AFRL-HE-BR-TR-2007-0014

ADVANCED IMAGE PROCESSING TECHNIQUES FOR MAXIMUM INFORMATION RECOVERY

James E. Cross
Jiecai Luo

Southern University Office of Grants and Sponsored Programs

November 2006
Final Report for August 2005 to November 2006

Air Force Research Laboratory
Human Effectiveness Directorate
Information Operations and Special Programs Division
Brooks-City-Base TX 78235

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory, Brooks City-Base, Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-HE-BR-TR-2007-0014 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//Signed//
RICHARD ALBANESE, GM-15
Project Manager

//Signed//
JAMES W. RICKMAN, MAJ, USAF
Chief, Special Programs Branch

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** (From - To)
24-11-2006	Technical Report	Aug 2005 - Nov 2006

4. TITLE AND SUBTITLE
Advanced Image Processing Techniques for Maximum Information Recovery

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8650-05-1-6645

5c. PROGRAM ELEMENT NUMBER
05-06-HE

6. AUTHOR(S)
James E. Cross
Jiccai Luo

5d. PROJECT NUMBER
7184

5e. TASK NUMBER
XO

5f. WORK UNIT NUMBER
7184XO5B

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Southern University
Office of Grants and Sponsored Programs
Southern Branch Post Office
Baton Rouge, LA 70813

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Materiel Command
Air Force Research Laboratory
Information Operations and Special Projects Division
Human Effectiveness Directorate
2486 Gillingham Drive
Brooks City-Base, TX 78235

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/HEX

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-HE-BR-TR-2007-0014

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited Refer other requests for this document to AFRL/HEX, 2486 Gillingham Drive, Brooks City-Base, TX 78235.

13. SUPPLEMENTARY NOTES
03-15-07 Cleared for public release: PA-07-091

14. ABSTRACT
The objective of this project was to investigate methods to recover the maximum amount of available information from an image. Some radio frequency and optical sensors collect large-scale sets of spatial imagery data whose content is often obscured by fog, clouds, foliage and other intervening structures. Often, the obstruction is such as to render unreliable the definition of underlying images. Various mathematical operations used in image processing to remove obstructions from images and to recover reliable information were investigated, to include Spatial Domain Processing, Frequency Domain Processing, and non-Abelian group operations. These imaging techniques were researched and their effectiveness determined. Some of the most effective techniques were selected, refined, extended and customized for this project. Several examples are presented showing applications of such techniques with the MATLAB code included. A new advanced image processing technique was developed, tested, and is being proposed for the removal of clouds from an image. This technique has been applied to certain images to demonstrate its effectiveness. The MATLAB code has been developed, tested and appended to this report.

15. SUBJECT TERMS
image processing, spatial domain processing, frequency domain processing

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Richard A. Albanese
Unclassified	Unclassified	Unclassified	SAR	69	**19b. TELEPHONE NUMBER** *(include area code)*

THIS PAGE INTENTIONALLY LEFT BLANK

I. Introduction

The objective of this project was to investigate methods to recover the maximum amount of available information from an image. Some radio frequency and optical sensors collect large-scale sets of spatial imagery data whose content is often obscured by fog, clouds, foliage and other intervening structures. Often, the obstruction is such as to render unreliable the definition of underling images. Various mathematical operations used in image processing to remove obstructions from images and to recover reliable information were investigated, to include Spatial Domain Processing, Frequency Domain Processing and non-Abelian group operations. These imaging techniques were researched and their effectiveness determined. Some of the most effective techniques were selected, refined, extended and customized for this project. Several examples are presented showing applications of such techniques with the MATLAB code included. A new advanced image processing technique was developed, tested and is being proposed for the removal of clouds from an image. This technique has been applied to certain images to demonstrate its effectiveness. The MATLAB code has been developed, tested and appended to this report.

II. Some Fundamental Pre-processing Concepts

A. Introduction

If processing is being performed with the goal of identifying objects in an image, pre-processing to enhance the image is often helpful. Therefore, the first topic addressed in this report involves some popular methods of image enhancement. The first type of concept being presented is Spatial Domain Processing which involves the direct manipulation of pixels. The spatial domain refers specifically to the (x, y) image plane. The second type of concept will be based on Frequency Domain Processing. This will involve taking the Discrete Fourier Transform (DFT) of the spatial image f(x, y) to produce a frequency domain image F(u, v), processing the image in the frequency domain, and then taking the inverse of the DFT to obtain the filtered spatial image g(x, y). The choice of Spatial Domain Processing versus Frequency Domain Processing depends on the nature of the problem. Frequency Domain Processing offers a great deal of flexibility in filter design. A combination of these two methods produces the best results in some cases. The authors, Gonzalez, Woods and Eddins [1] have created, documented in their book, and made available via the Internet, a set of image processing functions that extents the Image Processing Toolbook (IPT) package by about 35%. These functions will be referred to as GWE functions.

B. Some Concepts of Spatial Domain Processing

1. Some Spatial Filtering Techniques

Another term sometimes used for spatial filtering is neighborhood processing. This technique involves replacing the value of a center point with a new value computed based upon the values of the points within the neighborhood. A popular method of defining the neighborhood, along with the center, is a group of nine points. These points consist of the center, the two points directly above and below the center, the two points to the right and left of the center and the four points at the end of the two diagonals on a rectangle drawn about the center. Functions to isolate

the edge of an image are often used as a pre-process to image segmentation. These functions typically use the first or second derivative about the center. The IPT has several popular functions used to compute the edges. Among these are the Prewitt method, the Roberts method, the Sobel Method, the Canny method, the Laplacian and the zero-crossing method. The first four methods are based on the first derivative. The Laplacian is based on the second derivative. The zero-cross method finds edges by looking for zero crossings after filtering the image with a filter specified by the user. The first five methods all use a 9 point mask, w, to be applied to the center point and the 8 points surrounding the center, as designated above. The "edge" function in the IPT is used to find the edge of an image with the user designating which of the six above methods to be used. The "fspecial" function in the Image Processing Toolbook (IPT) can be used to generate the mask, w, for the Prewitt, Sobel or Gaussian methods (along with 6 other types of masks).

However, the mask can be generated by the user if so desired. The IPT function, "imfilter", with one form having the syntax imfilter(f, w), filters the image, f, with the mask provided by the user. Linear spatial filtering is applied. The following example shows the use of the Sobel edge function as applied to a tank vehicle.

```
% Program tank_edge1.m
% This program uses the Sobel method to find the edges of a tank.

Z1 = imread('tank_pic1.jpeg'); %Read the tank image.
Z2 = rgb2gray(Z1);
Z3 = edge(Z2,'sobel');
imshow(Z3)
```

Fig.1. The Original Image

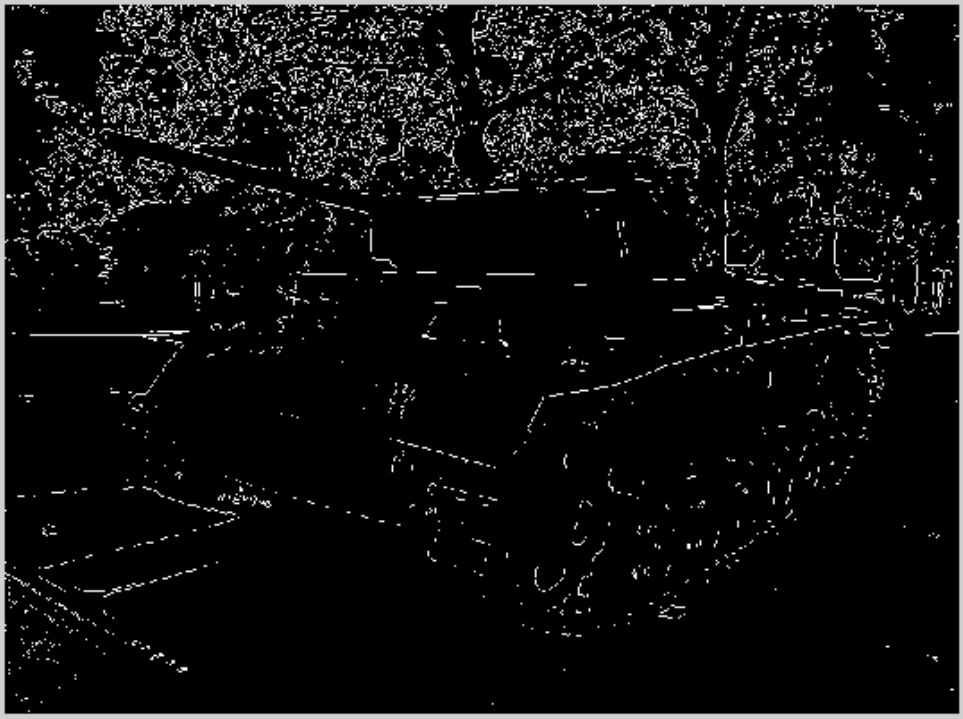


Fig. 2. The Edge of the Image for Fig.1

2. Some Intensity Transformation Techniques

Intensity transformation methods depend only on the intensity values of the image but not explicitly on the (x, y) location of the pixels. One such example is the IPT histogram equalization function “histeq”. The syntax of this function is g = histeq(f, num) where f is the input image, g is the output image and num is the number of intensity levels specified for the output image. For images of class unit8, 255 are commonly used for num. The function histeq enhances the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified histogram. The example below shows the effect of the histogram equalization function on an image consisting of a set of twelve coins. The edge function, discussed above under the heading of Spatial Filtering Techniques, is used along with this function. The three plots will show the original (gray) image, the image of the edges before histogram equalization and the edges of the image after histogram equalization. In comparing Fig. 4 and Fig. 5, it is seen that the edges of several of the coins are much more distinct after applying the histogram equalization function.

```
% Program histeq_coins2.m
% Program to show the effect of the histogram equalization function.
% The three plots will show the original (gray) image, the image of the
```

```
% edges before histogram equalization and the edges of the image after
% histogram equalization.
X1 = imread('Coins1.jpg'); % The coin file must be in the directory.
X2 = rgb2gray(X1);
X3 = rot90(X2,3);
X3 = X3(60:570,1:455);
X4 = histeq(X3,256);  % Apply the histogram equalization process.
X5 = edge(X3,'sobel'); % The edge before histogram equalization.
X6 = edge(X4,'sobel'); % The edge after histogram equalization.
figure, imshow(X3)
figure, imshow(X5)
figure, imshow(X6)
```

Fig. 3. The Original (gray) Coin Image

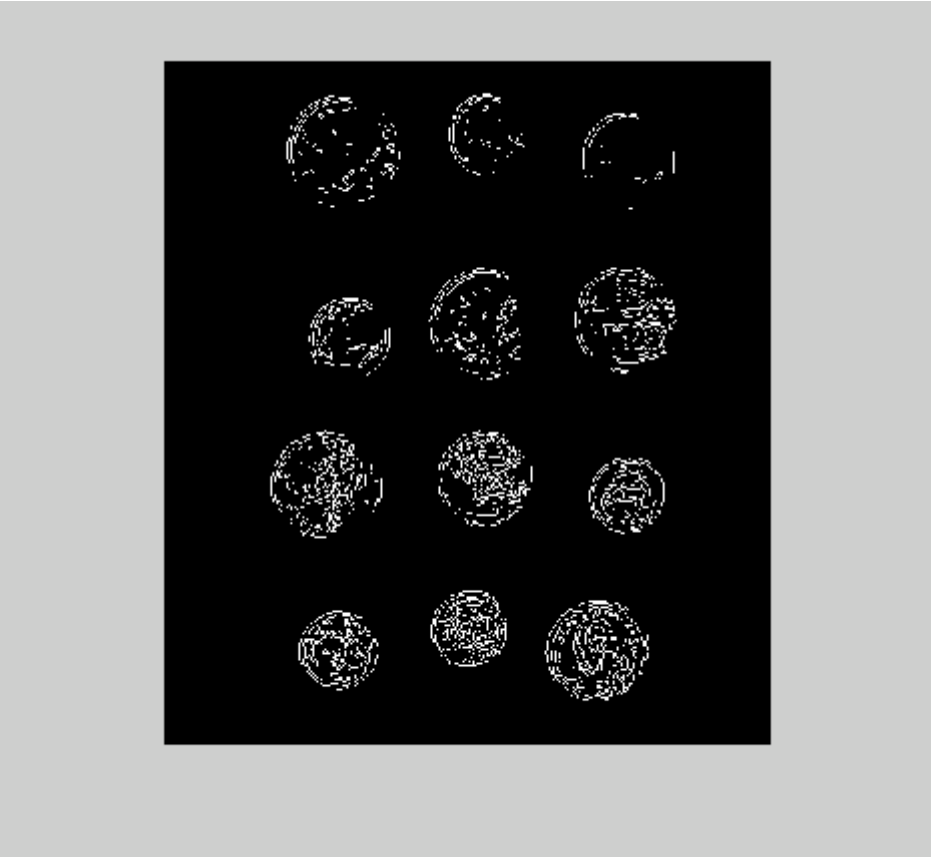


Fig.4. The Edges of the Coin Image Before Histogram Equalization

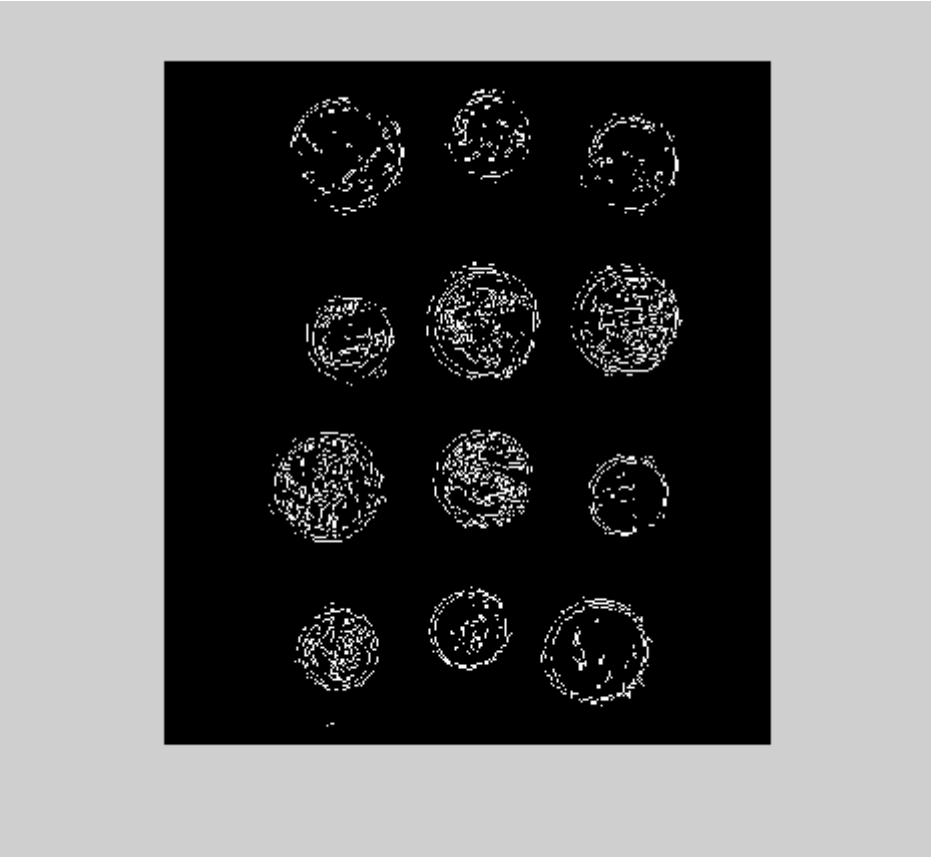


Fig.5. The Edges of the Coin Image After Histogram Equalization

C. Some Concepts of Frequency Domain Processing

The discrete Fourier transform (DFT) is the cornerstone for linear digital filtering. It offers a high degree of flexibility in image enhancement (as well as for other image processing applications). The image file is transferred into the frequency domain using the DFT and the processing is performed in the frequency domain. The data file is then transformed back into the spatial domain using the inverse DFT. The two most widely used types of digital filters based on the band of frequencies filtered are the lowpass and highpass filters. Lowpass filtering results in image blurring or smoothing. Highpass filtering results in image sharpening.

