colors that have intensities that are similar so that the number of colors needed by such image be reduced.

4. Design Process:

This scheme covers the design of the open file, processing the file, displaying the results of the process, showing the histogram and finally saving the image processed file. The design of general process is as follows:

1. Open the image file or pick up. Image files that are used must be in the form of a bitmap image, with the inner 24-bit color.
2. Thresholding process is preceded by calculating the value of each pixel, and then specifying the range of color intensity. (outMin for the lowest intensity and the highest intensity for outMax, whereas the intensity of original color image search includes: the lowest intensity inMin = and = the highest intensity inMax) specifying the number of colors or resolution will be used for the new image. The color intensity slider the whole point as far as the inMin lowest intensity, so that is the position of zero.

Then this is followed by scaling or determining the distance between the intensity of the colors for the new image, scale = (outmax – outmax)/(inmax-inmin). Scaling to say that switching to a new system of intensity. ColorperRes = (outmax – outmin)/resolutions to shift to a new position using the calculation: value = new [round (result/colorperRes)] * colorperRes after the new value is obtained, then the value is put in the new image.
3. Intensity of color differences can also be seen as the result of reading histogram.
4. Results of thresholding image can be saved.

4.1 Thresholding Algorithm:

The design process in a detail thresholding algorithm:

1. Measure the height of the image:
   for y: = 0 to Bmp. Height-1 do
2. Read the value of the pixels horizontally:
   row: = Bmp. ScanLine [y];
3. Enter the respective values in the variables:
   r: = row [x]. rgbtRed; (for the color Red)
   g: = row [x]. rgbtGreen; (for the color Green)
   b: = row [x]. rgbtBlue; (for the color Blue).

4.2 Modification of Thresholding Algorithm:

The detailed explanation of the modification algorithm is as below:

1. Finding the minimum and the maximum value
   [read value of pixels vertically from the starting point 0]
   for y: = 0 to Bmp. Height-1 do
   [read value of pixels horizontally from a starting point of 0]
   row: = iBmp. ScanLine [y];
   [search the minimum value of each color component]
   if (inMin. rgbtBlue > Res. rgbtBlue) then rgbtBlue: = inMin. Res. rgbtBlue;
   if (inMin. rgbtGreen > Res. rgbtGreen) then rgbtGreen: = inMin. Res.rgbtGreen;
   if (inMin. rgbtRed > Res. rgbtRed) then rgbtRed: = inMin. Res. rgbtRed;
[search the maximum values of each color component]
if (inMax. rgbtBlue < rgbtBlue Res.) then rgbtBlue: = inMax. Res. rgbtBlue;
if (inMax. rgbtGreen < rgbtGreen Res.) then rgbtGreen: = inMax. Res. rgbtGreen;
if (inMax. rgbtRed < rgbtRed Res.) then rgbtRed: = inMax. Res. rgbtRed;

2. **Determine the value scale (scale):**

\[ \text{scale} = \frac{(\text{outmax} - \text{outmin})}{(\text{inmax} - \text{inmin})} \]

if (inMax. rgbtBlue = inMin. rgbtBlue) then scale. rgbtBlue: = 1 else
scale. rgbtBlue: = \frac{(\text{outMax} - \text{outMin})}{(\text{inMax} - \text{inMin}) \cdot \text{rgbtBlue.. rgbtBlue}};
if (inMax. rgbtGreen = inMin. rgbtGreen) then scale. rgbtGreen: = 1 else
scale. rgbtGreen: = \frac{(\text{outMax} - \text{outMin})}{(\text{inMax} - \text{inMin}) \cdot \text{rgbtGreen.. rgbtGreen}};
if (inMax. rgbtRed = inMin. rgbtRed) then scale. rgbtRed: = 1 else
scale. rgbtRed: = \frac{(\text{outMax} - \text{outMin})}{(\text{inMax} - \text{inMin}) \cdot \text{rgbtRed.. rgbtRed}};

