International Refereed Journal of Engineering and Science (IRJES)
ISSN (Online) 2319-183X, (Print) 2319-1821
Volume 2, Issue 7 (July 2013), PP. 59-73
www.irjes.com

Application of Matlab in Weather Satellite Images

Abdullah M. Dobaie[1], Mubashshir Husain[2]
Electrical and Computer Engineering Department, King Abdul Aziz University P.O. BOX: 80204, Jeddah 21589

ABSTRACT:- *Weather images are taken these days by satellites for all parts of the earth. These images are used by weathermen all around the world for weather analysis and predictions. Weather Images are mainly taken from high above and are generally black and white or gray images. The technology revolution that we are enjoying today along with the information highways made available to the public through computers and internet many of these weather images. As the general public are not quite experienced with the weather images and reading them it becomes necessary to make these images easier to read and get information from. The different types of weather images that we have these days made it even harder and more confusing to the public to read an image and get information out of them.*

The aim of this paper is to develop a technique that enhances weather image qualities and removes any noise that could exit on an image by using matlab. And also to develop a coloring technique in order for the public to enjoy a colorful image where water is in blue, land in green or brown, and clouds are in white or more than one color to reflect the different cloud concentrations.

I. INTRODUCTION

1.1 Weather Satellites

A weather satellite is a type of artificial satellite that is mainly used to moniter the weather and climate of the Earth. Weather satellites play an essential role in weather forecasting all around the world. National Oceanic and Atmospheric Administration started launching weather satellites over than 40 years ago. Over the years basic, old weather satellites were retired and replaced by new and advanced satellites. Now days, many weather satellites run in orbit and cover the whole world. The main function of weather satellites is to take images of parts of the earth from high above and send it to earth for use in weather forecasting and climate monitoring. Old satellites had low tech cameras that produced low quality images. Technological advances helped improve the quality of those images and introduced new types of images that further improved climate monitoring and weather forecasting. There are two types of meteorological satellites that are used for weather forecasting, Geostationary and Polar Orbiting. Geostationary satellites orbit the earth above the equator at an altitude of 35,880 kilometres and transmite images of the entire himisphere below. Such satellites rotate with the earth and stay stationary with respect to it.

Polar orbiting satellites circle the earth contineously in a north to south flight or vice versa at altitudes of 720 - 800 km above the surface of the earth. If the satellite is left at such orbits without movement, it will come crashing down to earth due to gravity force. To maintain polar satellites at lower altitudes they need to run around the earth faster than the earth rotates around itself. At a certain speed (greater than the speed of earth rotation) the centrefugal force equals the earth gravity force and the satellite stays in orbit at the same altitude. This explains why polar satellites keep circling the earth and take pictures of different parts of the world. Polar satellites take picture of the same location once every day and night as they pass over it. Pictures taken by Polar satellites show a closer and more detailed look at the region it is covering.

1.2 Communication with Weather Satellites

The weather satellite imagery is down-linked to earth in two major frequency bands. The low resolution imagery is transmitted in the VHF (137MHz) frequency range. The high resolution imagery is transmitted on 1698 and 1707 MHz. Down-converters for these frequencies are typically used to bring the signal down to 137 MHz receiver. There are a number of commercially manufactured receivers designed specifically to receive weather satellite signals.

1.3 Types of Weather Images

Weather images started as low quality visible images taken by cameras mounted on weather satellites. These days visible images are still being used for weather forecasting. However Technological advances lead for better, high quality visible images through the use of better lenses, digital processing and improved filters. A sample visible image is shown in (figure 1).

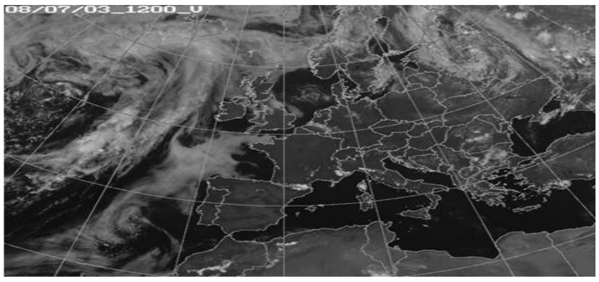


Figure 1: Visible Image of Europe.

Different types of imaging were also introduced for use in weather forecasting through the use of water vapor sensors and Thermal or Infrared sensors.

Thermal or infrared images recorded by the weather satellite sensors enable a trained analyst to determine cloud heights and types, to calculate land and surface water temperatures, and to locate ocean surface features. Infrared pictures depict offshore pollution and map currents such as the Gulf Stream valuable to the shipping industry hoping to save precious fuel by using the ocean movements wisely. A sample Image taken by Infrared cameras is shown in (figure 2).

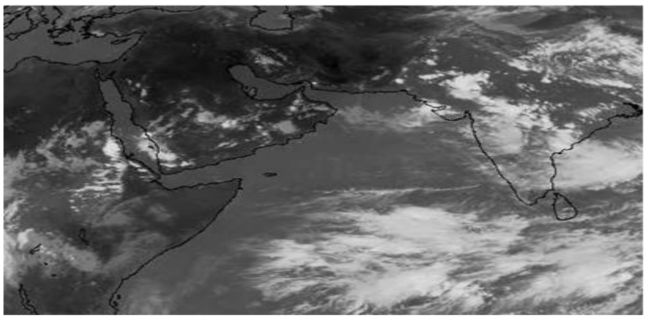


Figure 2: Infrared Image of Saudi Arabia.

Water vapor sensors produce images of the water vapor concentration over the area of earth being covered. This helps weathermen determine humidity concentrations and other climate information of the area being studied. A sample Image taken by Water vapor cameras is shown in (figure 3).

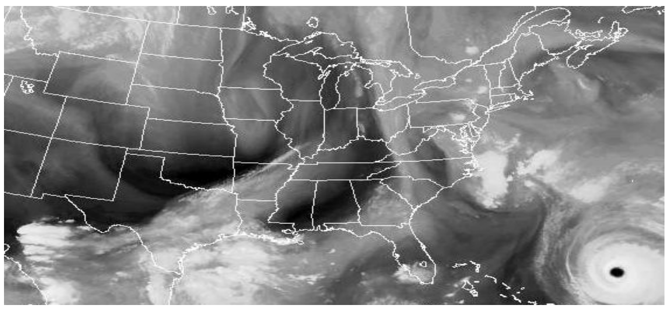


Figure 3: Water Vapor Image of the eastern USA.

All three types of images are now adays being used by weathermen all over the world for better weather forecasting and climate monitoring.