The use of the DFT for image processing closely parallels its use in filtering one-dimensional (1-D) signals such as sound. For image processing, two-dimensional (2-D) filtering is employed. The IPT function used for 1-D filtering has the syntax F = fft(f). The IPT function for 2-D filtering has the syntax F = fft2(f), with the 2 used to designate 2-D filtering. The syntax to obtain the inverse Fourier transform for an image function has the syntax f = ifft2(F). For filtering in the spatial domain, a mask h(x, y) is used to modify the image f(x, y) in some desired manner. For frequency domain filtering, a transfer function H(u, v) is designed to modify the frequency function F(u, v) in some desired manner. The design process involves devising a function H(u, v) to produce the desired effect. Filtering in the spatial domain is faster for a small number of points, h(x, y). As the size of the filtering function increases, filtering in the frequency domain becomes more efficient.

For filtering in the frequency domain, the transfer function, H(u, v), can be obtain by two different methods. The first method is to generate H(u, v) directly from the spatial mask h(x, y). The IPT function fft2 can be used to do this. As examples, one can directly obtain H(u, v) for the Sobel, Prewitt and other such masks. The other method is to generate H(u, v) directly in the frequency domain. Circularly symmetric filters are often used, based on various functions formulated on the basis of the distance of the points from the origin of the transform. The Gonzalez, Woods and Eddins (GWE) [2] function dffuv can be used to compute the distance, D.

The Butterworth and the Gaussian are two popular types of filters. Letting D_o be the distance form the origin that will give a cutoff frequency, the lowpass Butterworth filter can be expressed as

$H(u, v) = 1/[1 + (D(u, v)/D_o)^2]$

The Gaussian lowpass filter can be expressed as

$H(u, v) = e^{-Q}$

Where $Q = D^2(u, v)/[2(D_o)^2]$

For both the Butterworth and the Gaussian filters, given the lowpass filter, the corresponding highpass filter can be computed using the relationship

$H_{hp}(u, v) = 1 – H_{lp}(u, v)$

The lpfilter and the hpfilter are both available in the GWE function set to implement the lowpass and the highpass functions directly in the frequency domain. The filter type can be specified as ideal, Butterworth or Gaussian. Examples are given below.

Fig.6 Original Image

```
% Program Michelle1_lpgaus2
% A lowpasss Gaussian Filter
X = imread('Michelle1.JPEG');
X2 = rgb2gray(X);
rect=[125 100 500 767];
X3 = imcrop(X2,rect);
[M, N] = size(X3);
F = fft2(X3);
sig = 20;
H = lpfilter('gaussian',M,N,sig);
G = H.*F;
g = real(ifft2(G));
figure, imshow(X3)
figure, imshow(g, [ ])
```

Fig. 7. Image after processing with a lowpass Gaussian filter

```
% Program Michelle1_hpgaus2; A highpass Gaussian Filter
X = imread('Michelle1.JPEG');
X2 = rgb2gray(X);
rect=[125 100 500 767];
X3 = imcrop(X2,rect);
[M, N] = size(X3);
F = fft2(X3);
sig = 20;
H = hpfilter('gaussian',M,N,sig);
G = H.*F;
g = real(ifft2(G));
figure, imshow(X3)
```

Fig. 8. Image after processing with a highpass Gaussian filter with companion values used for the lowpass filter of Fig. 7

The below image was produced with the value of sig in the above program chanced to 1.5

Fig. 9. Image after processing with the same highpass Gaussian filter from Fig. 8 but with the value of sig changed to 1.5

Below is an additional example to remove fog from and image.

```
% Program st_tank1_lpfilter_Gaussian2a
c2 = imread('MVC-054S.JPG'); % Read the cloud image.
% c2 is 480x640x3
Z1 = imread('tank_pic1.jpeg'); %Read the tank image.
% The tank image is 594x800x3.
Z1a = Z1(1:2:594,1:2:800,:); % Downsize by selecting every
% other pixel. Z1a is 297 by 400 by 3; ½ the original size.
c2(100:396,120:519,:) = Z1a;
% Embed the low intensity tank image into the cloud image.
Z2 = c1 +.02*c2;
% Most of the desired operations required will not work on 3-D images
% so change to a gray image.
Z3 = rgb2gray(Z2);
[M,N] = size(Z3);
Z3 = double(Z3); % Use double when using fft2 as shown below.
F = fft2(Z3);
sig = 60;
H = lpfilter('gaussian', M,N,sig);
```

```
[U, V] = dftuv(M, N);
H = lpfilter('gaussian', M,N,sig);
G = H.*F;
g = real(ifft2(G)); %Take the inverse fft.
figure, imshow(Z1a); %The downsized tank image.
figure, imshow(Z2);% The tank embedded with the clouds
figure, imshow(g,[]) % The recovered image. Note that the image g will
% show as all white if we do not use [] in the imshow function.
```

Fig.10. The Original Tank Images Downsized

Fig.11. The Tank Embedded in a Cloud Image

Fig.12. The Tank Image Recovered Using a Lowpass Gaussian Filter

III. Comments on Image Restoration and Enhancement

Image restoration and image enhancement have a lot in common but fundamentally have different objectives. The techniques presented above can be employed for both concepts. As the name implies, image restoration has the objective of restoring an image that has been degraded to its previous quality. This implies knowledge of the original appearance of the image and knowledge of the method in which it was degraded. Given knowledge of how the image was degraded, if an inverse of the process is applied, the image will be restored to its original appearance. An example is shown below. The first image shows a small airplane flying over a house top with clouds. The second image is a fog scene. The third image shows the plane embedded in the fog which was fabricated by adding 50% of the intensity of the fog image to the plane image. Since it is known how the plane image was corrupted, recovery of the plane image was made by subtracting out the image that was added. The result is shown in the fourth image.

```
% Program planefog1 to add fog to the plane image
% with the intensity of the fog
% suppressed. Then, recover the image from fog.
X1 =  imread ('Plane1.jpeg');
X2 = imread('fog8.jpeg');
a = .5;
X3 = a*X2 + X1;  % Will add the fog image at 50% intensity.
```

```
X4 = X3 - a*X2;  % Subtract the fog out of the image.
figure,imshow(X1) % The original plane image.
figure, imshow(X2) % The original fog image.
figure, imshow(X3) % The plane embedded in the fog.
figure,imshow(X4, [ ]);%The recovered plane
```

Fig.13. A Small Airplane Flying Over a Housetop With Clouds

Fig.14. A Fog Scene

Fig.15. The Airplane Embedded in the Fog with 50% Intensity of the Fog

Fig.16. The Recovered Airplane With Clouds

A second example is shown below where an image is corrupted with salt and pepper noise using the IPT function imnoise. The IPT median function medfilt2 is then used to restore the image.

```
% Program Michelle_saltpep_median1
% Program to corrupt and image with salt and pepper noise and to restore
% it using a median filter.
X = imread('Michelle1.JPEG');
rect=[125 100 500 767];
Xm = imcrop(X,rect);
Xm = rgb2gray(Xm); %The filtering function below is for 2-D inputs only.
D=.1; % The noise density
X1 = imnoise(Xm,'salt & pepper',D);
M = 7;
N = 7;
X2 = medfilt2(X1,[M N], 'symmetric'); % Perform median filtering with
% each output pixel containing the median value in the M-by N
% neighborhood around the corresponding pixel in the input image.
figure, imshow(X1)
figure, imshow(X2)
```

Fig. 17. Image Corrupted With Salt and Pepper Noise

Fig.18. The Image of Fig.13 restored Using a Median Filter

Whereas image restoration is largely an objective process, image enhancement is largely a subjective process. As the name implies, the goal of image enhancement is to make the image look "better" in some way. This means having some criteria of goodness. As examples, removing wrinkles in a lady's face or changing the red eyes that often appear in images are considered enhancements. Image software developed for professional photographers is heavily weighted towards image enhancement. In the book The Digital Photographer's Guide to Photoshop Elements, one chapter has the title "Making Your Photographs Look Good" and another chapter has the title "Photo Retouching Techniques". The purpose of the book is to show how "to improve your photos and create fantastic special effects" and "is concerned with making pictures better". It states that red eyes occur when light from an on-camera flash reflects off the blood vessels at the back of the subject's eyes. This problem is so common that the Photoshop Elements software package, produced by the Abode Corporation, includes a tool specifically designed to deal with red eyes. The package is called "The Red Eye Removal Tool" [28].

IV. Some Comments on Morphology

In image processing, morphology deals with extracting image components that are useful in describing region shapes, especially boundaries. Morphological techniques are also used in pre-

and post-image filtering. One such IPT function to be used in the next section is bwlabel. This function labels connected components in a binary image. Its syntax is

L = BWLABEL(bf, N)

As described from the MATLAB help directory, the output, L is a matrix of the same size as the input binary image, bf, and contains labels for the connected components in bf. N can have a value of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies 8-connected objects. The default value of N is 8, if N is omitted in the function. The elements of L are integer values greater than or equal to 0. The pixels labeled 0 are the background. The pixels labeled 1 make up one object. The pixels labeled 2 make up a second object, and so on. The number of connected objects can be obtained by using the function with the syntax

[L, NUM] = BWLABEL(bf, N),

where NUM returns the number of connected objects found in the image, bf. Whereas the function BWLABEL supports 2-D inputs only, the function BWLABELN supports any input dimension. It should be noted that bf must be a logical or numeric, real, 2-D, non-sparse image. The output, L, is double. An example is given below showing how the bwlabel function is used to find the number of connected objects in an image. The IPT "find" function is used to give the row and column indices for the pixels associated with a particular object in the image. This, along with thresholding, also used here, will be discussed in the next section. Here, the image shows a set of letters. This program gives an introduction to object recognition techniques. Twenty-one objects were detected. This is because the small dots were also considered to be objects. The eighth object was selected and using the "find" function, a rectangle was determined and it was isolated based on its minimum x and y coordinates and its maximum x and y coordinates. As is seen, the particular object isolated was the letter E.

```
% Program my_bwlabel_test1a
% This program will show how the bwlabel function computes the number of
% connected objects in an image and how the "find" function can be used
% to return the row and column indices for all the pixels associated with
% a particular object number.
X1 = imread('MVC-005S.JPG')
X2 = rgb2gray(X1);
X3 = rot90(X2,3);
X4 = X3(50:620,1:455);  % Crop it.
      for i = 1:571   %  Note i = 620-50 +1 = 571
          for k= 1:455
            if X4(i,k) > 172   % Thresholding is being used here.
          % Note, using 172 here gave the smallest number
          % of objects, a value of Num = 21.
             X5(i,k) = 1;
           else
            X5(i,k) = 0;
           end
       end
   end

[L, num] = bwlabel(X5,8)   % num was given as 21, the number of objects.
```

```
[r,c] = find(L==8)          % Select the 8th object and see what we get.
                            % See the results below.
r1 = min(r)                 % r1 = r_min was 228.
r2 = max(r)                 % r2 = r_max was 311
c1 = min(c)                 % c1 = c_min was 172
c2 = max(c)                 % c2 = c_max was 239
x1 = r1 - 8;                % Start 8 pixels below the min.
x2 = r2 + 8;                % Go to 8 pixels beyond the max.
y1 = c1 - 8;                % Same as above but for c1 and c2
y2 = c2 +8;
X6 = X5(x1:x2, y1:y2);
figure, imshow(X4)
figure, imshow(X5)
figure, imshow(X6)
% Note: The figure shown was the letter E so since we let L== 8, it is
% evident that out of the 21 objects, E is the eighth object.
```

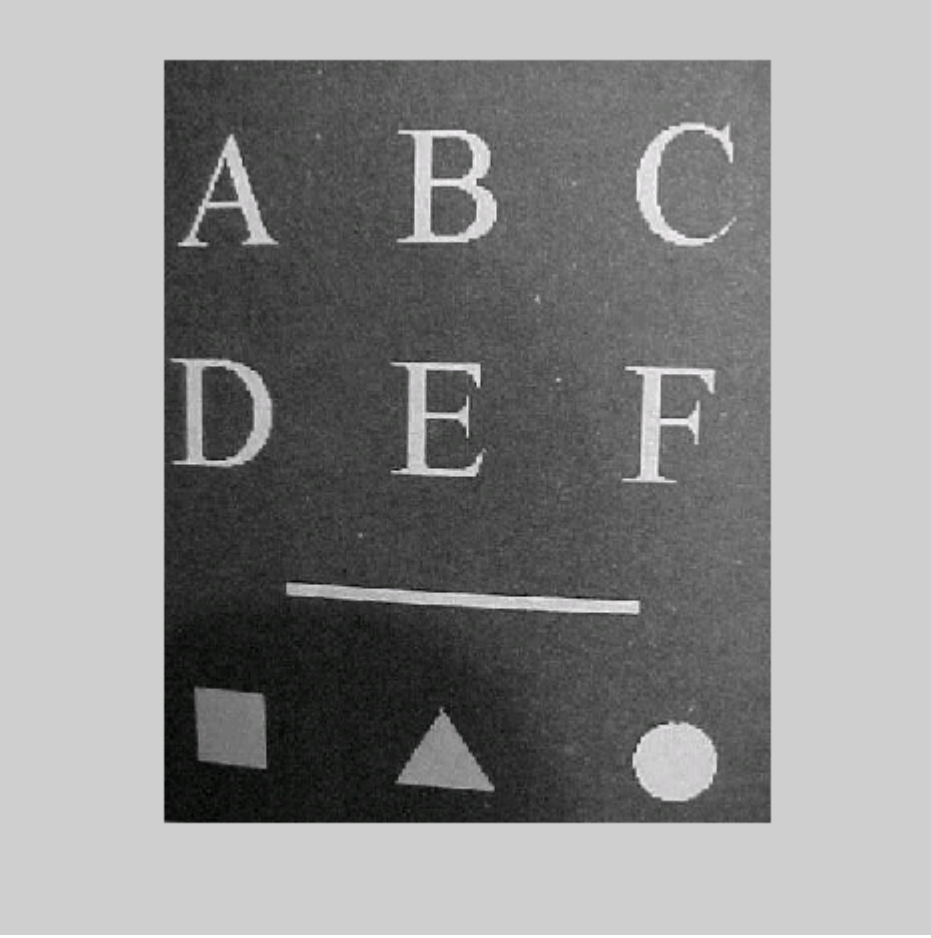


Fig.19. The Original Image After Cropping

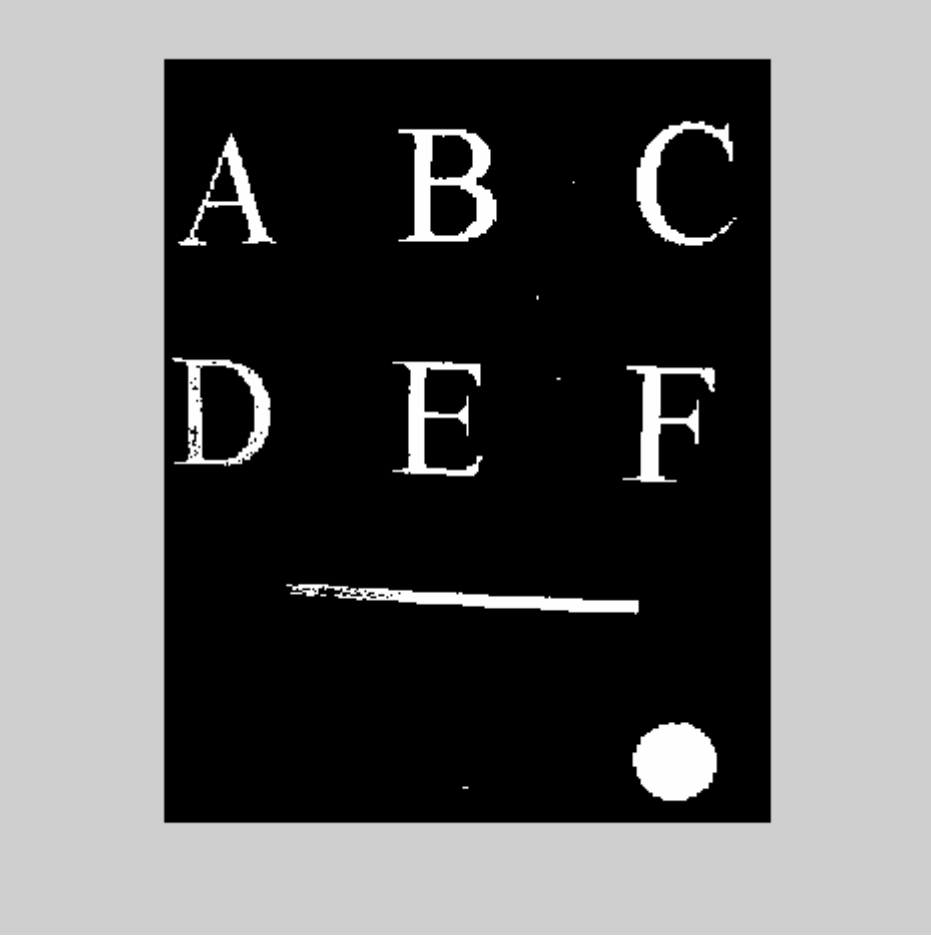


Fig.20. The Image of Fig.19 After Thresholding

Fig.21. The Eighth Object Selected out of 21 Objects Detected

V. Some Segmentation, Representation, Description and Image Recognition Concepts

A. Introduction

The objective of this project was to recover the maximum amount of available information from an image using digital image processing techniques. Recovering information implies rendering

