Contents

Chapter 1

Part - I

1.1 Read and Display Images

Writing and testing a MATLAB script that perform the following:
Read [1] and display [2] RGB and gray image.

1.2 MATLAB Code

```
%% Function where our program starts
function asg1_part_1()
%---------------------------------------------------------------------
% USAGE: asg1_part_1
% Output: Displays the RGB and Grayscale images
%---------------------------------------------------------------------
    % Given the Image filename/path
    pic1_name = 'lena_color.tiff';
    pic2_name = 'lena_gray.pgm';
    % Calls the imagecheck function to check image type
    img1_type = imagecheck(pic2_name);
    % Reads the image specified the image name
    var_gmat = imread(pic2_name,'pgm');
    % Calls the imagecheck function to check image type
    img2_type = imagecheck(pic1_name);
    % Reads the image specified the image name
    var_cmat = imread(pic1_name,'tiff');
    % Calls the displayimages function to plot the data/images
    displayimages(var_cmat,img2_type,var_gmat,img1_type);
end
```

```
%% Function to check Image Type
function[img_check] = imagecheck(var_img)
%--------------------------------------------------------------------
% USAGE: x = imagecheck(image_path);
% Inputs: image_path = Image filename/URL
% Output: x = Type of the Image('GrayScale Image'/'Color Image')
%--------------------------------------------------------------------
    % var_info get information about the image file
    var_info = imfinfo(var_img);
    % Checks if the image is grayscale/color and return image type
    if(strcmp(var_info.ColorType,'grayscale'))
        img_check = 'GrayScale Image';
    elseif(strcmp(var_info.ColorType,'truecolor'))
        img_check = 'Color Image';
    else
        img_check = 'Invalid';
    end
end

%% Function to display images
function displayimages(var1,var2,var3,var4)
%--------------------------------------------------------------------
% USAGE: displayimages(pic1,pic1_type,pic2,pic2_type);
% Inputs: var1 = Matrix of the Image
%         var2 = Image type of Image var1
%         var3 = Matrix of the Image
%         var4 = Image type of Image var3
% Output: this function does not return anything
%--------------------------------------------------------------------
    % created the subplots to display the images
    subplot(1,2,1);
    % Dispalys the image
    imshow(var1);
    title(var2);
    % creates tiles like position
    subplot(1,2,2);
    imshow(var3);
    % Shows the title of the plot
    title(var4);
end
```

1.3 MATLAB Output

Both the RGB and Grayscale images are read and displayed in the plot as below:

Figure 1.1: RGB and Grayscale images.

Chapter 2

Part - II

2.1 Enhancing image using stretching

Writing and testing a MATLAB script that perform the following:
Enhancing the image using stretching [3] for gray-scale/RGB.