3. **Specify the ColorPerRes:**

\[ \text{cpres} = \frac{(\text{inmax} - \text{inmin})}{\text{res}} \]

if resR > 0 then cpres. rgbtRed: = (\text{InMax} - \text{InMin} \cdot \text{rgbtBlue.. rgbtBlue}) / res else cpres. rgbtRed: = 0;
if 0 then cpres > resG. rgbtGreen: = (\text{InMax} - \text{InMin} \cdot \text{rgbtGreen.. rgbtGreen}) / resG else
cpres. rgbtGreen: = 0;
if resB > 0 then cpres. rgbtBlue: = (\text{InMax} - \text{InMin} \cdot \text{rgbtRed.. rgbtRed}) / resB else cpres. rgbtBlue: = 0;

4. **Enter new value of intensity:**

[shift pointer vertically]
for y: = 0 to iBmp. Height-1 do
[slide the pointer horizontally, as well as to read the value of pixels]
row: = iBmp. ScanLine [y];
outRow: = oBmp. ScanLine [y];
for x: = 0 to iBmp. Width-1 do
[\text{resX} = \text{rated value of the pixels are read}]
res. rgbtBlue: = row [x]. rgbtBlue;
res. rgbtGreen: = row [x]. rgbtGreen;
res. rgbtRed: = row [x]. rgbtRed;
[rated round \text{resX} = \{(\text{resX-inmin}) \cdot \text{scale/cpresX} \} \cdot \text{cpresX}]
if (resB > 0) and (cpres. rgbtBlue < > 0) then res. rgbtBlue: =
round \{(\text{Res. rgbtBlue-inMin. rgbtBlue}) \cdot \text{scale. rgbtBlue/cpres; rgbtBlue) \} \cdot \} \cdot \text{cpres. rgbtBlue)
else res. rgbtBlue: = 0;
if (resG > 0) and (cpres. rgbtGreen < > 0) then res. rgbtGreen: =
round \{(\text{Res. rgbtGreen-inMin. rgbtGreen}) \} \cdot \text{scale. rgbtGreen/cpres; rgbtGreen) \} \cdot \text{cpres. rgbtGreen)
else res. rgbtGreen: = 0;
if (resR > 0) and (cpres. rgbtRed < > 0) then res. rgbtRed: =
round \{(\text{Res. rgbtRed-inMin. rgbtRed}) \} \cdot \text{scale. rgbtRed/cpres; rgbtRed) \} \cdot \text{cpres. rgbtRed)
else res. rgbtRed: = 0;
if (Res. rgbtBlue > 255) then Res. rgbtBlue: = 255;
if (Res. rgbtGreen > 255) then Res. rgbtGreen: = 255;
if (Res. rgbtRed > 255) then Res. rgbtRed: = 255;
[the value of the output pixel = round (resX outMin)]
outRow [x]. rgbtBlue: = round (Res. rgbtBlue outMin);
outRow [x]. rgbtGreen: = round (Res. rgbtGreen outMin);
outRow [x]. rgbtRed: = round (Res. rgbtRed outMin);
end;
end;

The flowchart modification of thresholding algorithm is shown in Figure (1). Outmin and outmax have been mentioned in paragraph 4.

5. **Experimental Result:**

After file image is open, the image that will be processed would be placed to the left side. And several switches that during early stages were not active, now have been activated. The switch or orders that are already
active are ComboBox color red, green, a blue Range, output minimal and the maximum, and the order thresholding.

![Flowchart modification of thresholding algorithm.](image1)

**Fig. 1:** Flowchart modification of thresholding algorithm.

![Image ready to process.](image2)

**Fig. 2:** Image ready to process.

The following step specifies the combination of the colours that will be used: red, green, and blue. The number of combinations for the colour equals 0-10. After that the range output was determined by user with chose minimum (min) and the maximum (Max). The number of range is between 0-255.

Havin given the ComboBox for the distance of the intensity the value from 0 up to 255. Values initially all ComboBox was given by the value = 0. The value of the colour = 0 meant not to have the colour, but the colour 0 = the black colour. So if the red colour that was chosen 2, then will use 3 red values of the colour, i.e. 0.1, and 2.

The picture in Figure (2), is ready to carry out of thresholding. The value of the colour put in the combination was as follows: the number of colours that combined red = 3 (meant 4 values of the colour), the number of colours that combined blue = 2, (meant 3 values of the colour) the number of colours that combined green = 3, (meant 4 values of the colour) the distance of the intensity, that was used for the new image: outmin = 0 and outmax = 255. Results that will be received could be seen in Figure (3).