1.4 Creating images in Matlab

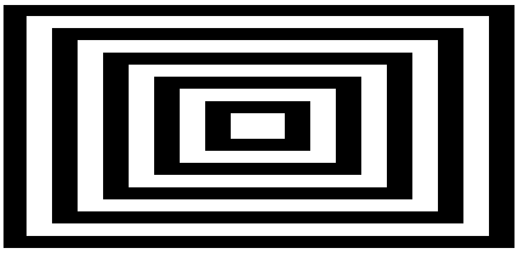
We can create a black and white (Binary) image in Matlab by creating a matrix of a certain size (m by n) with elements being either 0 or 1. When the matrix is filled it can be displayed by using the function **imshow(**

). The matrix A created is similar to the matrix shown in (figure 11) but of a size (100 by 100). A program to create the matrix and display it is shown below.

$$A=\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&0&1&0\\
0&1&0&1&0&1&0&0&0&0&0&0&0&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&1&1&1&1&1&1&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&1&0&0&0&0&1&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&1&0&0&0&0&1&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&1&1&1&1&1&1&0&1&0&1&0&1&0\\
0&1&0&1&0&1&0&0&0&0&0&0&0&0&1&0&1&0&1&0\\
0&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&0&1&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

Matrix of a Binary Image

Matrix of a Binary Image

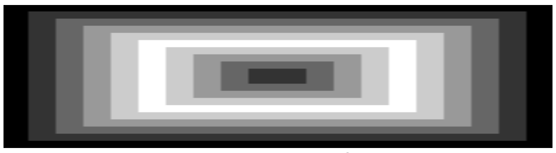


BINARY IMAGE RESULTING FROM AMTREX (A)

To create an image that has a gray scale rather than just black and white. The values of the elements in the matrix need to be any real number between 0 and 1 similar to the matrix in figure 13. We use the function imshow( ) to display the matrix as an image. The program used to create the matrix and display the image is shown below. The resulting image is shown in (figure 14).

$$B=\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0\\
0&0.2&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.2&0\\
0&0.2&0.4&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&1&1&1&1&1&1&1&1&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.6&0.6&0.6&0.6&0.6&0.6&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.6&0.4&0.4&0.4&0.4&0.6&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.6&0.4&0.2&0.2&0.4&0.6&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.6&0.4&0.2&0.2&0.4&0.6&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.6&0.4&0.4&0.4&0.4&0.6&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.6&0.6&0.6&0.6&0.6&0.6&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&1&1&1&1&1&1&1&1&1&1&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.8&0.6&0.4&0.2&0\\
0&0.2&0.4&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.6&0.4&0.2&0\\
0&0.2&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.4&0.2&0\\
0&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0.2&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

MATRIX OF A GRAY SCALED IMAGE

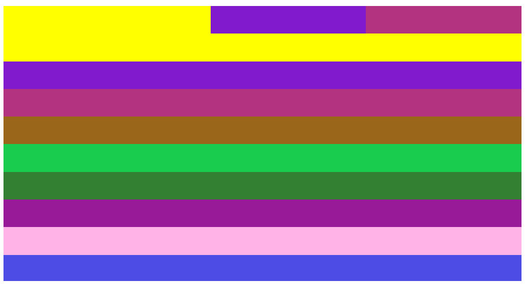


Gray scaled image resulting from matrix (B).

We can create a colored image using colored maps. We define a color map array of size (3 by n) and an index matrix with values ranging from one to n. An index matrix and a color map are shown in (figure 18). The program to create the matrix and color map is presented below. The resulting image is shown in (figure 19).

$$G = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1&1&1 \\ 2&2&2&2&2&2&2&2&2&2 \\ 3&3&3&3&3&3&3&3&3&3 \\ 4&4&4&4&4&4&4&4&4&4 \\ 5&5&5&5&5&5&5&5&5&5 \\ 6&6&6&6&6&6&6&6&6&6 \\ 7&7&7&7&7&7&7&7&7&7 \\ 8&8&8&8&8&8&8&8&8&8 \\ 9&9&9&9&9&9&9&9&9&9 \end{bmatrix} \qquad colmap = \begin{bmatrix} 0&0&0 \\ 0.5&0.1&0.8 \\ 0.7&0.2&0.5 \\ 0.6&0.4&0.1 \\ 0.1&0.8&0.3 \\ 0.2&0.5&0.2 \\ 0.6&0.1&0.6 \\ 1&0.7&0.9 \\ 0.3&0.3&0.9 \\ 0.9&0.6&0.9 \end{bmatrix}$$

Figure 18: Index Matrix and color map array.



Coloured Image using Index Matrix and Colour map

II. READING WEATHER IMAGES

The first weather image (US_Visible.bmp) is read into Matlab. The resulting array is of class unsigned integer 8 and sometimes it needs to be changed to class double for processing the array. To accomplish this class change, we can create a new array of class double that is equal to the original array in values. Also we can use the function **im2double( )**. The problem with the later one is that it will scale all array elements to new values between 0 and 1 which is not desirable in our case.

The image size is 443 pixels high by 532 pixels wide. Although the image is gray scaled the array size was found to be 443 by 532 by 3. This denotes that the image was saved as a colored RGB image rather than a gray scaled image. Checking all three arrays representing Red, Green, and blue colors we find that
the arrays have identical values. To treat the image as a gray scaled image we can simply use one of the 443 X 532 arrays to represent the image. Displaying any of the 443 X 532 arrays produces the exact same image as that was read.

Matlab program reads an RGB image and checks if the image is colored or gray scaled saved as RGB. This is done by checking the three m by n arrays if they are identical or not. The ouput is shown if fig20 & fig 21.