the information such that objects or patterns can be recognized in the image. As mentioned in the previous sections, pre-processing such as filtering is often performed first. The image is then segmented, subdividing the image into its various regions or objects. The next step is to formulate some criteria for representing or describing particular regions or objects in the image. Such descriptors might be based on edges, shapes, sizes, areas, lines, pixel intensity, color, texture, etc. Some descriptors may be interior characteristics while others may be exterior characteristics. The final step is that of being able to recognize one subdivision (object or pattern) of the image from other subdivisions. The various objects or patterns might be labeled in some way to clearly indicate that they have been identified or distinguished from others. The first examples presented will be based mainly on intuitive methods and not very mathematically intense. This will be followed by examples using methods that are rather mathematically intensive. The mathematically intensive methods will involve Logical Image Operations, Connected Operations, Stack Filters and Adaptive Filters using adaptive blind learning algorithms for image processing. A new advanced image processing technique will be demonstrated.

B. An Algorithm to Recognize Different Coins in an Image

The algorithm in the Appendix, listed as A1, was developed and used to successfully segment and then recognize the objects (coins) in an image consisting of a set of 12 coins. There are 3 nickels, 6 dimes and 3 quarters (for a total of $1.50), as shown below. The recognition process is confirmed by the program determining and indicating the sum of the coins.

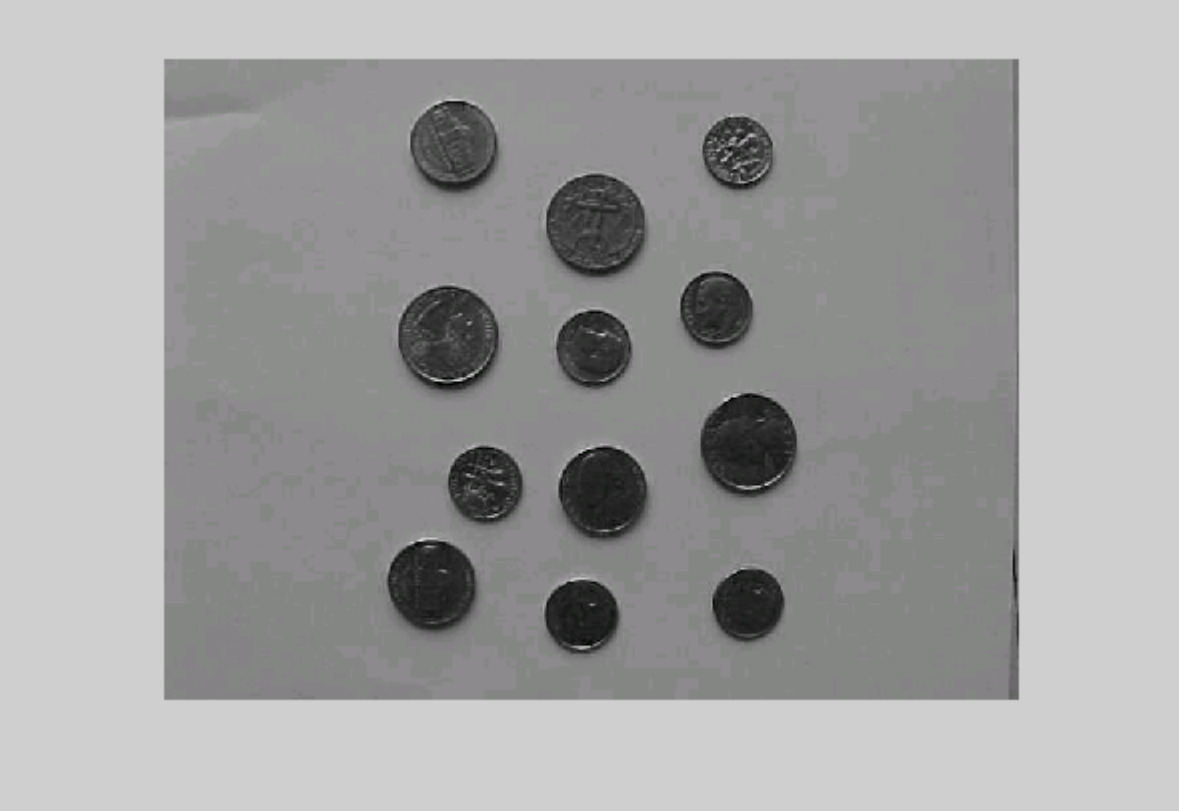


Fig.22. Original Gray Image of Coins

As is shown in the program documentation, thresholding is first performed. The result was to display the coins as all white on a black background (not shown). This is the pre-processing step. The second step was that of segmentation. This was performed using the IPT function bwlabel. As shown earlier, it has the syntax

[L,NUM] = BWLABEL(bf,N)

The algorithm of A1 shows this function written as

[L, num] = bwlabel(X5,N)

Its operation (as explained earlier) is as follows:

It returns a matrix L, of the same size as X5, containing labels for the connected components in X5. N can have a value of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies 8-connected objects. If the argument is omitted, it defaults to 8. The elements of L are integer values greater than or equal to 0. The pixels labeled 0 are the background. The pixels labeled 1 make up one object. The pixels labeled 2 make up a second object, and so on.

The third step was that of classification. This was done by first eliminating any objects that obviously did not belong to the set of objects of interest. Such objects are sometimes designated as outliers. A method was then chosen to classify the remaining objects according to size. Some of the classical methods (such as K-means clustering) could have been used but a simple method was used instead. The various sizes were first observed to see the 3 patterns (or size of clusters) consisting of nickels, dimes and quarters. A second “unsupervised” version of the program, not presented here, ran successfully. It used the relative sizes of the clusters to identify the coins such that programmer intervention was not required. The values of the coins were summed and the result printed to a file called sum.out. The result was also outputted to the screen. The algorithm successfully recognized the total value and printed out $1.50. The below image shows the results of the 12 coins being segmented.

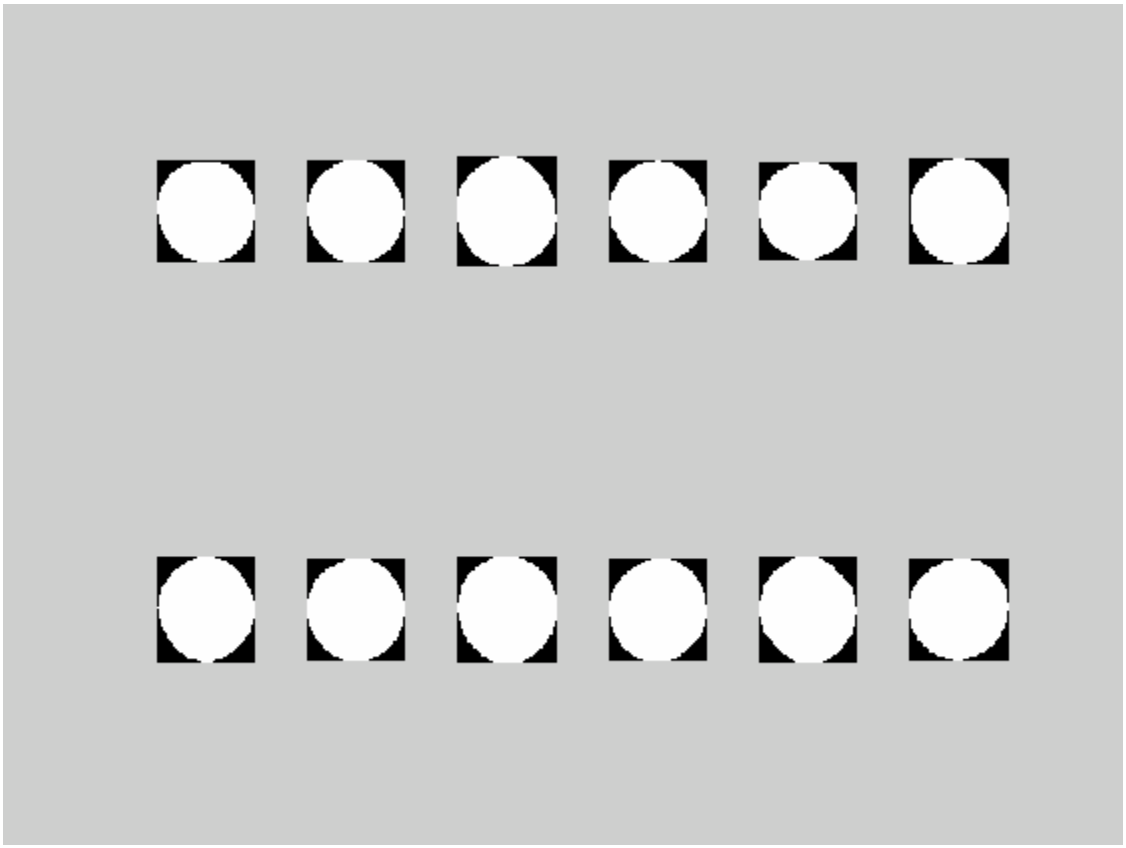


Fig.23. The Set of 12 Segmented and Recognized Objects (Coins)

C. Simple Algorithms to Segment Objects Using the Quadtree Decomposition Method

A simple algorithm to segment objects using the Quadtree Decomposition method will be demonstrated here. This will provide some insight into the next example that is much more complicated. This is shown as A2 in the Appendix. This is an efficient regional base segmentation method. The use of connectivity of pixels is a fundamental requirement for the algorithm. The image is subdivided into quadregions. The regions are merged and/or split to satisfy some stated condition or constraint given as the predicate. All Quadtree regions that satisfy the predicate are filled with 1s. The Quadtree regions that do not satisfy the predicate are filled with 0s. The adjacent sub-regions are merged. The image is segment by this procedure into regions of 1's and 0s. The documentation for the algorithm is provided in A2. The standard deviation and the mean value of the pixels in the sample region were used for the predicate, the criteria for splitting and/or merging. The below image shows the result of segmentation to recognize the 12 coins.

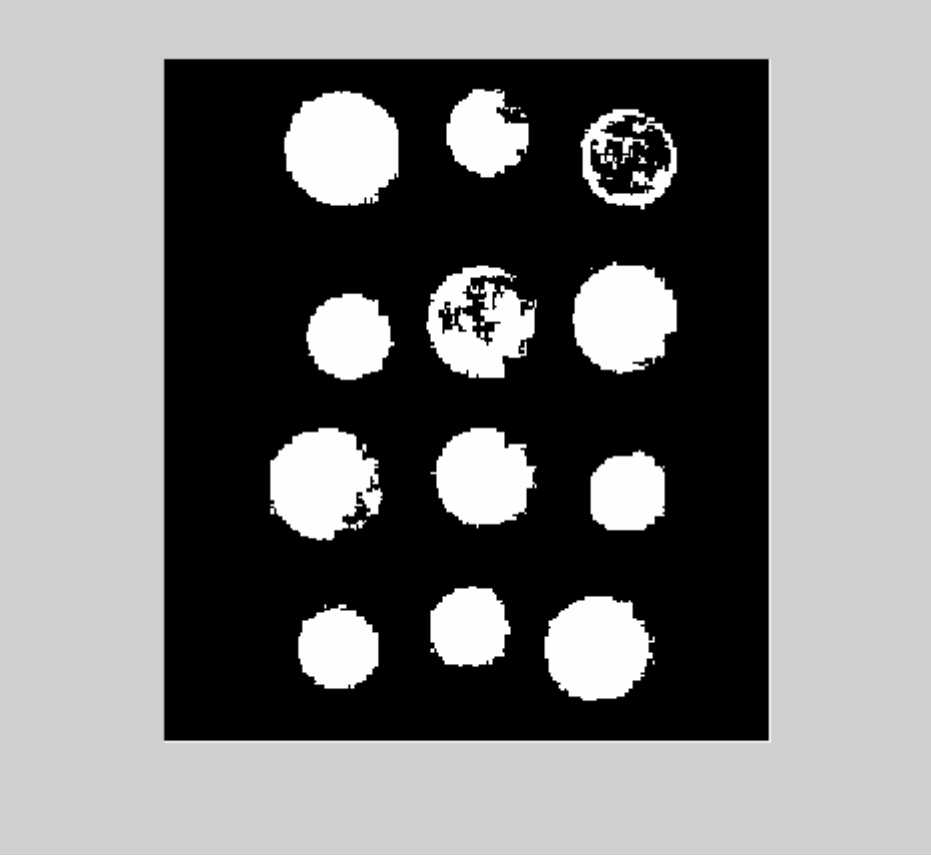


Fig.24. The Result of Operating on the Coin Image with the Quadtree Decomposition Method Without Pre-processing

VI. An Advanced Segmentation Method

A. Introduction and Basic Background

Finally, a much more mathematically intensive method of image segmentation will be presented. In addition to, and including what has been presented above, there are several methods used for removing obscuration information from images: Logical Image Operations, Connected Operations, Stack Filters, and Adaptive filters (using adaptive blind learning algorithms for image processing). A new advanced image processing technique has been developed as a result of this research based on combining these techniques. In order to understand this new method, the essential mathematical basis for each of the last four mentioned techniques is presented below.

Logical image operations: Logical operators are generally derived from *Boolean algebra*, which is a mathematical way of manipulating the *truth values* of concepts in an abstract way without being concerned about what the concepts actually *mean*. The truth value using the Boolean concept can have just one of two possible values, true or false [6].

In the context of image processing, the pixel values in a binary image, which are either 0 or 1, can be interpreted as truth values as above. Using this convention we can carry out logical operations on images simply by applying the truth-table combination rules to the pixel values from a pair of input images (or a single input image and the NOT image by changing the 1's to 0's and the 0's to 1's). Normally, corresponding pixels from each of two identically sized binary input images are compared to produce the output image, which is another binary image of the same size. As with other image arithmetic operations, it is also possible to logically combine a single input image with a constant logical value, in which case each pixel in the input image is compared to the same constant in order to produce the corresponding output pixel. Logical operations can also be carried out on images with integer pixel values. In this extension the logical operations are normally carried out in *bitwise* fashion on binary representations of those integers, comparing corresponding bits with corresponding bits to produce the output pixel value. For instance, suppose that we wish to XOR the integers 47 and 255 together using 8-bit integers. The binary value of 47 is 00101111 and the binary value of 255 is 11111111. XORing these together in bitwise fashion produces the binary number11010000 or the decimal number 208.