![Original image and post modification thresholding image.](image3)

**Fig. 3:** (a) Original image, (b) Post of modification thresholding image.

The thresholding process reveals the following result. The decline in the colour happened a great number of times. The total number of colours of the original image is 18329, and the number of colours of results thresholding was 27. The number of colours of results thresholding was gotten from the combination 4 value of

The histogram graph below shows the spreading, thought to be the value of the colour that was used.
Fig. 4: Histogram original image.

Figure (4) shows the value of each flat color, and full of color intensity from 0 to 255. The histogram graph after thresholding can be seen in the Figure (5):

Fig. 5: Histogram result of image thresholding.

In Figure (5), only three intensities of color value are seen; but if we see more details, by activating a component colors, then we see how the colors are used.

Fig. 6: Component red Histogram of thresholding result.

In Figure (6), the red color component that is used there has four values, namely: 0, between 50 and 100, between 150 and 200, the last between 250 and 255. In Figure (7), the green color is 3, namely 0, between 100 and 150 and between 250 and 255. In Figure (8) the blue color occupies the fourth position: 0, between 50 and 100, Among 150 and 200, the last between 250 and 255.

Fig. 7: Component green Histogram of thresholding result.
5.1 Comparisons Between the Image Pixel size and the Number of Colors a lot:

What happens if you use a larger image of the image was in though? Earlier we used images with a resolution of 200 x 150 pixels, we now use images with size 800 x 600 pixels. The results obtained in Figure (9) as follows:

Fig. 8: Component blue Histogram of thresholding result.

The result of thresholding process with a combination of RGB values equal to the first image and the intensity range 0-255, initial image has a color number 115,079, reduced to 37 colors. Smoothness of the image is also reduced, but the color difference from the original image is less visible, because the number of pixels of the original image is quite a lot, so the image still shows slightly visible smoothness. Unlike the first image that we have processed, the image size 200 x 150 pixels, causing after thresholding images looks rough. The histogram for this second image shows that the number of color combinations that are used together with the first image.

Fig. 9: Thresholging result for image size 800x600 pixels.

Fig. 10: Histogram original image.

Fig. 11: Histogram result of image thresholding.
5.2 Difference in the Results of Thresholding with Value Change in Distance of Intensity Output:

1. Distance of intensity 0-100:

In the first experiment, we used the distance of the intensity 0-100; the results obtained are shown in Figure (12).

![Image](thresholding_result_0_100.png)

**Fig. 12:** Thresholding result with inMin=0; inMax=100.

Diagram histogram that was produced shown in Figure (13):

![Image](histogram_thresholding_result_0_100.png)

**Fig. 13:** Histogram thresholding result with inMin=0; inMax=100.

In this diagram, there is only the color intensity histogram at the value 0 and ±125. intensity, the maximum shifts slightly from the value entered, but the maximum value of all the color components equal.

2. Distance of intensity 100 – 200:

At the same image, we did the thresholding with minimum intensity value 100 and maximum intensity 200, as shown in Figure (14).

![Image](thresholding_result_100_200.png)

**Fig. 14:** Thresholding result with inMin=100; inMax=200.

Figure (14) shows that the result of image intensity thresholding with a distance of 100-200, resulted in a brighter image. This happened because the RGB color components were at the intensity value above 100. More precisely, the position of the component values RGB can be viewed on the histogram in Figure (15):

The minimum value of each color component is at position 100, and maximum 230.

3. Distance of intensity 200-255:

The image produced by using the distance of the intensity of 200-255 is the image with a bright color. This can be clarified by looking at the histogram diagram, the resulting value of the color components is at a value of...
200, as shown in Figure (16).

![Histogram thresholding result with inMin=100; inMax=200.](image)

**Fig. 15:** Histogram thresholding result with inMin=100; inMax=200.

![Thresholding result with inMin=200; inMax=255.](image)

**Fig. 16:** Thresholding result with inMin=200; inMax=255.

The image that was produced in Figure (16) by using the distance of the intensity 200-255 was the image with one clear colour. This could be clarified by the diagram histogram Figure (17) produced at the value of the colour component in one value that was 200.