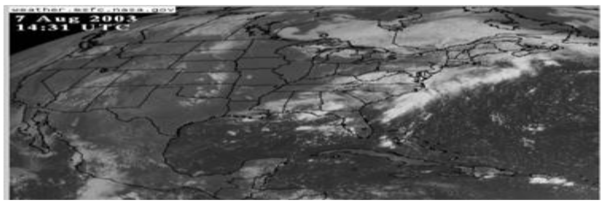
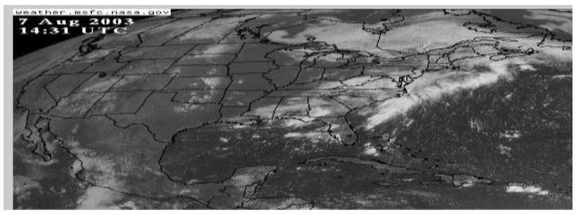
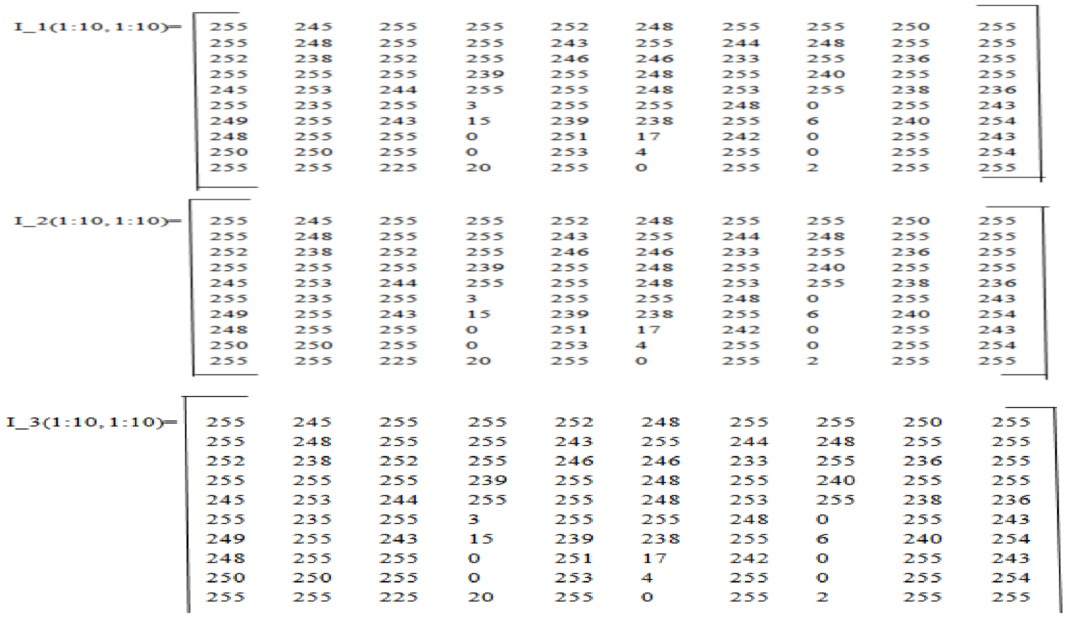


Figure 20: Original image US_Visible.bmp.

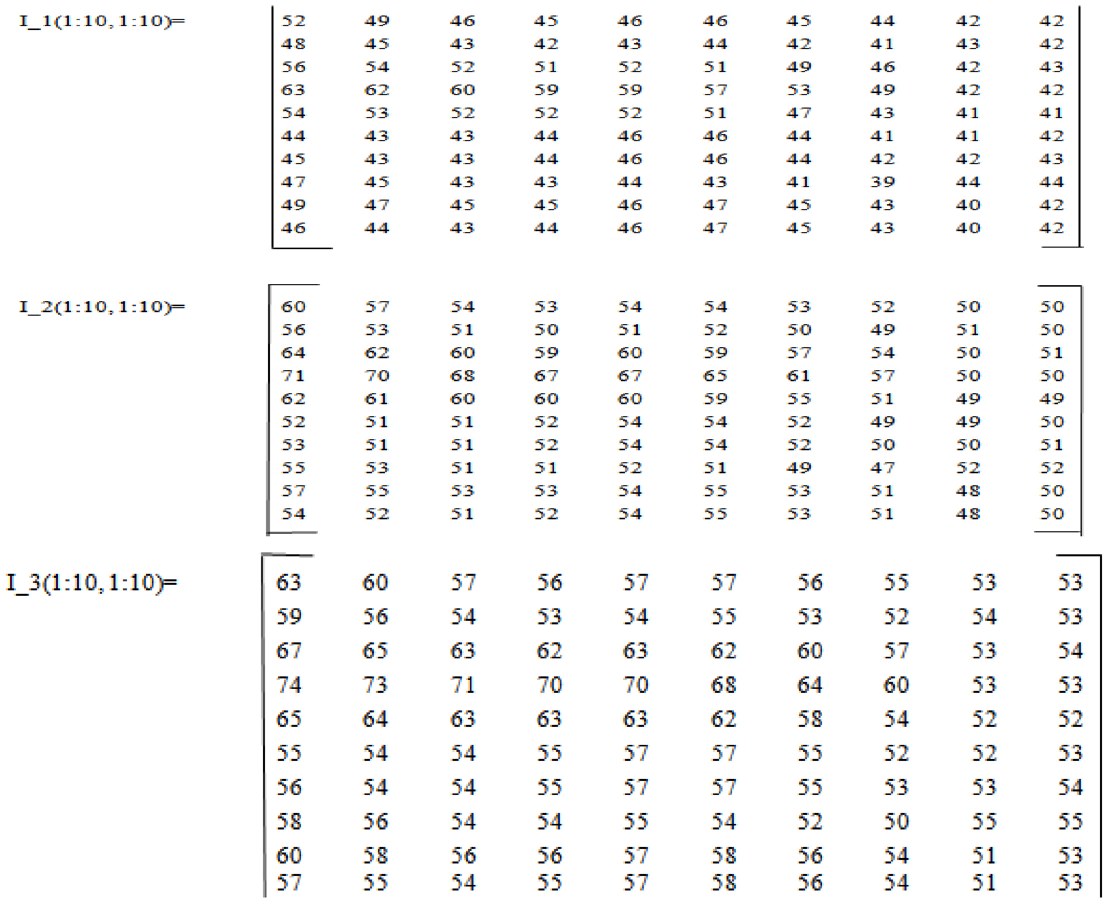




```
I_1(1:10,1:10)=  255  245  255  255  252  248  255  255  250  255
                 255  248  255  255  243  255  244  248  255  255
                 252  238  252  255  246  246  233  255  236  255
                 255  255  255  239  255  248  255  240  255  255
                 245  253  244  255  255  248  253  255  238  236
                 255  235  255    3  255  255  248    0  255  243
                 249  255  243   15  239  238  255    6  240  254
                 248  255  255    0  251   17  242    0  255  243
                 250  250  255    0  253    4  255    0  255  254
                 255  255  225   20  255    0  255    2  255  255

I_2(1:10,1:10)=  255  245  255  255  252  248  255  255  250  255
                 255  248  255  255  243  255  244  248  255  255
                 252  238  252  255  246  246  233  255  236  255
                 255  255  255  239  255  248  255  240  255  255
                 245  253  244  255  255  248  253  255  238  236
                 255  235  255    3  255  255  248    0  255  243
                 249  255  243   15  239  238  255    6  240  254
                 248  255  255    0  251   17  242    0  255  243
                 250  250  255    0  253    4  255    0  255  254
                 255  255  225   20  255    0  255    2  255  255

I_3(1:10,1:10)=  255  245  255  255  252  248  255  255  250  255
                 255  248  255  255  243  255  244  248  255  255
                 252  238  252  255  246  246  233  255  236  255
                 255  255  255  239  255  248  255  240  255  255
                 245  253  244  255  255  248  253  255  238  236
                 255  235  255    3  255  255  248    0  255  243
                 249  255  243   15  239  238  255    6  240  254
                 248  255  255    0  251   17  242    0  255  243
                 250  250  255    0  253    4  255    0  255  254
                 255  255  225   20  255    0  255    2  255  255
```

Figure 21 Image produced by displaying one of the arrays.
Figure 22: Matlab output displaying 10 X 10 part of the image array representing (US_Visible.bmp).