Note that not all implementations of logical operators work in such bitwise fashion. For instance some will treat zero as false and any non-zero value as true and will then apply the conventional 1-bit logical functions to derive the output image. The output may be a simple binary image itself, or it may be a gray level image formed perhaps by multiplying what would be the binary output image (containing 0's and 1's) with one of the input images. This operation belongs to morphological image processing.

Connected operations: A grey-scale image partitions the underlying space into regions where the grey-level is constant, the so-called flat zones. A connected operator is an image transformation that coarsens such partitions. Such operators can delete edges, but they cannot change their shape or their location. As a result, connected operators are well-suited for many imaging tasks, such as segmentation, filtering, and coding. Connected operators have become popular in recent years. This is mainly due to the fact that they do not work at the pixel level, but rather at the level of the flat zones of an image. A connected operator can strengthen or weaken boundaries (or even remove them), but as stated above, it cannot shift boundaries or introduce new ones. Therefore, it preserves contour/shape information, which is known to carry most of the image content perceived by human observers. The flat zones of an image are defined as the maximally connected regions of its domain of definition with constant gray level value. In the case of binary images, the flat zones are called grains (foreground) and pores (background). The defining property of a connected operator is that it must coarsen the partition generated by the flat zones of an image.

Stack filters - Dynamic Analysis of Statistical and Deterministic Properties of Stack Filters**:** Many modern signal processing systems and structures incorporate discrete valued operators as basic building blocks. One example is the well known class of stack filters, based on monotone Boolean functions. Another example is the class of threshold Boolean filters, commonly used in document image processing. A number of multi-scale/multi-resolution pyramidal decomposition structures based on the median operation (special case of stack filters) and used in compression and de-noising applications have been recently proposed by a number of authors. Traditionally, analysis of deterministic or statistical properties of such systems or structures has been

conducted in a "static" sense; that is, the system's dynamic characteristics have not been utilized, precluding long-term or steady state analysis in all but the trivial cases.

A new, dynamic analysis approach has been developed for the analysis of such systems. By modeling the sliding window as a Markov chain, it can determine the output distribution function of any *recursive* stack filter as well as its breakdown probabilities and can determine the output distributions of a new, more general, class of stack filters based on mirrored threshold decomposition. The method used relies on *finite automata* and Markov Chain theory. The distribution of any recursive stack filter is expressed as a vector multiplication of steady-state probabilities by the truth table vector of the Boolean function defining the filter. Furthermore, the proposed dynamical analysis approach allows us to study filter behavior along the time dimension. Analogously to recursive linear (IIR) filters which can be unstable, recursive stack filters also can possess a kind of instability. However, this instability manifests itself in a different sense - the filter can get "stuck" on certain values, unable to change. This phenomenon is sometimes referred to as *streaking*. Using the dynamical approach, we can analyze streaking by computing so-called run-length distributions. Additionally, the dynamic analysis approach allows us to study deterministic properties of stack filter systems, or more generally, of systems based on Boolean functions. Finite automata provides a convenient tool for studying invariant (root) signals of stack filters [12-14].

Adaptive blind learning algorithms for image separation (filters): The field of blind separation and de-convolution has grown dramatically during recent years due to its similarity to the separation feature in the human brain, as well as its rapidly growing applications in various fields, such as telecommunication systems, image enhancement and biomedical signal processing. The blind source separation (BSS) problem is to recover independent sources from sensor outputs without assuming any priori knowledge of the original signals besides certain statistic features. Although there exist a number of models and methods, such as the infomax, natural gradient approach and equi-variant adaptive algorithms, for separating blindly independent sources, there still are several challenges in generalizing mixture to dynamic and nonlinear systems, as well as in developing more rigorous and effective algorithms with general convergence [4],[5]. As for using adaptive blind learning algorithms for image separation, it is interesting to note that one of the very early works on independent component analysis (ICA) already proposed a nonlinear method. Although being based on an interesting principle it was rather impractical and computationally heavy. The essential uniqueness of the solution of linear ICA, together with the greater simplicity of linear separation and with the fact that many naturally occurring mixtures are essentially linear, led to a quick development of linear ICA. The work on nonlinear ICA probably was slowed down mostly by its inherent ill-posedness and by its greater complexity, but development of nonlinear methods has continued steadily. Ensemble learning is a Bayesian method and, as such, uses prior distributions as a form of regularization, to handle the ill-posedness problem. It is computationally heavy, but has produced some interesting results, including an extension to the separation of nonlinearly mixed dynamical processes. Kernel-based nonlinear ICA essentially consists of linear ICA performed on a high-dimensional space that is a nonlinear transformation of the original space of mixture observations. In the form in which it was presented in the cited reference, it used the temporal structure of the signals to perform the linear ICA operation. This apparently helped it to effectively deal with the ill-posedness problem, and allowed it to yield some impressive results

on artificial, strongly nonlinear mixtures. The method seems to be quite tractable, in computational terms.

MISEP is an extension of INFOMAX into the nonlinear domain. It uses regularization to deal with the ill-posedness problem, and is computationally tractable. A special class of methods that deserves mentioning deals with nonlinear mixtures which are constrained so as to make the result of ICA essentially unique, as in linear ICA. The most representative class corresponds to the so-called post-nonlinear (PNL) mixtures. These are linear mixtures followed by component-wise invertible nonlinearities. The interest of this class resides both in its unique separability and in the fact that it corresponds to well identified practical situations: linear mixtures observed by non-linear sensors. PNL mixtures and their extensions have had a considerable development. For more details see [4, 15-23].

So far, the methods in the four categories mentioned above have been used individually to remove interferences from images using digital image processing [1, 2, 4, 6], but the individual effectiveness for each method for removing the interferences is not good. There are some combined intelligent computational methods developed in [5], [7] for other image processing purposes other than removing the interferences.

As a result of this research, a combined computational method has been developed based on methods mentioned above: Logical Image Operations, Connected Operations, Stack Filters and Adaptive Filters (Adaptive blind learning algorithms for image processing). The principles of the newly developed combined computational method for removing interferences will be presented, followed by some test results.

B. A New Combined Computational Approach

A new combined combinational approach is being proposed to remove interferences from images and to recovery the maximum amount of available information. It is based on the following three steps:
First step: To identify areas of the interferences on the images; a combination of image segmentation methods, adaptive threshold gain and morphological methods, has been developed;
Second step: To refill the identified areas from step one with wanted areas on the images; a histogram-statistical approximately equivalent method has been developed;
Third step: To smooth the neighborhood of the refilled areas; the MATLAB function roifill will be used here.

The technical detailed steps for each method will be given.

First step: Image segmentation is used to group similar pixels together to form a set of coherent image regions, giving a single image. The pixel similarity could be measured based on the consistency of location, intensity, color, and texture of different pixels. Generally, we can compound these elements together to represent an image pixel, or use some of them. For example, we can only use color components or use both location and intensities. So, for each image pixel, we associate it with a feature vector x. Mainly, there are four approaches to this

problem, including (1) segmentation by clustering; (2) segmentation by graph cut, (3) segmentation by EM algorithm and (4) segmentation by region growing. In this research, focus is on the first method, that of segmentation by clustering.

Image Segmentation by Clustering: Clustering basically means grouping similar data points into different clusters or groups. This section presents two related approaches for clustering: the K-means algorithms and the self-organizing map. The two most important issues in clustering include similarity measurement and the clustering procedure.

K-Means Algorithm:
It is assumed that the number of clusters, *K*, is given. The center of each clusters C_i is used to represent the cluster. The center of each cluster is the mean of the data points which belong to such a cluster. How is the center of a group of data point determined? Basically, a distance similarity measurement is defined, *D*(*x; y*).

For example, $D(x, y) = \|x - y\|^2$ might be used to define such a measurement. We can compare the distance of a data point to these cluster centers, and such a data point belongs to the nearest cluster:

$$l_k(x_k) = \arg\min_i D(x_k, C_i) = \arg\min_i \|x_k - C_i\|^2$$

where l_k is the label for the data point x_k. The K-means algorithm tries to find a set of such cluster centers such that the total distortion is minimized. Here, the distortion is defined by the total summation of the distances of data points from its cluster center:

$$\phi(\mathcal{X}, \mathcal{C}) = \sum_{i \in \mathcal{C}} \sum_{j \in \text{i-th cluster}} ||x_j - C_i||^2$$

To minimize ϕ, K-means algorithm iterates between two steps:

Labeling: Assume the *p*-th iteration results in a set of cluster centers $C_i^{(p)}, i = 1,2,...K.$ Label each data point based on such a set of cluster centers, i.e., $\forall x_k$, find

$$l_k^{(p+1)}(x_k) = \min_i ||x_k - C_i^{(p)}||^2$$

and group data points belonging to the same cluster

$$\Omega_j = \{x_k : l_k(x_k) = C_j\}$$

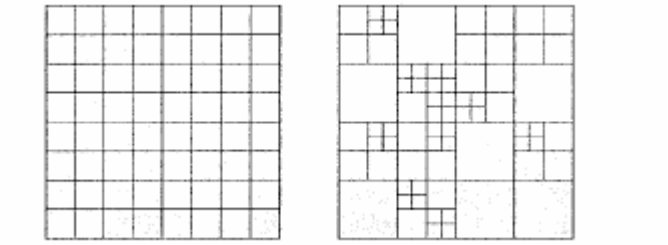
Re-centering: Re-calculating the centers for all the clusters

$$C_i^{(p+1)} = \frac{\sum_{x_k \in \Omega_i} x_k}{|\Omega_i|}$$

The algorithm iterates between such labeling and re-centering steps, and is expected to converge to a local stationary status. Based on the principle of the algorithm above, three general image segmentation methods are proposed here.

Metric distance defined on normalized color histogram:
First, the color histogram used here is normalized, that is, all color histograms are given by percentages (instead of by true values). Next, a large-scale spatial image is partitioned into

$n \times m$ smaller sub-images. Here, one of two ways for partitioning the large-scale spatial image are used, fixed block size or quad-tree as shown below:



Fixed block size (left) and quad tree (right)

The metric measurement based on spatial color histogram for the reference images $I_{r\_i}, i = 1,2,...k$ and the sub-images $I_{m\_j}, j = 1,2,...n \times m$ is based on two factors: spatial color histogram and spatial color histogram varying differences, and defined as

(1) Distance measurement on reference image I_{r_i} and sub-image I_{m_j} for spatial color histogram:

$$d(h_{I_{r\_i}}, h_{I_{m\_j}}) = \frac{\sum_{n=1}^{N} (h_{I_{r\_i}}(n) - h_{I_{m\_j}}(n))^2}{\sum_{n=1}^{N} (h^2{}_{I_{r\_i}}(n) + h^2_{I_{m\_j}}(n))}$$

(2) Distance measurement on reference image I_{r_i} and sub-image I_{m_j} for spatial color histogram varying differences:

$$d(n) = h(n) - h(n-1), n = 2,3,,,,N$$

$$d(d_{I_{r\_i}}, d_{I_{m\_j}}) = \frac{\sum_{n=1}^{N-1} (d_{I_{r\_i}}(n) - d_{I_{m\_j}}(n))^2}{\sum_{n=1}^{N-1} (d^2_{I_{r\_i}}(n) + d^2_{I_{m\_j}}(n))}$$

The whole metric measurement is

$$d(I_r, I_{m\_j}) = \min_{i=1,2,...k} (\lambda d(h_{I_{r\_i}}, h_{I_{m\_j}}) + (1-\lambda) d(d_{I_{r\_i}}, d_{I_{m\_j}})),$$

where $\lambda \le 1$ is a positive and adjustable parameter. (It should be noted that the metric given above is also a special EMD).

The regions to be segmented are decided by

$$\Omega_j : j = 1,2,...n \times m : \begin{cases} d(I_r, I_{m\_j}) \le \rho, & (I_{m\_j} \in I_r) \\ d(I_r, I_{m\_j}) > \rho, & (I_{m\_j} \notin I_r) \end{cases}$$

Metric distance defined on image (statistical) moments:

When a large-scale spatial image is partitioned into $n \times m$ smaller sub-images, the metric measurement based on statistical moments for the reference images $I_{r\_i}, i = 1,2,...k$ and the sub-images $I_{m\_j}, j = 1,2,...n \times m$ is considered by three facts, the original images, the first derivatives of original images, and second derivatives of original images. These are defined as follows:

(1) Distance measurement on reference image I_{r_i} and sub-image I_{m_j} for each seven invariant moments: $\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6$ and ϕ_7 (more details see reference 1 at page 675):

$$d(m_{I_{r\_i}}, m_{I_{m\_j}}) = \frac{\sum_{n=1}^{7} (m_{I_{r\_i}}(n) - m_{I_{m\_j}}(n))^2}{\sum_{n=1}^{7} (m^2{}_{I_{r\_i}}(n) + m^2_{I_{m\_j}}(n))}$$

(2) Distance measurement on the first derivatives of reference image: $I_{r\_1x\_i}, I_{r\_1y\_j}$ and sub-image: $I_{m\_1x\_j}, I_{m\_1y\_j}$:

$$d(m_{I_{r\_1\_i}}, m_{I_{m\_1\_j}}) = 0.5d(m_{I_{r\_1x\_i}}, m_{I_{m\_1x\_j}}) + 0.5d(m_{I_{r\_1y\_i}}, m_{I_{m\_1y\_j}})$$

(3) Distance measurement on the second derivatives of reference image:
$I_{r\_2xx\_i}, I_{r\_2xy\_i}, I_{r\_2yx\_i}, I_{r\_2yy\_i}$ and sub-image: $I_{m\_2xx\_j}, I_{m\_2xy\_j}, I_{m\_2yx\_j}, I_{m\_2yy\_j}$

$$d(m_{I_{r\_2\_i}}, m_{I_{m\_2\_j}}) = 0.25d(m_{I_{r\_2xx\_i}}, m_{I_{m\_2xx\_j}}) + 0.25d(m_{I_{r\_2xy\_i}}, m_{I_{m\_2xy\_j}})$$
$$0.25d(m_{I_{r\_2yx\_i}}, m_{I_{m\_2yx\_j}}) + 0.25d(m_{I_{r\_2yy\_i}}, m_{I_{m\_2yy\_j}})$$

The whole metric measurement is

$$d(I_r, I_{m\_j}) = \min_{i=1,2,...k} (\lambda_1 d(m_{I_{r\_i}}, m_{I_{m\_j}}) + \lambda_2 d(m_{I_{r\_1\_i}}, m_{I_{m\_1\_j}}) +$$
$$(1 - \lambda_1 - \lambda_2) d(m_{I_{r\_2\_i}}, m_{I_{m\_2-j}}))$$

where $\lambda_1 \le 1, \lambda_2 \le 1$ are positive and adjustable parameters.
The segmented regions are decided by

$$\Omega_j : j = 1,2,...n \times m : \begin{cases} d(I_r, I_{m\_j}) \le \rho, & (I_{m\_j} \in I_r) \\ d(I_r, I_{m\_j}) > \rho, & (I_{m\_j} \notin I_r) \end{cases}$$

Metric distance defined on normalized histogram statistical moments:
In this method, the random variables are defined on the normalized color histogram. Let z_i be a discrete random variable that denotes intensity levels in an image, and let $p(z_i), i = 0,1,2,...L-1,$ be the corresponding normalized histogram, where L is the number of possible intensity values. A histogram component, $p(z_j)$, is an estimate of the probability of occurrence of an intensity value z_j, and the histogram may be viewed as an approximation of the intensity PDF. One of the principal approaches for describing the shape of the histogram is via its central moments (moments about the mean), which are defined as

$$\mu_n = \sum_{i=0}^{L-1} (z_i - m)^n \, p(z_i)$$

where n is the moment order, and m is the mean:

$$m = \sum_{i=0}^{L-1} z_i \, p(z_i)$$

Because the histogram is assumed to be normalized, the sum of all its components is one, so, (from the equation above) $\mu_0 = 1, \mu_1 = 0$. The second moment,

$$\mu_2 = \sum_{i=0}^{L-1} (z_i - m)^2 \, p(z_i)$$

is the variance. Once all order moments are defined, then, metric distance is defined as follows: Histogram statistical moments for the reference images $I_{r\_i}, i = 1,2,...k$ are given by

vectors $\mu_{r\_i} = \begin{bmatrix} \mu_{2\_i} \\ ... \\ \mu_{n\_i} \end{bmatrix}, i = 1,2,...k$. The $n \times m$ smaller sub-images partitioned on a large image I_{m_j}, $j = 1,2,...n \times m$, the histogram statistical moments for each of them are given by

vectors $\mu_{m\_j} = \begin{bmatrix} \mu_{2\_j} \\ ... \\ \mu_{n\_j} \end{bmatrix}, j = 1,2,...n \times m$, and distance measurement on reference image I_{r_i} and sub-image I_{m_j} for each histogram statistical moments is defined as

$$d(m_{I_{r\_i}}, m_{I_{m\_j}}) = \left\| \mu_{r\_i} - \mu_{m\_i} \right\| = \sqrt{\sum_{k=2}^{n} (\mu_{k\_i} - \mu_{k\_j})^2}$$

The segmented regions are decided by

$$\Omega_j : j = 1,2,...n \times m : \begin{cases} d(I_r, I_{m\_j}) \le \rho, & (I_{m\_j} \in I_r) \\ d(I_r, I_{m\_j}) > \rho, & (I_{m\_j} \notin I_r) \end{cases}$$

Here the threshold value ρ is adjusted by trial and error, which is adaptive processing.