![Histogram thresholding result with inMin=200; inMax=255.](image)

**Fig. 17:** Histogram thresholding result with inMin=200; inMax=255.

### 5.3 Comparison of Results of Tests on Image Thresholding:

Today will be thresholding to many images. The images have the size and number of different colors, but everything is truecolor image. The used in all image is the same.

- Red color component = 2
- Green color component = 2
- Blue color components = 2
- OutMin = 0
- OutMax = 255

*From the results of thresholding the following results can be detected:*

1. Image1.bmp with (1024 x 768 pixels) amount of original image color equals 67.764; the number of color image thresholding results equals 20 as shown in Figure (18).
2. Image2.bmp with (352 x 288 pixels) the number of colours of the original image is equals 34.687 and the number colours of the image produced is equals 17, as shown in Figure (21) below.
3. Image3.bmp with (739 x 487 pixels) amount of original image color is equals 16.277, and the number of
colours in the image produced is equals 11 shown in Figure (24).
4. Image4.bmp with (640 x 480 pixels) amount of original image color is equals 95.943, and the number of colours in the image produced is equals 20 shown in Figure (27).

Fig. 18: Thresholding result for image1.bmp.

Fig. 19: Histogram original image1.

Fig. 20: Histogram thresholding result for image1.

Fig. 21: Thresholding result for image2.bmp.
Fig. 22: Histogram original image2.

Fig. 23: Histogram thresholding result for image2.

Fig. 24: Thresholding result for image3.bmp.

Fig. 25: Histogram original image3.

Figure (26). Histogram thresholding result for image3.
5. Image5.bmp with (275 x 300 pixels) amount of original image color is equals 14.614, and the number of colours in the image produced is equals 19 shown in Figure (30).

6. Image6.bmp with (75 x 110 pixels) amount of original image color is equals 6.668, and the number of colours in the image produced is equals 11 shown in Figure (33).

7. **Findings:**

   This modification thresholding algorithm has not been used in reducing colours. It was used in many other fields. Then from the experimental results, the following differences are detected:
Fig. 31: Histogram original image5.

Figure (32). Histogram thresholding result for image5.

Fig. 33: Thresholding result for image6.bmp.

Fig. 34: Histogram original image6.

Fig. 35: Histogram thresholding result for image6.
a. Small image, after the thresholding appears to result gives a very clear effect of criticism. This causes subtle color gradations and regrouping into one or two colors only seen in Figure (33).
b. Great image, thresholding results show that the color is reduced and in some pictures, colors looks rough and in cluster.
c. Cartoon image, the image dose not show very visible difference between images before and after thresholding. This is due to the fact that the spread of color is not as much as the image of the photo. Images in (Figure (18), and Figure (30) the images thresholding result is not so much different; and this is indicated in the histogram of original image and histogram thresholding results.
d. But there are similarities in these images; that image thresholding result was a reduction of color is quite significant.
e. Histogram thresholding results show that almost all have the same pattern; the colors are clustered on the left side, right side and also in the middle.

6.1 Conclusion:
From the results the researchers conclude the following:
1. If thresholding technique is applied on cartoon image or image color with less diversity; less difference from the original image will be detected; whereas if applied on the photo image with many colours, the image will be distorted because of the reduction of colors.
2. If the distance output intensity is closer to the point zero, it gives a dark image effects; while the output intensity ranging to pixel 255 provides a bright image effects.
3. In some of the post thresholding images there is no real deformation. In fact, the amounts of colours which have been lessened led to a sort of blurring images. Each image has its own features.

REFERENCES
Papamarkos, N., C. Strouthopoulos and LA ndreadis, 1999. Multithresholding of Color and gray-level Images through a Neural Network Technique, Electric Circuits Analysis Laboratory, Department of Electrical and Computer Engineering, Democritus University of Thrace, 23.
Kevin A., A Fast Wavelet algorithm for the Reduction of Color Information, Assistant Professor in Information Technology, Department of Civil & Environmental Engineering, MIT.
Konstantinos, Z., P. Nikos, and K. Ioannis, 2008. Color Reduction using the Combination of the Kohonen Self-Organized Feature Map and the Gustafson-Kessel Fuzzy Algorithm Image Processing and Multimedia Laboratory Department of Electrical & Computer Engineering Democritus University of Thrace 67100 Xanthi, Greece, papamark@ee.duth.gr.