We can see clearly from the original and reproduced image that there is no noticeable difference whatsoever. Checking another gray scaled image (South_America.bmp) and checking the individual arrays, we find out that the three arrays have close values but different. In such a case we can't use one of the three arrays to represent the picture accurately. The output data for parts of the arrays is displayed.



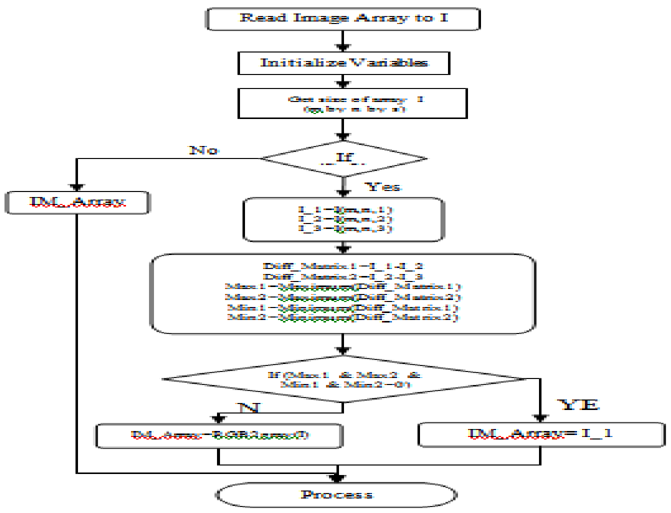
```
I_1(1:10,1:10)=     52  49  46  45  46  46  45  44  42  42
                    48  45  43  42  43  44  42  41  43  42
                    56  54  52  51  52  51  49  46  42  43
                    63  62  60  59  59  57  53  49  42  42
                    54  53  52  52  52  51  47  43  41  41
                    44  43  43  44  46  46  44  41  41  42
                    45  43  43  44  46  46  44  42  42  43
                    47  45  43  43  44  43  41  39  44  44
                    49  47  45  45  46  47  45  43  40  42
                    46  44  43  44  46  47  45  43  40  42

I_2(1:10,1:10)=     60  57  54  53  54  54  53  52  50  50
                    56  53  51  50  51  52  50  49  51  50
                    64  62  60  59  60  59  57  54  50  51
                    71  70  68  67  67  65  61  57  50  50
                    62  61  60  60  60  59  55  51  49  49
                    52  51  51  52  54  54  52  49  49  50
                    53  51  51  52  54  54  52  50  50  51
                    55  53  51  51  52  51  49  47  52  52
                    57  55  53  53  54  55  53  51  48  50
                    54  52  51  52  54  55  53  51  48  50

I_3(1:10,1:10)=     63  60  57  56  57  57  56  55  53  53
                    59  56  54  53  54  55  53  52  54  53
                    67  65  63  62  63  62  60  57  53  54
                    74  73  71  70  70  68  64  60  53  53
                    65  64  63  63  63  62  58  54  52  52
                    55  54  54  55  57  57  55  52  52  53
                    56  54  54  55  57  57  55  53  53  54
                    58  56  54  54  55  54  52  50  55  55
                    60  58  56  56  57  58  56  54  51  53
                    57  55  54  55  57  58  56  54  51  53
```

Figure 23: Matlab output displaying 10 X 10 part of the image array representing (South_America.bmp).

The difference in the arrays is clear as shown in (figure 23). In such a case we can use a function called RGB2gray( ) which converts the image array from RGB to gray. The function uses a special averaging technique to perform the conversion Thus we need to come up with a technique to automatically check if the three color arrays are identical or not. If they are identical, we use one of them to represent the image. If they are different we use the function **RGB2gray( )**. To check that all three arrays are identical, we generate a new array (Diff_Matrix1) that is equal to the difference between Array1 representing Red and

Array2 representing Green. For both to be identical all elements of the difference array (Diff_Matrix1) should be zeros. Similarly we generate another array (Diff_Matrix2) that is equal to the difference between Array2 representing Green and Array3 representing Blue. To make sure all elements of the difference arrays are zeros we obtain the maximum and minimum of all elements in the difference arrays. Then we check if the minimum and maximum are both zeros for both arrays. If that is true then the three arrays are identical and one of them can be used to represent the image. If not we will use the function **RGB2gray( )**. The following program deploys the mentioned mechanism.

The previous program changes RGB images or gray scaled images that are saved as RGB to gray scaled images. However some gray scaled images are saved in gray scale representation. Thus the previous program needs to be adjusted to check for the size of the image array that is read. If the size of the array is m by n by 3 then the array is either RBG or gray scaled image saved as RGB and the previous technique must be used to change it to gray scale representation. However if the size is m by n (m by n by 1) then the image is already in gray scale representation and there is no need to apply the previous technique. The following program deploys the mentioned mechanism and will be used throughout the paper to obtain the array representing the image to be processed.

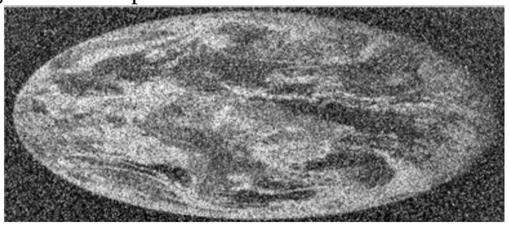


Flow Chart

III. IMAGE ENHANCEMENT

3.1 Noisy image Enhancement

Certain images can be noisy and are very unclear. A noisy weather image of the earth is shown in (figure 24). Such a weather image cannot be processed or colored due to the excess noise.



We can reduce the noise by applying a noise filter. In this paper we will use a median filter which is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. The function **medfilt2(**A,**[**m n**])** performs median filtering of the matrix A in two dimensions. Each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image. Applying the median filter function to the array produces a much clearer image.