At the same time, segmented regions based on the three metric distances defined above may have some small unexpected spots. Some morphological methods such as dilation, erosion, opening and closing will be used here to remove these unwanted spots. Once the segmented regions are obtained, the quad-tree decomposition method is used to obtain approximations to the regions for the next application.

Step two: It is assumed that if two images have some similarities, both of histograms have some similarities. Here a histogram-statistical approximately equivalent method has been developed to refill the identified areas from step one with wanted areas on the images. The principle is as stated below:

Assumed that there is a wanted reference image I_r with the histogram $h_r(i), i = 0,1,2,...L-1$, let z_i be a discrete random variable that denotes intensity levels in the image, and let $p_r(z_i), i = 0,1,2,...L-1,$ be the corresponding normalized histogram, where L is the number of possible intensity values. Define a cumulative variable $c_r(k) = \sum_{i=0}^{k} p_r(z_i), k = 0,1,2,...L-1,$ and have $c_r(L-1) = 1$. Assuming that there is an identified image I_m to be removed with the histogram $h_m(i), i = 0,1,2,...L-1$, let z_j be a discrete random variable that denotes intensity levels in the removed image, and let $p_m(z_j), j = 0,1,2,...L-1,$ be the corresponding normalized

histogram, where L is the same number of possible intensity values as in the wanted reference image. Also define a cumulative variable $c_m(k) = \sum_{j=0}^{k} p_m(z_j), k = 0,1,2,...L-1$, and have $c_m(L-1) = 1$. Once the wanted reference image PDF is obtained, the new intensity levels $\bar{z}_j, j = 0,1,2,...L-1$ in the removed image can be assigned by the rule:

1. Given the original z_j on an identified to be removed image I_m;
2. Obtain $c_m(j) = \sum_{i=0}^{j} p_m(z_i)$;
3. set $c_r(k) = c_m(j)$;
4. $\bar{z}_j = c_r^{-1}(k) = z_k$, here z_k is the intensity level in the wanted reference image.

Step three: When the first two steps are used on the given image to remove the interferences, there may be some edges on the processed image. A smoothing process for the neighborhood of the refilled areas is then necessary. There are many ways that can be adopted to realize this operation such as the MATLAB functions roifill, filtering average, etc.

The MATLAB code has been developed based on the above three steps and the effectiveness for the removal of interferences tested. Simulation results will be provided in the next section.

C. Simulation Results Based on the New Combined Computational Approach

Three groups of images are used to test the effectiveness of the proposed new combined computational approach.

Group One- Radar Images: A radar image is shown in figure 25. Some interferences (clouds) in this figure are located in figure 26 as yellow squares.

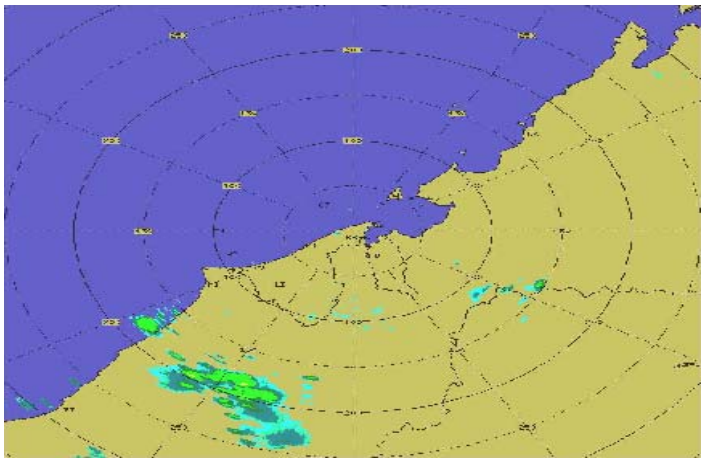


Fig. 25: Radar Image One.

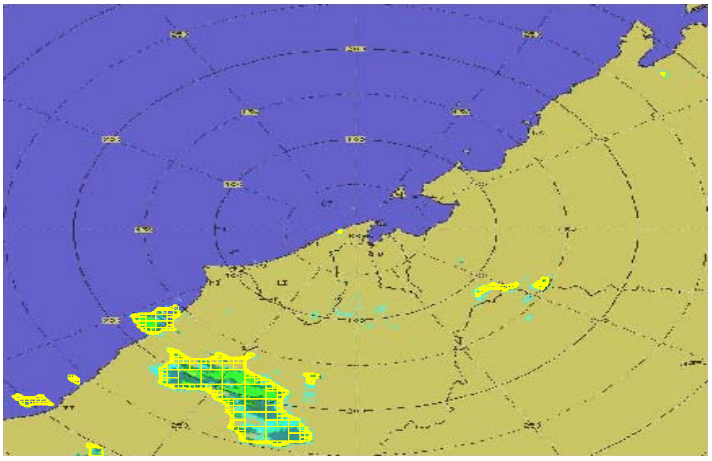


Fig. 26: Radar image one with some identified interferences to be removed.

In order to get the locations in figure 26, the threshold $\rho = 0.1$ was used. The spots will be deleted if these areas are less than 16 pixels square. The radar image with removed interferences after applying the second and third steps is shown in figure 27.

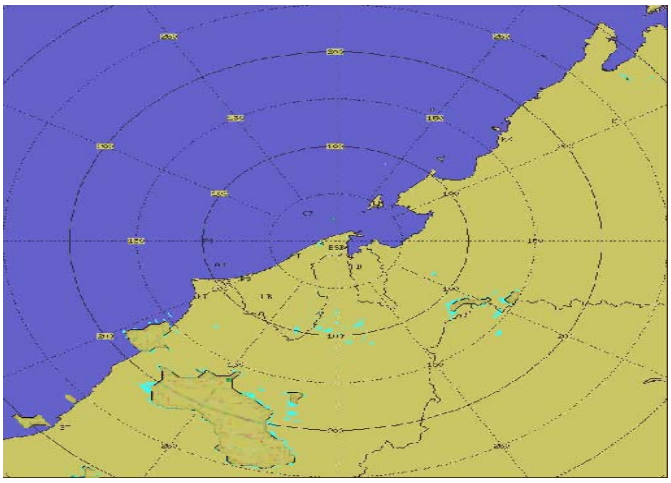


Fig. 27: Radar image one after interferences removed.

The second radar image is shown in figure 28. Some interferences (clouds) in the figure are located in figure 29 as yellow squares.

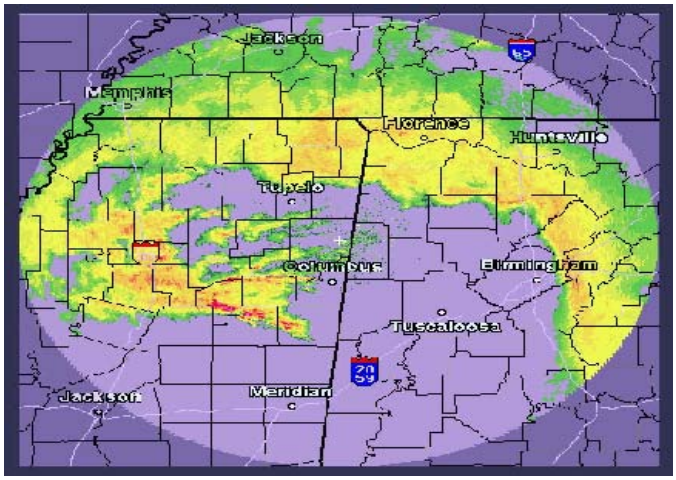


Fig. 28: Radar image two.

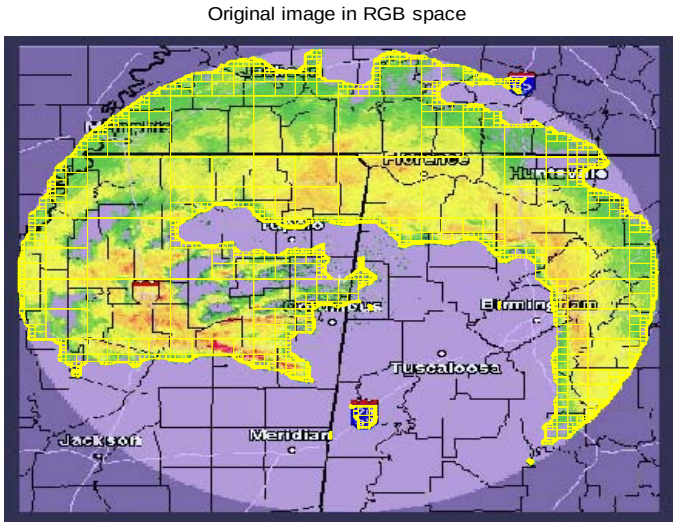


Fig. 29: Radar image two with some identified interferences to be removed.

In figure 29, a threshold of $\rho = 0.1$ was used. The identified spots will be deleted if these areas are less than 16 pixels square. Figure 30 shows the results on the radar image after the second and third steps are applied.

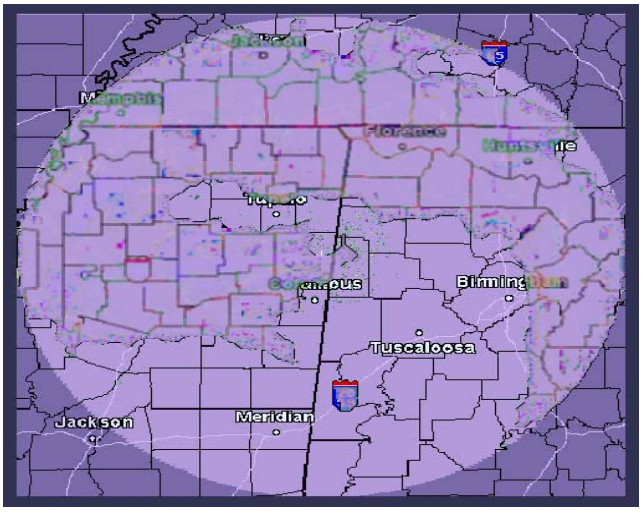


Fig. 30: Radar image two after interferences are removed.

The third radar image is shown as figure 31. Some interferences in the figure are located in figure 32 as yellow squares.

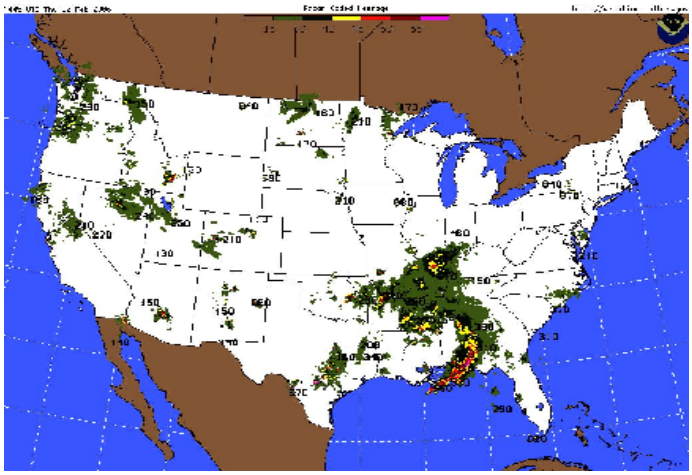


Fig.31: Radar image three

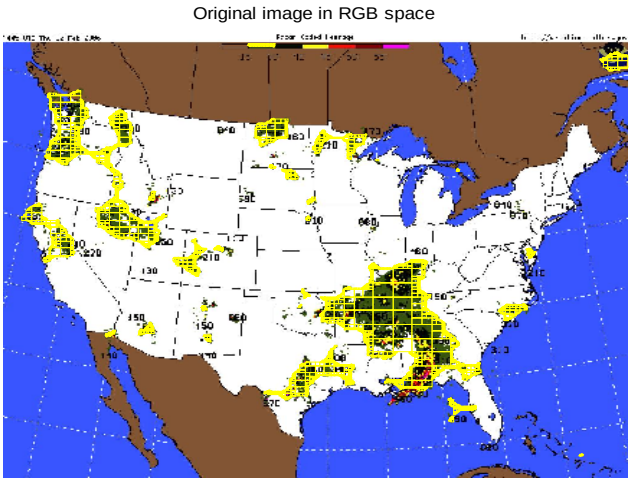


Fig. 32: Radar image three with some identified interferences to be removed.

In figure 32, a threshold value $\rho = 0.035$ was used. The spots will be deleted if these areas are less than 16 pixels square. Figure 33 shows the results on the radar image after the second and third steps are applied.

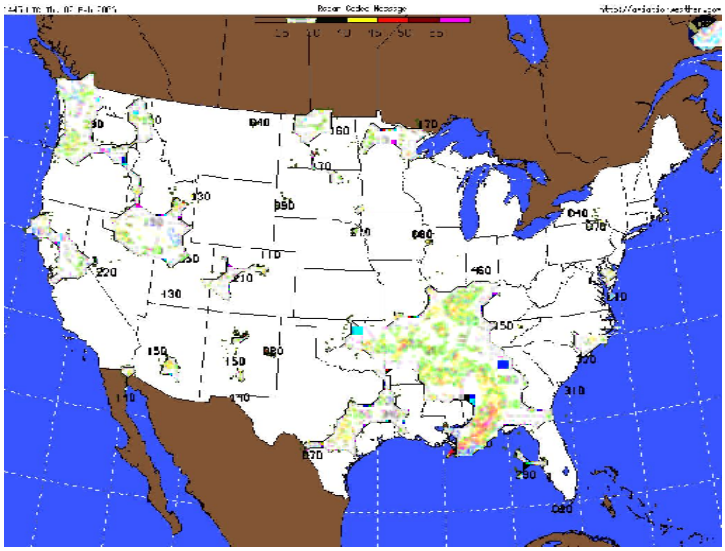


Fig. 33: Radar image three after interferences removed.

Step three, smoothing, is applied on figure 33. The new image resulting is shown in figure 34.

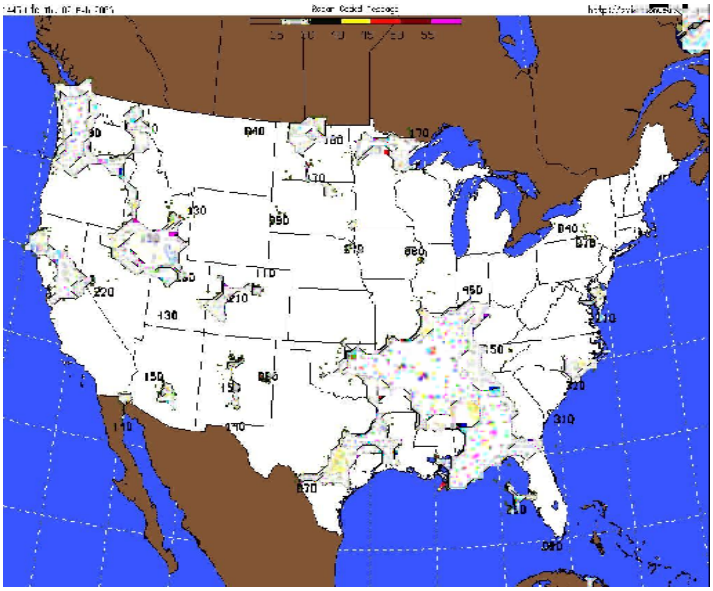


Fig. 34: Radar image three after the smoothing processing is applied.