2.2 MATLAB Code

```
%% Function where our program starts
function asg1_part_2()
%---------------------------------------------------------------------
% USAGE: asg1_part_2
% Output: Displays the enhanced image by linear stretching
%---------------------------------------------------------------------
    % Given the Image filename/path
    pic1_name = 'yose05.jpg';
    pic2_name = 'BAND1.pgm';
    % Calls the imagecheck function to check image type
    img1_type = imagecheck(pic2_name);
    % Reads the image specified the image name
    var_gmat = imread(pic2_name,'pgm');
    % Calls the imagecheck function to check image type
    img2_type = imagecheck(pic1_name);
    % Reads the image specified the image name
    var_cmat = imread(pic1_name,'jpg');
    % Performing linear stretching for grayscale/RGB image
    enhanceImageByStretching(img1_type,var_gmat);
    enhanceImageByStretching(img2_type,var_cmat);
```

```
end

%% Function to check Image Type
function[img_check] = imagecheck(var_img)
%-----------------------------------------------------------------
% USAGE: x = imagecheck(image_path);
% Inputs: image_path = Image filename/URL
% Output: x = Type of the Image('GrayScale Image'/'Color Image')
%-----------------------------------------------------------------
    % var_info get information about the image file
    var_info = imfinfo(var_img);
    % Checks if the image is grayscale/color and return image type
    if(strcmp(var_info.ColorType,'grayscale'))
        img_check = 'GrayScale Image';
    elseif(strcmp(var_info.ColorType,'truecolor'))
        img_check = 'Color Image';
    else
        img_check = 'Invalid';
    end
end

%% Function to enhance image by linear stretching
function enhanceImageByStretching(var1,var2)
%-----------------------------------------------------------------
% USAGE: enhanceImageByStretching(typeOfImage,MatOfImage);
% Inputs: var1 = Type of Image
%         var2 = Matrix of Image
% Output: This function does not return anything
%-----------------------------------------------------------------
    figure
    if(strcmp(var1,'GrayScale Image'))
        % Plots the data/images
        subplot(2,2,1); imshow(var2);
        title('Original Image');
        subplot(2,2,2); imhist(var2);
        title('Histogram of Original Image');
        % Used imadjust() to perform linear streatching
        output_var = imadjust(var2);
        subplot(2,2,3); imshow(output_var);
        title('Enhanced Image');
        subplot(2,2,4); imhist(output_var);
```

```
        title('Histogram of Enhanced Image');
    end
    if(strcmp(var1,'Color Image'))
        rchannel = var2(:, :, 1);
        gchannel = var2(:, :, 2);
        bchannel = var2(:, :, 3);
        subplot(4,4,[1:2 5:6]); imshow(var2);
        title('Original Image');
        subplot(4,4,3); imhist(rchannel);
        title('Red Channel - Original Image');
        subplot(4,4,4); imhist(gchannel);
        title('Green Channel - Original Image');
        subplot(4,4,7); imhist(bchannel);
        title('Blue Channel - Original Image');
        output_var = imadjust(var2, [.0 .0 .0; .6 .8 .8], []);
        output_red = output_var(:, :, 1);
        output_green = output_var(:, :, 2);
        output_blue = output_var(:, :, 3);
        subplot(4,4,[9:10 13:14]); imshow(output_var);
        title('Enhanced Image');
        subplot(4,4,11); imhist(output_red);
        title('Red Channel - Enhanced Image');
        subplot(4,4,12); imhist(output_green);
        title('Green Channel - Enhanced Image');
        subplot(4,4,15); imhist(output_blue);
        title('Blue Channel - Enhanced Image');
    end
end
```

2.3 MATLAB Output

Grayscale images are enhanced using stretching and plotted the histograms as below:

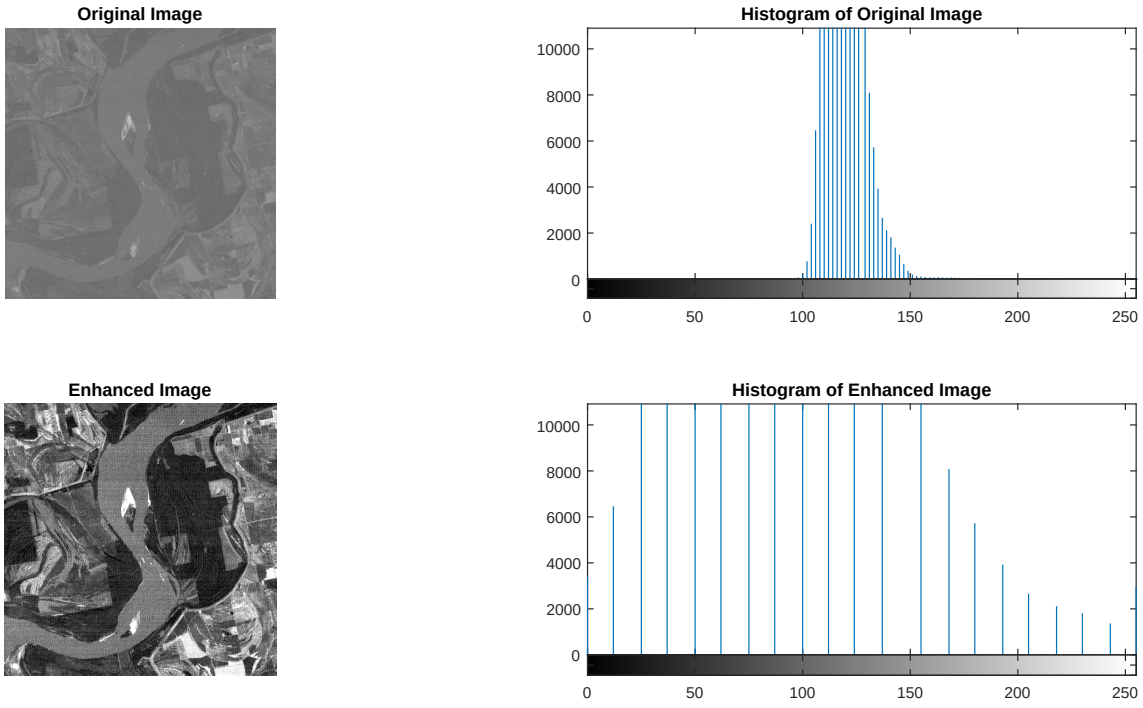

Figure 2.1: Grayscale input image.

RGB images are enhanced using stretching and plotted the histograms as below:

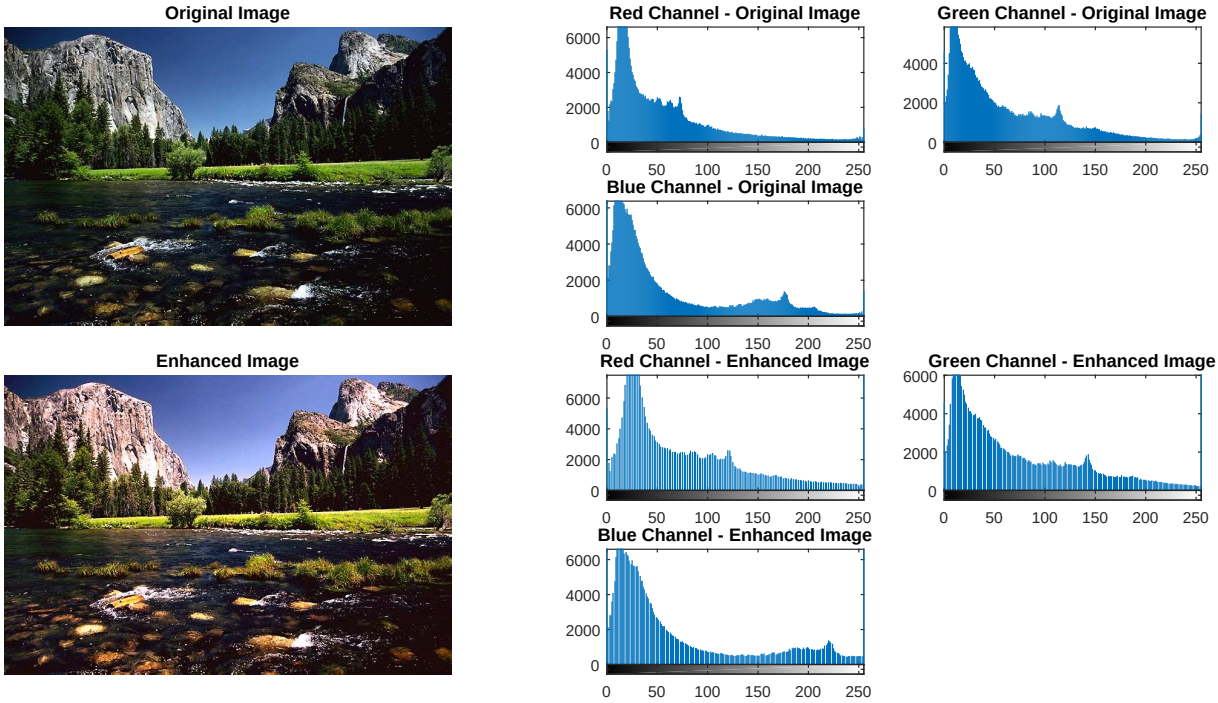

Figure 2.2: RGB input image.