The output image differs depending on the number of elements the filter uses to calculate each new pixel. Depending on the amount of noise that exists in the image, we can choose a filter that best fits it. In our example we can see that **medilt2(**ImArray,**[**5 5**])** produce a better image than **medilt2(**ImArray,**[**3 3**])** in terms of noise. Although this is true for noise, the detail in the image gets worse as the number of pixels that the filter uses increase. [1][2]

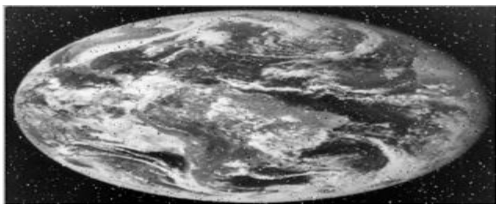


Figure 25: Resulting image after applying a medfilt2(A,[3 3]).

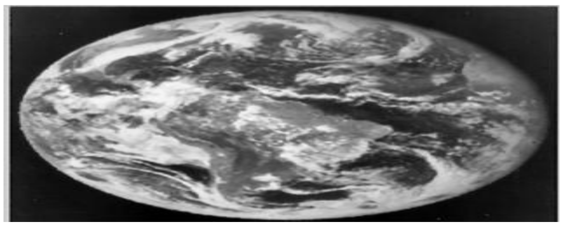


Figure 26: Resulting image after applying a medfilt2(A,[5 5]).

IV. CHANGING THE IMAGE CONTRAST

4.1 Manual Adjustment of Image Contrast

We can improve an image by change its contrast. This can be done by modifying the array that represents the image. To increase the contrast of an image we will have to scale up the values of the array representing the image. To reduce the contrast we have to scale down the values of the array. We can do this scaling manually or use a function called imadjust**(**I,**[**low_in high_in**]**,**[**low_out high_out**])**. The values of low_in and high_in are the intensities in the input image which are mapped to low_out and high_out in the output image.

For example if low_in = 0 , high_in = 0.8 and low_out = 0 , high_out =1.0, the function will map the lowest value of 0 to itself while raising the high value from 0.8 to 1.0. This will increase the contrast. If high_out = 0.5 this will decrease the contrast. The map of Europe (Europe_Visible.bmp) is read into Matlab and the contrast is adjusted.

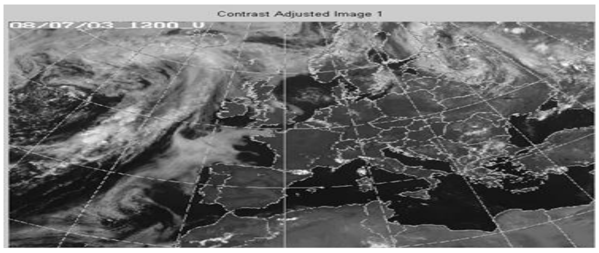


Figure 28: Europe visible image after adjustment using imadjust (ImArray,[0 .8],[0 1]).

Adjusting the contrast of another image (US_Visible.bmp) is done using the program. Different scaling values are used for adjusting this image since the values used for the previous image don't improve the contrast of this image. The (US_Visible.bmp) (figure 30) will produce the images shown in (figures 31 and 32) if we use the same scaling values as used for the previous image. The image shown in (figure 33) is produced when using a different scaling factor [0.2 0.5], [0.1 1].

V. AUTOMATIC ADJUSTMENT OF IMAGE CONTRAST

5.1 Histogram Stretching

Although we can improve the image quality manually it is desirable to find a technique that can improve the image contrast automatically without user intervention. The histogram of Europe visible image (Europe_Visible.bmp) (figure 34) is shown in (figure 35).

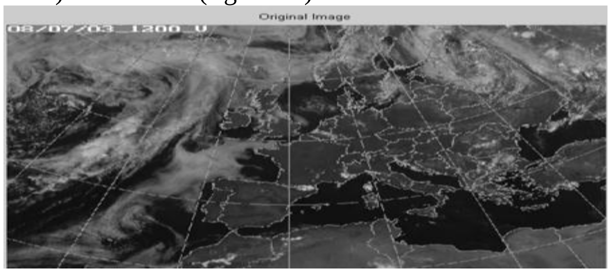


Figure 34: Europe visible image.

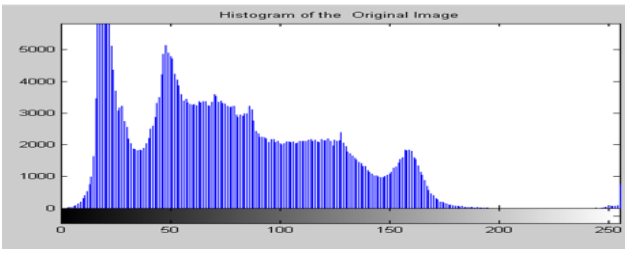


Figure 35: Histogram of the previous image showing intensity data distribution.

The histogram shows the intensity data distribution in the image. The X-axis reflects all possible intensity values. The Y-Axis reflects the count of occurrence of each intensity value. We can see that most of the data lies mainly between intensity values of 10 and 180. If we can stretch the data in this range to occupy the full range of 0 to 256 we will get a better image of a higher contrast. To accomplish this we can use the Matlab function called **stretchlim(**I**)**. The function will stretch the data to fill the whole range.

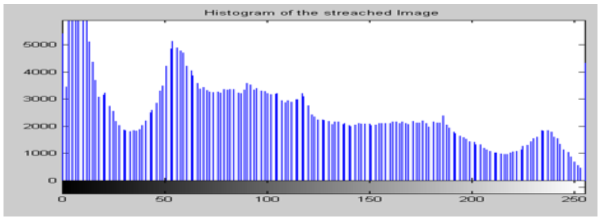


Figure 36: Histogram of the automatically stretched intensity data distribution.

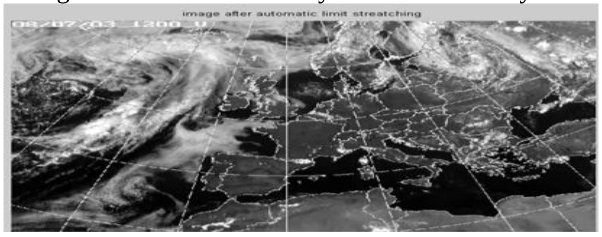


Figure 37: New image after automatic intensity data stretching.

In this example we can see some noticeable improvement in the image in the sense that we can better distinguish water from the land and clearly see the clouds. Applying the same technique to another image (US_Visible.bmp) (figure 38) we get the histogram shown in (figures 39). Stretching the intensity data we get the histogram shown in (figure 40) and the image shown in (figure 41).

VI. HISTOGRAM EQUALIZATION

Histogram Equalization is a technique that can be used for increasing the contrast of an image. Histogram Equalization enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram. If we do not specify a histogram to match, the equalization function will match the image histogram to a flat histogram. Applying the specified function to our image (Europe_Visible.bmp) is done in the program. The resulting histogram is shown in (figure 42) and the resulting image is shown in (figure 43).