Group Two- Coin Images: The first coin image is shown in figure 35. Here interferences in the figure are located as shown in figure 36 as yellow squares.

Figure 35: Coin image one

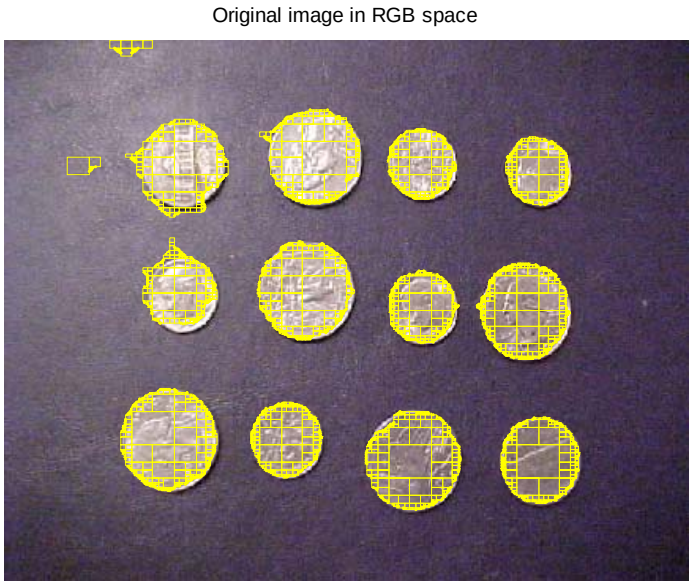


Fig. 36: Coin image with identified interferences to be removed.

In figure 36, a threshold of $\rho = 0.02$ was used. The spots will be deleted if these areas are less than 9 pixels square. Figure 37 shows the results on the coin image after the second and third steps are applied.

Fig. 37: Coin image one after interferences are removed.

Group Three- Clouds for Images Made From the Ground: The first cloud image is shown in figure 38. The interferences, clouds, are located in figure 39 as yellow squares.

Fig. 38: Cloud image one.

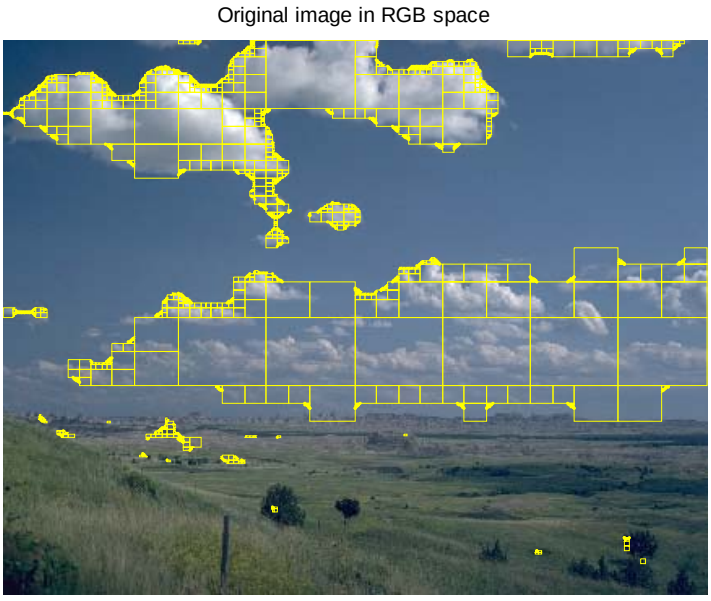


Fig. 39: Cloud image one with some identified clouds to be removed.

In figure 39, a threshold $\rho = 0.06$ was used. The spots will be deleted if these areas are less than 16 pixels square. Figure 40 shows the results on the cloud image after the second and third steps are applied.

Fig. 40: Cloud image one after interferences are removed.

Step three: The smoothing step is applied to figure 40. The resulting new image is shown in figure 41.

Fig. 41: Cloud image one after the smoothing processing is applied.

Cloud image two is shown in figure 42. The interferences, (clouds) are located in figure 43 as yellow squares.

Fig. 42: Cloud image two.

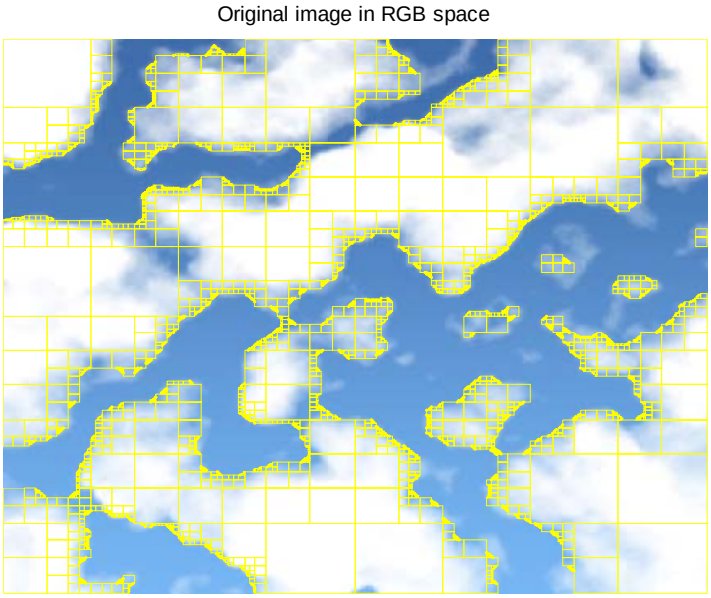


Fig. 43: Cloud image two with some identified clouds to be removed.

Here a threshold $\rho = 0.05$ was used. The spots will be deleted if these areas are less than 8 pixels square. Figure 44 shows the results on the cloud image after the second and third steps are applied.

Fig. 44: Cloud image two after interferences removed.

Fig. 45: Cloud image two after the smoothing processing is applied.

The cloud image three is shown in figure 46. The interferences (clouds) in the figure are located in figure 47 as yellow squares.

Fig. 46: Cloud image three.

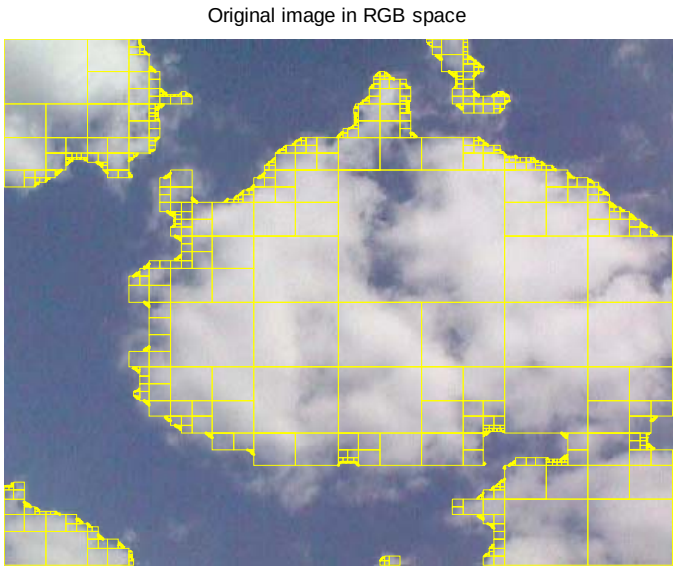


Fig. 47: Cloud image three with some identified clouds to be removed.

Here a threshold of $\rho = 0.065$ was used. The spots will be deleted if these areas are less than 8 pixels square. Figure 48 shows the results on the cloud image after the second and third steps are applied.

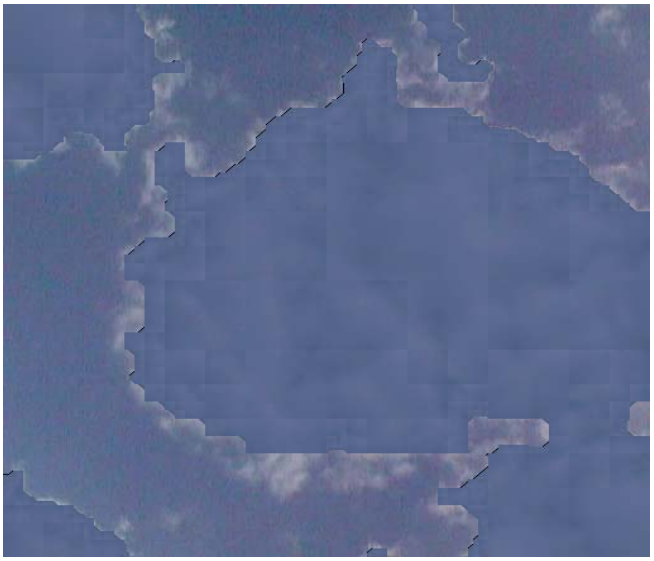


Fig.48: Cloud image three after the interferences are removed.

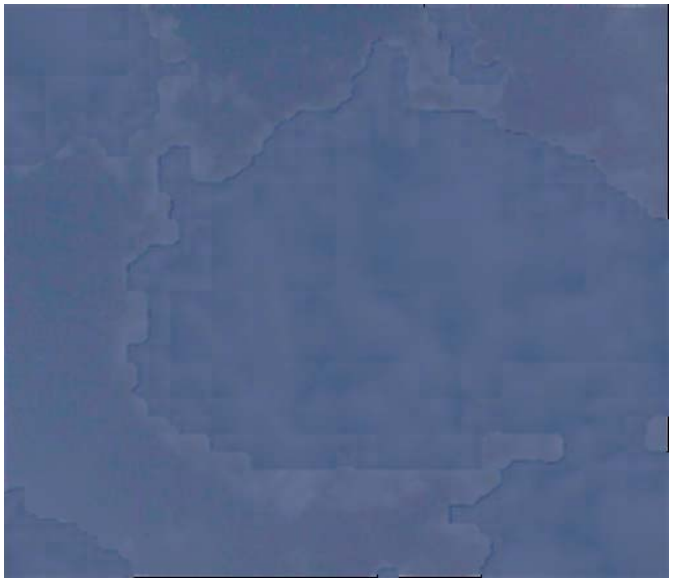


Fig. 49: Cloud image three after the smoothing processing is applied.

The test results provided from figure 25 through figure 49 show that the proposed new combined computational approach for interference removal is effective.

VII. On the Topic of Maximum Information Recovery

The topic of the Maximum Information Recovery from an image can be considered from several viewpoints. But in all cases, it must be recognized that images are simply described by numbers assigned to the x and y coordinates of the image. For a gray image, a single number is assigned to a coordinate, representing the intensity of the pixel at that point. For a color image, three numbers are assigned to the coordinate, representing the red, green and blue components of the image at the coordinate. This constitutes the information in the image.

The topic can be considered from the viewpoint of reconstructing the true image from incomplete data, or it can be considered from being able to glean as much information as possible form the actual data that is present in the image. Much of the research performed on image reconstruction has been carried out in the medical imaging area. Work performed at the Vancouver Health Sciences Centre's Medical Imaging Research Group (MIRG) is described at the Website http://www.physics.ubc.ca/~mirg/home/tutorial/recon.html. This group has performed imaging using a Single Photon Emission Computed Tomography (SPECT) which has allowed them to visualize functional information about a patient's specific organ or body system. It is explained that the problem of reconstructing medical images from measurements of radiation around the body of a patient belongs to the class of inverse problems which are characterized by the fact that the information of interest (the distribution of radioactivity inside the patient) is not directly available. Problems result from scatter and background radiation, as well as from the radioactivity distribution of interest. The mathematics of image reconstruction (an iterative process) is presented, along with the research performed. A demonstration of the reconstruction process is included. The presentation is highly mathematical.

Similar research is being performed at other sites such as at the University of Michigan. Researchers here have developed a MATLAB image reconstruction toolbox with both iterative and non-iterative algorithms. The algorithms are for SPECT (as described above) as well as for X-ray, PET and CT imaging. The software is available at the Website http://www.eecs.umich.edu/~fessler/code/

The course EE369C: Medical Image Reconstruction is taught at Stanford University. This course covers magnetic resonance imaging (MRI), X-ray computed Topology (CT) and positron emission tomography (PET). The syllabus for this course can be found at the Website http://www.stanford.edu/class/ee369c/

If the topic of Maximum Information Recovery from an image is considered from the viewpoint of being able to glean as much information as possible form the actual data that is present in the image, this can be addressed from two different viewpoints. Probably, the main interest would be that of removing interfering or unwanted data from the image so as to recover the true image. Methods of removing such interferences from images have been the main focus of this research. A second thought might center on encoding information in an image. From this viewpoint, it might be concluded that an unlimited amount of information can be encoded in an image. One such example of encoding information in an image will be presented. The program shown as A4 in the Appendix embeds the message "GOD BLESS AMERICA" in the 20th row of the clown image shown below. The encoded information (very difficult to detect) is at the immediate left in

the image, replacing the corresponding green pixels in the color image (having red, green and blue components).

Fig.18. The message “GOD BLESS AMERICA” is encoded in the 20th row of this image.

When this image is loaded into the computer, the program A5 in the Appendix recovers the encoded message. The program produces the below message in a file called messfile2.out:

GOD BLESS AMERICA

VIII. Conclusions and Further Works

Various methods to remove obstructions from images and to recover reliable information were developed. These methods were successfully tested and the results presented along with the MATLAB code. Included is a new advanced image processing method that was developed and tested. This method uses a combination of Logical Image Operations, Connected Operations, Stack Filters, and Adaptive filters (using adaptive blind learning algorithms for image processing). The effectiveness of these techniques was demonstrated on a variety of images with obstructions to include fog and clouds.

Further work is need on the identification/recognition of objects following the segmentation process.

IX. Acknowledgment

This is to acknowledge with appreciation the program manager for this project, Dr. Richard Albanese from the AFRL Brooks City-Base. The contributions of two graduate students and two undergraduate students are also acknowledged. The two graduate students were Mashama S. Osborne and Marlon Richardson. The two undergraduate students were Jason Winfrey and Jason Berry.