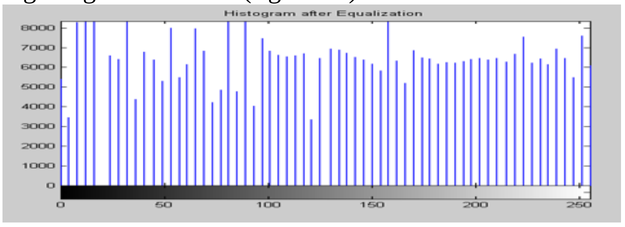


Figure 42: Histogram of Europe Visible Image after qualization.

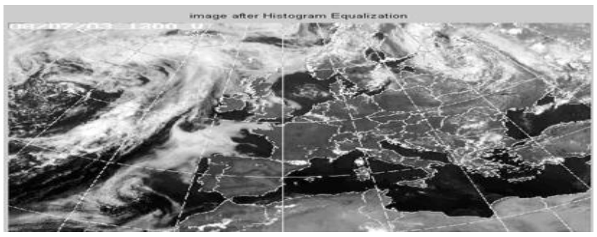


Figure 43: Europe Visible Image after Equalization.

In the resulting image we can see a much clearer picture in which we can distinguish water from land and can still see the clouds very clearly. Applying the program to the Image (US_Visible.bmp) produce the histogram shown in (figure 44) and the resulting image is shown in (figure 45).

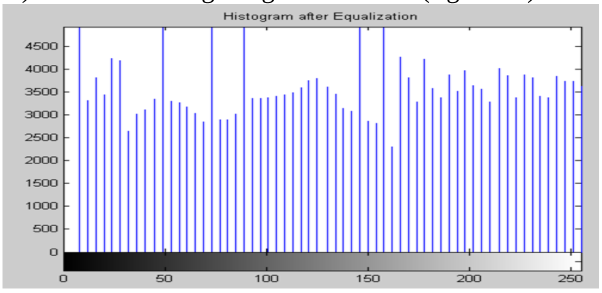


Figure 44: Histogram of US Visible Image after Equalization.

VII. GAMMA CORRECTION

Gamma Correction imadjust maps low to bottom, and high to top. By default, the values between low and high are mapped linearly to values between bottom and top. For example, the value halfway between low and high corresponds to the value halfway between bottom and top. Imadjust can accept an additional argument which specifies the gamma correction factor. Depending on the value of gamma, the mapping between values in the input and output images may be nonlinear. For example, the value halfway between low and high may map to a value either greater than or less than the value halfway between bottom and top. Gamma can be any value between 0 and infinity. If gamma is 1 (the default), the mapping is linear. If gamma is less than 1, the mapping is weighted toward higher (brighter) output values. If gamma is greater than 1, the mapping is weighted toward lower (darker) output values. The figure below illustrates this relationship. The three transformation curves show how values are mapped when gamma is less than, equal to, and greater than 1. (In each graph, the x-axis represents the intensity values in the input image, and the y-axis represents the intensity values in the output image.)

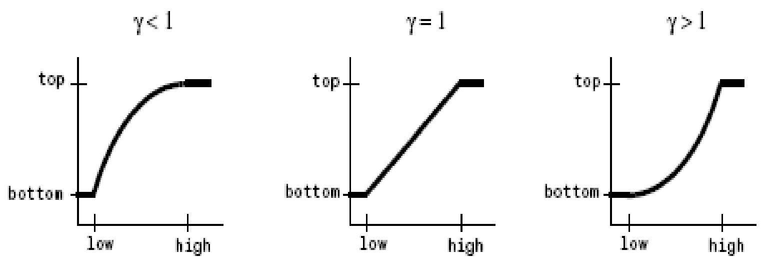


Figure 46: Plots Showing Three Different Gamma Settings.

The technique is applied to the image (Europe_Visible.bmp) using the program.
The resulting image has a histogram shown in Figure 47 and is displayed in (figure 48).

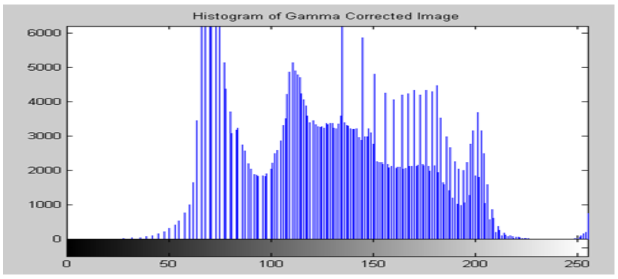


Figure 47: Histogram of Europe Visible Image after Gamma Correction.

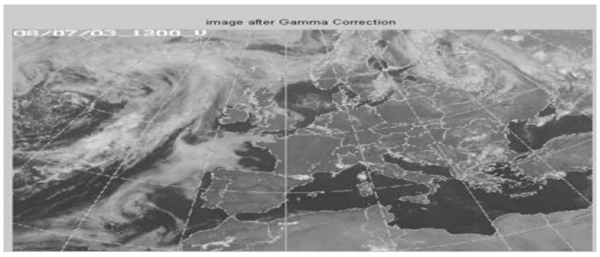


Figure 48: Europe Visible Image after Gamma Correction.

VIII. WEATHER IMAGE COLORING

8.1Coloring Visible Weather Images

In this chapter we intend to develop a technique to apply colors to gray scaled weather images. Studying different gray scaled weather images we can notice that any image consists mainly of three objects. The three objects are land, water, and clouds. Each object has an intensity concentration that is different than others. In all visible weather images clouds have high intensity values. They appear in white color or closer to white than other objects in the image. On the other hand, Water (seas and oceans) have very low intensity values. They appear in black (dark) color or darker than other objects in the image. Clouds and water in visible images represent the intensity extremes of the image. Land has an intensity that is in between the other objects. It mainly appears in a medium intensity gray color. In order for us to apply colors to a gray scaled image, we can define a color map and display the image using the color map. In order for us to use the color map we have to modify the elements of the gray scaled array to have distinct values between 0 (referring to the first color in the color map) and n (referring to the last color in the color map. We can define three colors blue, green and white to represent water, land and clouds. We can use other colors like blue, brown, and white or any other color combinations that we wish to apply to the image. In such a case the elements of the image array need to be modified to have one of three values 0, 1, or 2. In the modified array the values become indices where 0 is the index to refer to the blue color, 1 to refer for the green color and 2 to refer to white color. Considering the image (Europe_Visible.bmp) shown in (figure 51) and studying its histogram shown in (figure52), we notice that there is a concentration of intensities in the region less than 35.