X. References

1. R. Gonzalez and R. Woods, Digital Image Processing, 2nd edition, Prentice-Hall Inc., Upper Saddle River, NJ, 2002.
2. A. Jain, Fundamentals of Digital Image Processing, Prentice-Hall Inc. 1989.
3. R. Duda, Peter E. Hart and D. Stork, Pattern Classification, 2nd edition, John Wiley & Sons, Inc. 2000.
4. Andrzej Cichocki Shun-ichi Amari, Adaptive Blind Signal and Image Processing Learning Algorithms and Applications, John Wiley & Sons, Ltd, 2002.
5. Sturt William Perry, Hau-San Wong and Ling Guan, Adaptive Image Processing A Computational Intelligence Perspective, CRC Press, 2002.
6. E. R. Dougherty and J. T. Astola, Nonlinear Filters for Image Processing, IEEE Press, 1999.
7. Steve Mann, Intelligent Image Processing, John and Wiley & Sons, Inc. 2002.
8. Jiecai Luo and Kenneth W. Tobin, "A novel method based combined color features for large-scale spatial image data retrieval," 37th Southeastern Symposium on System Theory (SSST), Tuskegee University, Alabama, 2005.
9. Jiecai Luo, "Automated Image Retrieval Based Large-scale Spatial Data Management and Analysis" 2003 Technique Report, Oak Ridge Institute for Science and Education and Image Science & Machine Vision Group at ORNL, ORNL/ORAU HBCU and MEI Summer Faculty Research Program, 2003.
10. Jiecai Luo, "Content-Based (Unsupervised) Image Segmentation for Large-scale Spatial Images (with MATLAB)," 2004 Technique Report, Oak Ridge Institute for Science and Education and Image Science & Machine Vision Group at ORNL, Faculty and Student Teams (FaST) Summer Program, 2004.
11. Zhengmao Ye, Jiecai Luo, Pradeep Bhattacharya, Hamid Majlesein, "Processing of Slow Motion Images Out Of Underwater, Ground and Aerial Vehicles", Proceedings of 2005 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE05), 2005..
12. I. Shmulevich, T.M. Sellke, M. Gabbouj, E.J. Coyle, Stack Filters and Free Distributive Lattices, *Proceedings of the 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, Halkidiki, Greece, 1995.
13. I. Shmulevich, V. Melnik, K. Egiazarian, Optimization of Stack Filters Using Sample Selection Probabilities, *Proceedings of IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Antalya, Turkey, June 20-23, 1999.
14. I. Shmulevich, V. Melnik, K. Egiazarian, The Use of Sample Selection Probabilities for Stack Filter Design *IEEE Signal Processing Letters*, vol. 7, no. 7, pp. 189-192, July 2000.
15. Michael Elad and Arie Feuer, "Superresolution Restoration of an image sequence: Adaptive filtering approach," IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 8, NO. 3, MARCH 1999.
16. Haritopoulos, M., Yin, H., Allinson, N., "Image Denoising using SOM-based Nonlinear Independent Component Analysis", Neural Networks 15 (2002) pp.1085–1098, 2002.
17. Lappalainen, H., Honkela, A, "Bayesian nonlinear independent component analysis by multi-layer perceptrons", In Girolami, M., ed.: Advances in Independent Component Analysis, Springer-Verlag (2000) 93–121
18. Bell, A., Sejnowski, T., "An Information-maximization Approach to Blind Separation and Blind Deconvolution", Neural Computation 7 (1995), pp. 1129–1159, 1995.

19. Hyvarinen, A., Pajunen, P., “Nonlinear Independent Component Analysis: Existence and Uniqueness results”, Neural Networks 12 (1999) pp. 429–439, 1999.
20. Luis B. Almeida, “Separating a Real-life Nonlinear Image Mixture”, Journal of Machine Learning Research 6, pp. 1199-1230, 2005.
21. Luis B. and M. Faria, “Separating a Real-life Nonlinear Mixture of Image,” In Carlos G. Puntonet and Alberto Prieto, editors, Independent Component Analysis and Blind Signal Separation (Proc. ICA ’2004), number 3195 in Lecture Notes in Artificial Intelligence, pp.729–736, Springer-Verlag, Granada, Spain, 2004.
22. S. Amari, A. Cichocki, and H. H. Yang., “A New Learning Algorithm for Blind Signal Separation”, In David Touretzky, Michael Mozer, and Mark Hasselmo, editors, Advances in Neural Information Processing Systems 8, p. 757–763. MIT Press, 1996.
23. Y. Ye, J. Luo, “On Linear and Nonlinear Processing of Underwater, Ground, Aerial and Satellite Image,” pp. 3364-3368, 2005 IEEE International Conference on Systems, Man and Cybernetics Waikoloa, Hawaii October 10-12, 2005.
24. Z. Ye, and J. Luo, “Optimal Approach for Enhancement of Large and Small Scale Near-Infrared and Infrared Imaging”, IASTED, Oct. 31 – Nov. 2, 2005, Cambridge, USA., 2005.
25. J. Luo, Z. Ye, “Nonlinear Processing of Large Scale Satellite Images via Unsupervised Clustering and Image Segmentation”, WSEAS International Conference ICECS ’05, Miami, FL, USA, 2005.
26. J. Luo and J. Cross, “Content –Based (Unsupervised) Image Segmentation for Large – Scale Spatial Images (with MATLAB)”, SSST, March, 2006, TN, 2006.
27. David G. Long, Q. P. Remund and Douglas L. Daun, “A Cloud-removal Algorithm for SSM/I Data,” IEEE Trans. On Geoscience and Remote Sensing”, vol/ 37, No. 1, January, 1999, pp 54-62, 1999.
28. Barry Beckham, The Digital Photographer’s Guide to Photoshop Elements, Larks Books, New York, 2005.

X. APPENDIX

A1. Program to Recognize and to Determine the Sum of Coins in an Image

```
% Program coins_thresh3b_10_17_06
% Programming for thresholding Coins1a with the function mean2 included.
X = imread('Coins1a.jpg');%Coins1a; the file must be in the directory.
X = rgb2gray(X);
for i = 1:460  % Perform thresholding based on the mean value
for j = 1:620  % of a 20 by 20 set of pixels around the test pixel.
   if mean2(X(i:i+20,j:j+20)) < 120
  Xa(i:i+20,j:j+20) = 0;
else
Xa(i:i+20,j:j+20) = 1;
end
end
end
fid = fopen('sum.out', 'w'); % Create an output file for the sum of coins.
X5 = ~(Xa); % Note that [L, num] = bwlabel(Xa,8)will not operate work
    % because the above operation made all the objects in Xa to be black.
    % Using help bwlabel, we see that the background will be white for the
    % resulting output.
    N = 8;
[L, num] = bwlabel(X5,N); % Operator intervention might be required here
        % if segmentation is not successful. For some images, pre-
        % processing (filtering) may be required before thresholding.
% L = BWLABEL(X5,num) returns a matrix L, of the same size as X5,
% containing labels for the connected components in X5. N can have a value
% of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies
% 8-connected objects; if the argument is omitted, it defaults to 8.
% The elements of L are integer values greater than or equal to 0.  The
% pixels labeled 0 are the background.  The pixels labeled 1 make up one
% object, the pixels labeled 2 make up a second object, and so on.
p = 0;
for m = 1:num
[r,c] = find(L==m); % r will be returned as the row indices of L for the
% given object in L and c will be returned as the column indices.
r1(m) = min(r); % r1 min and max will be used to compute the X dimension.
r2(m) = max(r);
c1(m) = min(c); % c1 min and max will be used to compute the Y dimension.
c2(m) = max(c);
hor(m) = r2(m) - r1(m); % Determine the length in the X direction.
vert(m) = c2(m)-c1(m);  % Determine the length in the Y direction.
di(m) = ((hor(m))^2 + ((vert(m))^2)^.5); % % Use the 2 dimensions above
       % as the basis to compare the relative sizes of objects.
% Note that a small set of pixels connected together was labeled as
```

```
% objects. The small objects will be eliminated and only objects above
% a particular size will be retained. The decision on sizes to be retained
% was made by observing the output of L.
if (di(m) < 1000 |(di(m)> 10000)) % Omit all objects with a value of di
  % less than 1000 or greater than 10000.
object(m) = 0;
else
     p = p+1; % p will be the number of objects retained.
     object(p) = di(m);
     a1(p) = r1(m); % a1, a2, b1 and b2 will be used to locate and show
     a2(p) = r2(m); % the varies objects (sub-images).
     b1(p) = c1(m);
     b2(p) = c2(m);
end
end

v1 = find(di>1000 & di <10000); %v1 is a row vector, 1 to 12 for
%  this case but they may not be in order of size.
di = di(v1); % di will be a row vector showing all the (12) values of di.
v2 = find(di>1000 & di<10000); % v2 will now start at 1.
vm = max(max(v2)); %This should give a value of 12 for vm (for this case).
sum = 0;
for m =1:vm
% Use some method to classify the remaining objects according to size.
% Some of the classical methods could be used but a simple method
% was used instead. The various sizes were first observed to see the
% 3 patterns or size clusters of nickels, dimes and quarters.
% However, a second version of the program ran successfully that was
% "unsupervised". It used the relative sizes of the clusters
% to identify the coins so programmer intervention was not required.
if (di(m) > 5200 & di(m) < 6900)
sum = sum + .05; % It is a nickel.
else
   if (di(m) > 3000 & di(m) < 5000)
sum = sum + .1; % It is a dime.
   else
   if (di(m) > 7100 & di(m) < 8400)
sum = sum + .25; % It is a quarter.
   end
   end
end
end
% disp (sum);
sprintf('The total sum of coins is  $%5.2f',sum) % Output the sum screen.
fprintf(fid,'%s %4.2f\n','$',sum);
% Also, write the value to the output file sum.out.
```

```
fig1 = X5(a1(1):a2(1), b1(1):b2(1)); % Designate the various objects.
fig2 = X5(a1(2):a2(2), b1(2):b2(2));
fig3 = X5(a1(3):a2(3), b1(3):b2(3));
fig4 = X5(a1(4):a2(4), b1(4):b2(4));
fig5 = X5(a1(5):a2(5), b1(5):b2(5));
fig6 = X5(a1(6):a2(6), b1(6):b2(6));
fig7 = X5(a1(7):a2(7), b1(7):b2(7));
fig8 = X5(a1(8):a2(8), b1(8):b2(8));
fig9 = X5(a1(9):a2(9), b1(9):b2(9));
fig10 = X5(a1(10):a2(10), b1(10):b2(10));
fig11 = X5(a1(11):a2(11), b1(11):b2(11));
fig12 = X5(a1(12):a2(12), b1(12):b2(12));
subplot(2,6,1), imshow(fig1);subplot(2,6,2), imshow(fig2)
subplot(2,6,3), imshow(fig3);subplot(2,6,4), imshow(fig4)
subplot(2,6,5), imshow(fig5);subplot(2,6,6), imshow(fig6)
subplot(2,6,7), imshow(fig7);subplot(2,6,8), imshow(fig8)
subplot(2,6,9), imshow(fig9);subplot(2,6,10), imshow(fig10)
subplot(2,6,11), imshow(fig11);subplot(2,6,12), imshow(fig12)
```

A2. Program to Perform Segmentation Based on the Quadtree Decomposition Method Along with the Predicate Function

1. The Basic Function

```
% Program called my_splitmerge_qt_coins3b_bwp_if
% Function to return the perimeter is added. The imfill function is
% also used. The algorithm will be tested using the coin image.
% The predicate function uses the average and the standard deviation as
% the criteria for when to split the image with the Quadtree method.
X1 = imread('Coins1.jpg'); % The coin file must be in the directory.
X2 = rgb2gray(X1);
X3 = rot90(X2,3); % Make the horizontal axis the longest axis.
X4 = X3(60:570,1:455); % Perform cropping as follows:
            % 570-60 = 510 so it will be less than 512, the nearest power
            % of 2. It will be padded with zeros to give 512 by 512
XX = X4(150:260, 200:280); % The location of one of the coins was found
                  % and it will be used as the region for comparison.
% Note: The region for comparison can be chosen interactively by
% using the "region of interest" function roipoly. This function selects
% a polygonal region of interest within an image that can be
% used as a mask for masked filtering.
ave = average(XX) % The average function is not in the Image Processing
st = std2(XX)  % Toolbox. It is from Gonzalez, Woods and Eddins and must
             % be placed in the path (directory).
% imshow(XX)
 region= XX; % The following 2 equations will be given in the predicate
             % function and are not needed here.
 % sd_test = std2(region)
 % m_test = mean2(region)
 % region = region/255; %Normalizing it if you decide to use the histogram
```

```
flag = predicate(region); % A predicate function must be given written
   %and included as a separate program.
 [g,vals,r,c] = splitmerge(X4,2,@predicate); % Decompose X4. The splitmerge
 % function must be in the folder. The split-and-merge algorithm performs
 % image splitting and merging based on the
 % Quadtree decomposition approach.
 g = bwperim(g,4);
 g = imfill(g,'holes');
imshow(g)
```

2. The Predicate Function

```
function flag = predicate(region,m_test)
% Predicate function for the program called my_splitmerge_qt_coins3a
sd = std2(region);
m = mean2(region);
flag = ~((m > 100) & (m < 150)& (sd < 7));
```

A3. Functions for the Advanced Segmentation Method

1. hm_segment

```
clear all;

%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\images_other\airport.jpg');

%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\images_other\clouds.jpg');
%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\images_other\clouds.jpg');
%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\images_other\cloud_m.jpg');
Im = imread('C:\Documents and Settings\Jiecai Luo\My Documents\AI_wpafb\radar
image\dop.jpg');
Im=imresize(Im,[1024,1024]);

disp('********************************************** ');
T_name=input('The segmented image data name is  ','s');
Tol=input('please input the segmentation Thresholding value =');
disp('********************************************** ');
N=input('input the number of h_moment N =  ');
disp('********************************************** ');

n_samp=input('please input how many samples you want=');
figure
image(Im);
title('Original image in RGB space')
hold;

%set up the reference images
T_r=[];T_g=[];T_b=[];
for i_samp=1:1:n_samp;
rect=getrect;
```

```
nn_x=floor(rect(1));nn_y=floor(rect(2));
nnn_1=floor((rect(3)+rect(4))/2);
if nnn_1 <=4;
    nnn_1=4;
end;
nnn_2=nnn_1;
for ii=1:1:nnn_2;
    for jj=1:1:nnn_1;
        im_r(ii,jj,:)=Im(nn_y+ii,nn_x+jj,:);
    end;
end;
 [v_r,unv_r]=h_moments(imhist(im_r(:,:,1)),N);
 [v_g,unv_g]=h_moments(imhist(im_r(:,:,2)),N);
 [v_b,un v_b]=h_moments(imhist(im_r(:,:,3)),N);
T_r=[T_r; v_r];
T_g=[T_g; v_g];
T_b=[T_b; v_b];
x=[nn_x nn_x+nnn_1 nn_x+nnn_1 nn_x nn_x];
y=[nn_y nn_y  nn_y+nnn_2 nn_y+nnn_2 nn_y];
figure(1)
plot(x,y)
gtext(num2str(i_samp))
clear im_r;
end;

Im_o(:,:,1)=repmat(uint8(0),[1024 1024]);;
Im_o(:,:,2)=Im_o(:,:,1);
Im_o(:,:,3)=Im_o(:,:,1);
Im_o=Im;

% calculate the distances
S = qtdecomp(Im_o(:,:,1),.27);
   [inn,jnn,snn]=find(S);
   im=Im_o;
      N_length=length(jnn);
     n=1;
    for i=1:1:N_length;
        if Im_o(inn(i),jnn(i),1)~= 0;
            im_b(1:snn(i),1:snn(i),:)=...
            Im_o(inn(i):snn(i)+inn(i)-1,jnn(i):jnn(i)+snn(i)-1,:);
            [T_ro, unv_1]=h_moments(imhist(im_b(:,:,1)),N);
            [T_go, unv_2]=h_moments(imhist(im_b(:,:,2)),N);
            [T_bo, unv_3]=h_moments(imhist(im_b(:,:,3)),N);
            for i_samp=1:1:n_samp;
                  d_r = h_m_distance(T_r(i_samp,:), T_ro, N);
                d_g = h_m_distance(T_g(i_samp,:), T_go, N);
                d_b = h_m_distance(T_b(i_samp,:), T_bo ,N);
            dmm_all(i_samp)=0.55*d_r+0.25*d_g+0.2*d_b;
            end;
               d_all(n)=min(dmm_all);

            clear im_b;
            if d_all(n) > Tol
                im(inn(i):snn(i)+inn(i)-1,jnn(i):jnn(i)+snn(i)-1,:)=0;
            end;
            n=n+1;
        end;
```

```
    end;
figure
image(im);
save(T_name, 'Im_o', 'S', 'd_all')
```

2. h_m distance

```
function d = h_m_distance(v1,v2,n);

d=0;
for i=1:1:n;
    d=d+(v1(i)-v2(i))^2;
end;
d = sqrt(d);
```

3. h-moments

```
function [v, unv] = h_moments(p,n);
% See Digital Image Processing with MATLAB page 590
%%statmoments function
Lp = length(p);
if (Lp ~= 256) & (Lp ~= 65536)
    error('P must be a 256- or 65536- element vector.');
end;
G = Lp-1;
p = p/sum(p);p=p(:);
z = 0:G;
z = z./G; m = z*p;
z = z-m;
v = zeros(1,n);
v(1) = m;
for j = 2:n;
      v(j)=(z.^j)*p;
end;
if nargout > 1
    unv = zeros(1,n);
    unv(1) = m.*G;
    for j = 2:n
        unv(j) = ((z*G).^j)*p;
    end;
end;
```