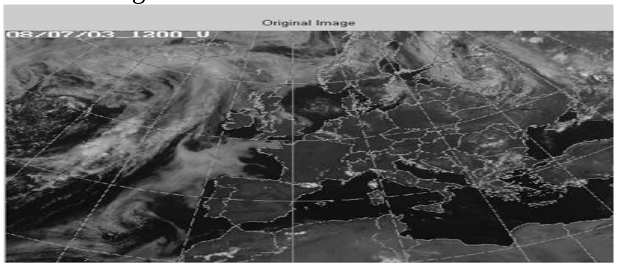


Figure 51: Image of Europe before coloring

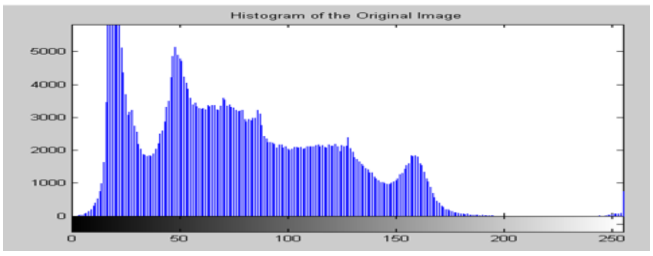


Figure 52: Histogram of the image (Europe_Visible.bmp)

This is basically the region of darkest intensities in the image that represents water. So our program should check all values in the array that are less than 35 and assign them a new value of 0 (index for the color blue). The second concentration region of intensities is between 35 and 80 roughly. This is the intermediate region that represents land in the image. Our program should assign a value of 1 (index for the color green) to all elements between 35 and 80. Anything above 80 is considered as the third concentration of intensities that represent clouds. Any element in the array that is higher than 80 is assigned a value of 2 (index for the color white). The new array is displayed with the color map as a new image..

In the program we use 4 different color combinations and the resulting images are shown in (figure 53), (figure 54), (figure 55), and (figure 56).

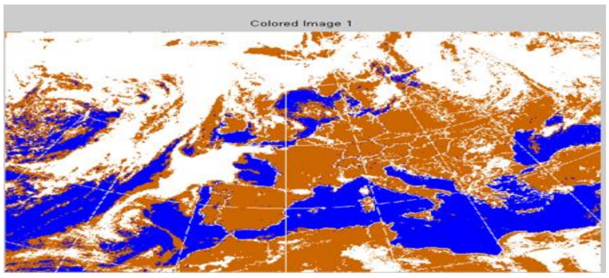


Figure 53: Europe image with the first color combination.

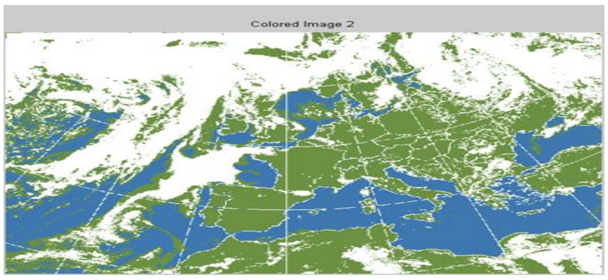


Figure 54: Europe image with the second color combination.

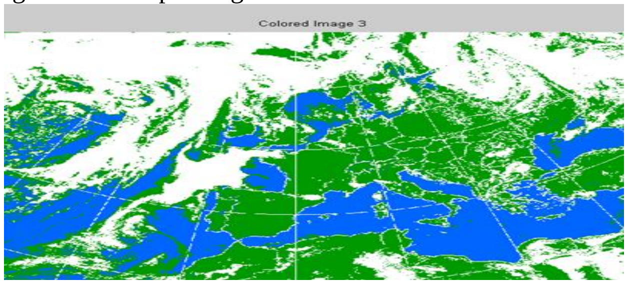


Figure 55: Europe image with the third color combination.

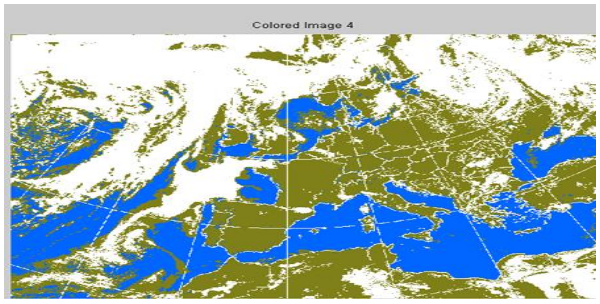


Figure 56: Europe image with the fourth color combination.

We can notice that some land in the colored map appear in areas where there is no land at all. This is because some low cloud concentrations have a medium gray scaled intensity similar to the gray scaled intensity of land as can be seen in the original image.

Applying the program to the image ('US_Visible.bmp') we can see from original image and the histogram (figure 57) and (figure 58) the histogram that the dark concentration is in the region less than 60. The medium gray scale is in the region between 60 and 100. The high intensity region is in the range above 100. The program needs to be adjusted to reflect these values. (Figures 59, 60, 61, and 62) show the resulting images after coloring.

IX. COLORING INFRARED WEATHER IMAGES

An infra-red image of the Arabian Peninsula is shown below in (figure 64). As can be noticed from the image, the dark intensity

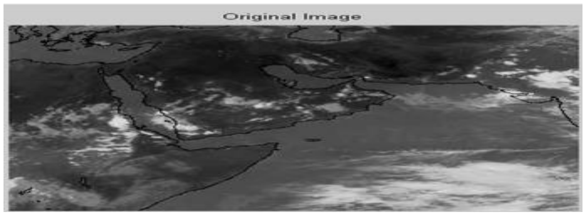


Figure 65: Original Saudi Infra Red image.

region in this case is the land, and the medium intensity gray region is the water. The histogram of this image is shown in (figure 65). Studying the histogram we can notice that the dark intensity concentration (representing land) is in the range less than 55. The medium intensity concentration (representing water) is in the region between 55 and 90. The rest of the upper region is the light intensity concentration representing clouds.

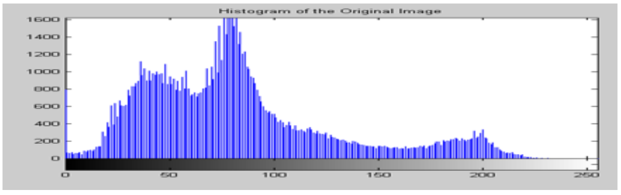


Figure 66: Histogram of Saudi Infra Red Image.

The result of the program that applies colors to Infra Red images is shown below(figure 67).