4. hm_markareas

```
% This file is (1) to get color image data
% from the txt files, then filtering, and calculate
% the color image's histogram in different color space
% do contented based image segmentation by spatial color histogram match

clear;
disp('********************************************** ');
```

```
ls *.mat
new_result = input('input the data file you want to load ','s');
load(new_result);

im=Im_o;
im1=im;
disp('The mean value is')
[ min(d_all) mean2(d_all) max(d_all)]

disp('The std value is ')
std2(d_all)

figure
plot(sort(d_all),'.')
disp('please get the segmentation Thresholding value from the figure');

Tol=input('please input the thresholding value=');
N_want=input('please input the small object (to be removed) pixel’s value
=');

figure
image(Im_o);
title('Original image in RGB space')
hold;

  [inn,jnn,snn]=find(S);

   N_length=length(jnn);
   n=1;
    for i=1:1:N_length;
         if Im_o(inn(i),jnn(i),1)~= 0;
            if d_all(n)> Tol;
                im(inn(i):inn(i)+snn(i)-1,jnn(i):jnn(i)+snn(i)-1,:)=0;

            else
                im(inn(i):inn(i)+snn(i)-1,jnn(i):jnn(i)+snn(i)-1,:)=255;

            end;
            n=n+1;
         end;
    end;

 bworiginal=im(:,:,1);
 bw_600=bwareaopen(bworiginal,N_want);

 se=strel('disk',20);
 bw_600=imclose(bw_600,se);
 bw_600=imfill(bw_600,'holes');

 figure
 imshow(bw_600)

S1 = qtdecomp(bw_600,.27);
[inn1,jnn1,snn1]=find(S1);

  N_length1=length(jnn1);
  n_label=[];
```

```
     for i=1:1:N_length1;
         if bw_600(inn1(i),jnn1(i),1)~= 0;

             n_label=[n_label; inn1(i) jnn1(i) snn1(i)];
                     nn_x=inn1(i);nn_y=jnn1(i);nnn_1=snn1(i);nnn_2=snn1(i);
                    x=[nn_x nn_x+nnn_1 nn_x+nnn_1 nn_x nn_x];
                y=[nn_y nn_y  nn_y+nnn_2 nn_y+nnn_2 nn_y];
                im1(inn1(i):inn1(i)+snn1(i)-1,jnn1(i):jnn1(i)+snn1(i)-1,:)=0;
            figure(2)
            plot(y-1/2,x-1/2,'y')

         end;
    end;

figure
image(Im_o)

J=imfill(im1,'holes');
 figure
imshow(J);

disp('********************************************** ');
T_name=input(' To be filled image data name is  ','s');
save(T_name, 'Im_o', 'S1', 'bw_600','im1')
```

5. hm_areafilled

```
clear all;

disp('********************************************** ');
ls *.mat
new_result = input('input the data file you want to load ','s');
load(new_result);
%load mixed_data;
Im=imresize(im1,[1024,1024]);
I_new=Im;
n_samp = 1;

figure

image(Im);
title('Original image in RGB space')
hold;

%set up the reference images
for i_samp=1:1:n_samp;
rect=getrect;
nn_x=floor(rect(1));nn_y=floor(rect(2));

nnn_1=rect(3);
nnn_2=rect(4);
for ii=1:1:nnn_2;
    for jj=1:1:nnn_1;
        im_r(ii,jj,:)=Im(nn_y+ii,nn_x+jj,:);
    end;
```

```
end;

I1=im_r;

x=[nn_x nn_x+nnn_1 nn_x+nnn_1 nn_x nn_x];
y=[nn_y nn_y  nn_y+nnn_2 nn_y+nnn_2 nn_y];
figure(1)
plot(x,y)
gtext(num2str(i_samp))
clear im_r;
end;

figure, imshow(I1);

[inn1,jnn1,snn1]=find(S1);
N_length1=length(jnn1);
n_label=[];
  for i=1:1:N_length1;
     if bw_600(inn1(i),jnn1(i),1)~= 0;
       n_label=[n_label; inn1(i) jnn1(i) snn1(i)];
       nn_x=inn1(i);nn_y=jnn1(i);nnn_1=snn1(i);nnn_2=snn1(i);
       I2=Im_o(inn1(i):inn1(i)+snn1(i)-1,jnn1(i):jnn1(i)+snn1(i)-1,:);
    for jj=1:1:3;
    [n1,m1]=size(I1(:,:,jj));
    h1=imhist(I1(:,:,jj))*100/(n1*m1);
    for i=1:1:256
        h1_s(i)=sum(h1(1:i));
    end;
        I_h1_b = find(h1_s == 0);
        I_h1_f = find(h1_s > 99.99);

    if length(I_h1_b)>0
        n_1_b = I_h1_b(end);
    else
        n_1_b=1;
    end;
    if length(I_h1_f) >0
        n_1_f = I_h1_f(1);
    else
        n_1_f=256;
    end;

    [n2,m2]=size(I2(:,:,jj));
    h2=imhist(I2(:,:,jj))*100/(n2*m2);
    for i=1:1:256
        h2_s(i)=sum(h2(1:i));
    end;

    [n0 m0]=size(I2(:,:,1));
    P = zeros(n0,m0);
    for i=n_1_b:n_1_f-1;
        I_temp = find(h2_s >= h1_s(i) & h2_s <= h1_s(i+1));
        if length(I_temp) > 0;
        P=P+ i*(I2(:,:,jj)>=I_temp(1) & I2(:,:,jj)<=I_temp(end));
        end;
    end;
        H = fspecial('average',[6,6]);
```

```
        I3(:,:,jj)=imfilter(P/256,H,'replicate');
        clear h1_s h1 h2_s h2;
   end;

    for ii=1:1:nnn_2;
        for jj=1:1:nnn_1;
            I_new(nn_x+ii,nn_y+jj,:)=im2uint8(I3(ii,jj,:));
        end;
    end;

 clear I2 I3;
 end;
end;
figure,imshow(I_new);
```

6. hm_finalsmoothed

```
% this code is for final image smoothing

clear all;

%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\images_other\clouds.jpg');
%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\radar image\coins_1.jpg');
Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\images_other\seg_cloud\clouds_new.jpg');
%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\radar image\dop.jpg');
%Im = imread('C:\Documents and Settings\Jiecai Luo\My
Documents\AI_wpafb\radar image\radar_1.jpg');
Im=imresize(Im,[1024,1024]);
I_new=Im;
n_samp = 2;

figure
image(Im);
title('Original image in RGB space')
hold;

%set up the reference images
for i_samp=1:1:n_samp;
rect=getrect;
nn_x=floor(rect(1));nn_y=floor(rect(2));
nnn_1=rect(3);
nnn_2=rect(4);
for ii=1:1:nnn_2;
    for jj=1:1:nnn_1;
        im_r(ii,jj,:)=Im(nn_y+ii,nn_x+jj,:);
    end;
end;

if i_samp ==1;
    I1=im_r;
else
```

```
     I2=im_r;
end;

x=[nn_x nn_x+nnn_1 nn_x+nnn_1 nn_x nn_x];
y=[nn_y nn_y  nn_y+nnn_2 nn_y+nnn_2 nn_y];
figure(1)
plot(x,y)
gtext(num2str(i_samp))
clear im_r;
end;
figure, imshow(I1);
figure, imshow(I2);

for jj=1:1:3;
[n1,m1]=size(I1(:,:,jj));
h1=imhist(I1(:,:,jj))*100/(n1*m1);
for i=1:1:256
    h1_s(i)=sum(h1(1:i));
end;
I_h1_b = find(h1_s == 0);
I_h1_f = find(h1_s > 99.99);

if length(I_h1_b)>0
n_1_b = I_h1_b(end);
else
    n_1_b=1;
end;
if length(I_h1_f) >0
n_1_f = I_h1_f(1);
else
   n_1_f=256;
end;
[n2,m2]=size(I2(:,:,jj));
h2=imhist(I2(:,:,jj))*100/(n2*m2);
for i=1:1:256
    h2_s(i)=sum(h2(1:i));
end;

[n0 m0]=size(I2(:,:,1));
P = zeros(n0,m0);
for i=n_1_b:n_1_f-1;
    I_temp = find(h2_s >= h1_s(i) & h2_s <= h1_s(i+1));
    if length(I_temp) > 0;
    P=P+ i*(I2(:,:,jj)>=I_temp(1) & I2(:,:,jj)<=I_temp(end));
    end;
end;
H = fspecial('average',[6,6]);
I3(:,:,jj)=imfilter(P/256,H,'replicate');
clear h1_s h1 h2_s h2;
end;

I3=im2uint8(I3);
figure, imshow(I3);

[n1,m1]=size(I1(:,:,1));
h1=imhist(I1(:,:,1))*100/(n1*m1);
for i=1:1:256
```

```
    h1_s(i)=sum(h1(1:i));
end;

[n2,m2]=size(I2(:,:,1));
h2=imhist(I2(:,:,1))*100/(n2*m2);
for i=1:1:256
    h2_s(i)=sum(h2(1:i));
end;

[n3,m3]=size(I3(:,:,1));
h3=imhist(I3(:,:,1))*100/(n3*m3);
for i=1:1:256
    h3_s(i)=sum(h3(1:i));
end;
figure,
subplot(3,1,1)
plot(h1)
hold; plot(h1_s,'r')

subplot(3,1,2), plot(h2)
hold;plot(h2_s,'r')

subplot(3,1,3), plot(h3)
hold;plot(h3_s,'r')

for ii=1:1:nnn_2;
    for jj=1:1:nnn_1;
        I_new(nn_y+ii,nn_x+jj,:)=I3(ii,jj,:);
    end;
end;
figure,imshow(I_new);

I4(:,:,1)=histeq(I3(:,:,1));
I4(:,:,2)=histeq(I3(:,:,2));
I4(:,:,3)=histeq(I3(:,:,3));
figure, imshow(I4)
```

A4. Program to Embed a Message in an Image

```
% Program indx2rgbchar2a  To write a message and imbedded it in an image.
% The matrix with the message will be called Y1a.
% This will start with program indx2rgbchar which is documented as follows:
% Program to convert an N x M index image (that has a map) into a
% RGB image N x M x 3 (without a map).
% Then change the pixel values for green. A message will be imbedded
% in the image by modifying the green color.
% This program assume a 200 x 320 image has been loaded (or read in) and
% the data is in a default matrix, X or the matrix is named X.
% If it is not 200 x 320, changed the values of i and j below.
load('clown')
% This will convert the matrix for the clown image.
for i = 1:200
        for j = 1:320
```

```
            k = X(i,j);
              Y(i,j,1:3) = map(k,1:3);
        end
end
Y1 = double(Y);
% Now write the message and change
% and create a vector char1 with the corresponding number for the
% letter such as A = 1, B = 2, ... ? = 30.
message2 = 'GOD BLESS AMERICA' ;
   k1 = length(message2);
   for n1 = 1:k1
       if message2(n1) == 'A'
            char1(n1) = 1;
       elseif message2(n1) == 'B'
           char1(n1) = 2;
       elseif message2(n1) == 'C'
           char1(n1) = 3;
       elseif message2(n1) == 'D'
           char1(n1) = 4;
            elseif message2(n1) == 'E'
           char1(n1) = 5;
       elseif message2(n1) == 'F'
           char1(n1) = 6;
       elseif message2(n1) == 'G'
           char1(n1) = 7;
           elseif message2(n1) == 'H'
           char1(n1) = 8;
       elseif message2(n1) == 'I'
           char1(n1) = 9;
       elseif message2(n1) == 'J'
           char1(n1) = 10;
            elseif message2(n1) == 'K'
           char1(n1) = 11;
       elseif message2(n1) == 'L'
           char1(n1) = 12;
       elseif message2(n1) == 'M'
           char1(n1) = 13;
       elseif message2(n1) == 'N'
           char1(n1) = 14;
       elseif message2(n1) == 'O'
           char1(n1) = 15;
       elseif message2(n1) == 'P'
           char1(n1) = 16;
            elseif message2(n1) == 'Q'
           char1(n1) = 17;
       elseif message2(n1) == 'R'
```

```
        char1(n1) = 18;
    elseif message2(n1) == 'S'
        char1(n1) = 19;
        elseif message2(n1) == 'T'
        char1(n1) = 20;
    elseif message2(n1) == 'U'
        char1(n1) = 21;
    elseif message2(n1) == 'V'
        char1(n1) = 22;
    elseif message2(n1) == 'W'
        char1(n1) = 23;
    elseif message2(n1) == 'X'
        char1(n1) = 24;
    elseif message2(n1) == 'Y'
        char1(n1) = 25;
        elseif message2(n1) == 'Z'
        char1(n1) = 26;
    elseif message2(n1) == ' '
        char1(n1) = 27;
    elseif message2(n1) == '.'
        char1(n1) = 28;
         elseif message2(n1) == ','
        char1(n1) = 29;
    elseif message2(n1) == '?'
        char1(n1) = 30;
    else
        break
    end
end
char1;
% Now change the whole numbers into fractions to be imbedded as pixels
% for the color green in the image Y1.
char = .03*double(char1);
% Call the matrix that has the message Y1a.
Y1a = Y1;
  Y1a(20,10:k1+9,2) = char;
% Y1a now has the message.
  % Save Y1a and k1 to the workspace
  save Y1a
save k1
imshow(Y1a)
```

A5. Program to Recover the Message From the Image as Described in A4.

```
% Program indx2rgbchar2   This program recovers the message from
% Program indx2rgbchar2a
```

```
% The purpose of Program indx2rgbchar2a was as follows:
% Program to convert an N x M index image (that has a map) into a
% RGB image N x M x 3 (without a map).
% Then change the pixel values for green. A message will be imbedded
% in the image by modifying the green color.
% This program assume a 200 x 320 image has been loaded (or read in) and
% the data is in a default matrix, X or the matrix is named X.
 % If it is not 200 x 320, changed the values of i and j below.
% The matrix for the image in which the message was embedded using
% the above process was called Y1a.
% The imbedded message was imbedded from a vector char that had
% 17 characters using the statement: Y1(20,10:k1+9,2) = char;
% For this message, k1 was 17 so the message was in the matrix
% row 20, columns 10 through 26 for the color green.
% We will follow the previous method used to read the message which
% had the following documentation:
% Now use a modified version of program testmess1
% Where as that program read in a number,
% we will get the number for the letter or other character
% using the above program.
% We will then write the letter, a space, a
% period, a comma, or a question mark
% based on the letter being between 1 and 30.
% i = input ('Input a number between 1 and 30. \n'); Don't do this.
% We will load the matrix called Y1a that has the message and was save
% to the workspace. We used Y1a(20,10:k1+9,2) = char;
load('Y1a');
fid = fopen('messfile2.out', 'w'); % Create the output file messfile2.out
% We enter the value for k1 here but we saved it in the workspace from
% the embedding program.
k1 = 17;
char =Y1a(20,10:k1+9,2);
% char is a matrix having numbers from .03*(1) through .03*(30).
% So divide by .03 to get a matrix num2 having values between
% 1 and 30. Use rounding to make certain it is a whole number.
num2 = round(char/.03);
for j1 = 1:k1
   i = num2(j1)
if i < 1
   break
elseif i < 11
     char1 = str2mat('A', 'B', 'C', 'D', 'E','F', 'G', 'H', 'I', 'J');
        char = char1(i);
elseif i < 21
     i = i -10;
   char2 = str2mat('K', 'L', 'M', 'N', 'O','P', 'Q', 'R', 'S', 'T');
```

```
        char = char2(i);
elseif i < 31
      i = i - 20;
   char3 = str2mat('U', 'V', 'W', 'X', 'Y','Z', ' ', '.', ',', '?');
        char = char3(i);
else
   break
end
letter(j1) = char;
end
letter
 fprintf(fid,'%s\n',letter);
% image(Y1a)
fclose(fid)
```