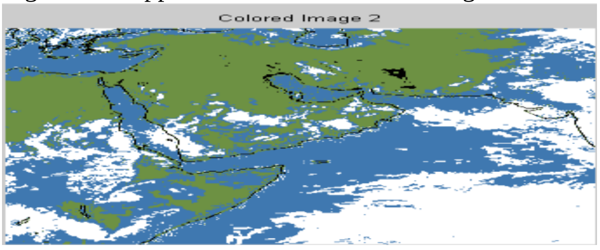


Figure 67: Infra red image with 3 color.

Applying more colors for the different cloud concentrations can also be done for infra red images. The program that applies 5 colors to the image (2 colors for land and water and 3 colors for the different cloud concentrations) is developed and the resulting image is shown in (figure 68).

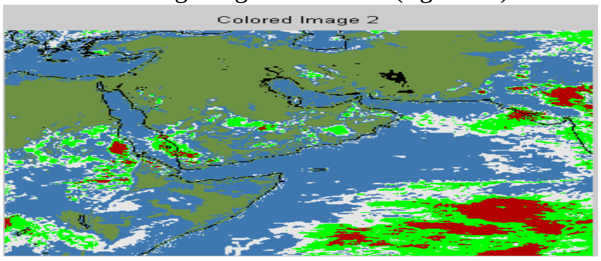


Figure 68: Saudi Infra Red Image with 5 colors.

X. INTEGRATION OF IMAGE ENHANCEMENT AND COLORING

In this chapter we integrate some of the image enhancement techniques developed previously with the coloring techniques developed in the previous chapter. The aim of this is to improve the quality of the final image. We found previously that automatic limit stretching improves the image contrast and makes the image clearer. We can apply this stretching technique to the image before we attempt to color the image. This should improve the quality of the image. It was also found that histogram equalization improves the contrast of the image considerably. Applying this enhancement technique to the image before coloring is believed to improve the colored image. Histogram equalization and limit stretching techniques improve the contrast and make the objects in the image more distinct. In terms of numbers the techniques shift the regions that represent land, water and clouds further apart from each other making it easier for us to set the regions limits for our coloring techniques. This allows for setting fixed regions for the coloring program. In such a case we don't have to change the coloring limits for different images. The program that combines all three techniques and is applied without any changes to both images (Europe and US). The resulting images are shown in (figure 69) and (figure 70).

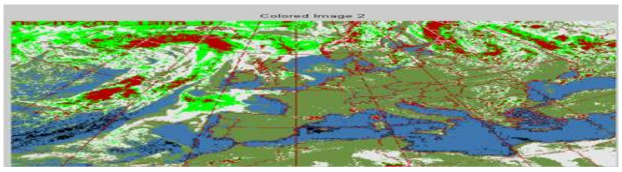


Figure 69: Europe image after enhancement and coloring with fixed limits.

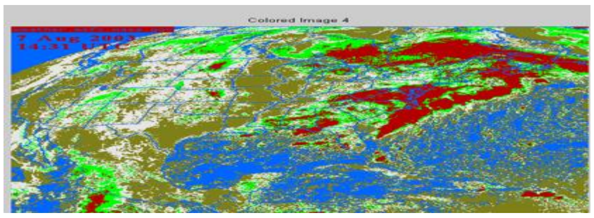


Figure 70: US Image after Enhancement and coloring with fixed limits.

Due to the big difference in the intensity and contrast between the original weather images of Europe and US, the resulting images with fixed limits are not as great. Minor changes to the coloring limits would improve the images greatly. Following is the resulting images after minor changes to the coloring limits.

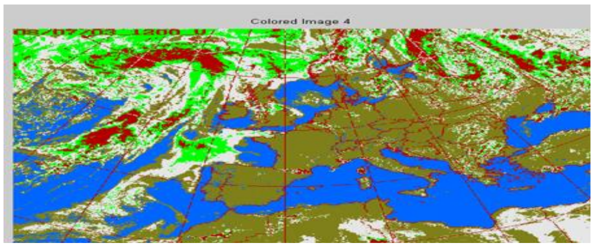


Figure 71: Europe image after enhancement and coloring.

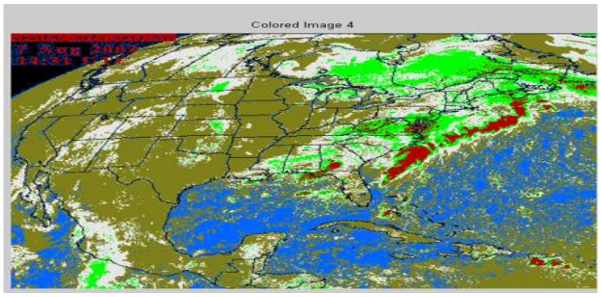


Figure 72: US image after enhancement and coloring.

This minor change wouldn't be needed when we deal with images of the same area and have similar intensities.

XI. CONCLUSION AND FUTURE DEVELOPMENT

In conclusion we studied the different types of weather images taken by different satellites orbiting the earth. The technique for changing analog images to digital images and allows for processing the images on the computer was presented. A detailed research of the different types of images and their representation in Matlab was carried out. The research included black and white images, gray scaled images, RGB images and colored images with color maps. We demonstrated producing, writing and reading of several images using Matlab.

Matlab was used to develop different programs and techniques to read and process different weather images. The visible weather image enhancement techniques developed in this paper greatly enhanced the quality and readability of each image. A noise filtering program was developed and applied to a noisy image. The filtered image was dramatically better than the original noisy image. Several enhancement techniques were also programmed and applied to different weather images. Very satisfying results were obtained by applying the different enhancement methods along with the noise filtering technique.

A coloring technique that adds colors to the image was also developed. The resulting images showed considerable improvement in visibility and readability. The technique was applied to gray scaled weather images using 3 colors and 5 colors. The method was demonstrated for visible and infra red images.
Automatic limit stretching along with histogram equalization techniques were combined with the 5 color technique to produce better images.

12. ACKNOWLEDGEMENT

We acknowledge the BSc students Salman M. Al- Sheri & Turki B. Al- Zwamil for his hard work and contribution.

XII. REFERENCES

[1]. Atul Bansal.; Rochak Bajpai; J. P. Saini . Simulation Of Image Enhancement Techniques Using Matlab Proceedings of the First Asia International Conference on Modelling & Simulation (AMS'07) © 2007 IEEE Page(s): 296– 301.

[2]. Hongchao Song.; Yuanyuan Shang; Baoyuan Han;Xuefeng Hou Research on Image Enhancement Algorithms Based on Matlab Proceedings of the 4th International Congress on Image and Signal Processing ©2011 IEEE Page(s): 733– 736.