Chapter 3

Part - III

3.1 Histogram Equalization

Writing and testing a MATLAB script that perform the following:
Enhancing the images using histogram equalization [4] and plotting histograms, and cumulative distribution functions (CDF) [5], before and after enhancement for grayscale/RGB images.

3.2 MATLAB Code

```
%% Function where our program starts
function asg1_part_3()
%--------------------------------------------------------------------
% USAGE: asg1_part_3
% Output: Displays the enhanced image by Histrogram Equalization
%--------------------------------------------------------------------
    % Given the Image filename/path
    pic1_name = 'yose05.jpg';
    pic2_name = 'BAND1.pgm';
    % Calls the imagecheck function to check image type
    img1_type = imagecheck(pic2_name);
    % Reads the image specified the image name
    var_gmat = imread(pic2_name,'pgm');
    % Calls the imagecheck function to check image type
    img2_type = imagecheck(pic1_name);
    % Reads the image specified the image name
    var_cmat = imread(pic1_name,'jpg');
```

```
    % Performing linear stretching for grayscale/RGB image
    enhanceImageByHisteq(img1_type,var_gmat);
    enhanceImageByHisteq(img2_type,var_cmat);
end

%% Function to check Image Type
function[img_check] = imagecheck(var_img)
%--------------------------------------------------------------------
% USAGE: x = imagecheck(image_path);
% Inputs: image_path = Image filename/URL
% Output: x = Type of the Image('GrayScale Image'/'Color Image')
%--------------------------------------------------------------------
    % var_info get information about the image file
    var_info = imfinfo(var_img);
    % Checks if the image is grayscale/color and return image type
    if(strcmp(var_info.ColorType,'grayscale'))
        img_check = 'GrayScale Image';
    elseif(strcmp(var_info.ColorType,'truecolor'))
        img_check = 'Color Image';
    else
        img_check = 'Invalid';
    end
end

%% Function to enhance image by linear stretching
function enhanceImageByHisteq(var1,var2)
%--------------------------------------------------------------------
% USAGE: enhanceImageByHisteq(typeOfImage,MatOfImage);
% Inputs: var1 = Type of Image
%         var2 = Matrix of Image
% Output: This function does not return anything
%--------------------------------------------------------------------
    figure
    if(strcmp(var1,'GrayScale Image'))
        % Plots the data/images
        subplot(2,3,1); imshow(var2);
        title('Original Image');
        subplot(2,3,2); imhist(var2,256);
        title('Histogram of Original Image');
        subplot(2,3,3); plot(cumsum(imhist(var2,256)));
        title('CDF of Original Image');
```

```
    % Used imadjust() to perform linear streatching
    output_var = histeq(var2);
    subplot(2,3,4); imshow(output_var);
    title('Enhanced Image');
    subplot(2,3,5); imhist(output_var,256);
    title('Histogram of Enhanced Image');
    subplot(2,3,6); plot(cumsum(imhist(output_var,256)));
    title('CDF of Enhanced Image');
end
if(strcmp(var1,'Color Image'))
    rchannel = var2(:, :, 1);
    gchannel = var2(:, :, 2);
    bchannel = var2(:, :, 3);
    subplot(4,6,[1:2 7:8]); imshow(var2);
    title('Original Image');
     subplot(4,6,3); plot(cumsum(imhist(rchannel)),'red');
    title('CDF Red Channel');
    subplot(4,6,4); plot(cumsum(imhist(gchannel)),'green');
    title('CDF Green Channel');
    subplot(4,6,9); plot(cumsum(imhist(bchannel)),'blue');
    title('CDF blue Channel');
    subplot(4,6,5); imhist(rchannel);
    title('Red Channel');
    subplot(4,6,6); imhist(gchannel);
    title('Green Channel');
    subplot(4,6,11); imhist(bchannel);
    title('Blue Channel');
    output_red = histeq(var2(:, :, 1));
    output_green = histeq(var2(:, :, 2));
    output_blue = histeq(var2(:, :, 3));
    outpic = cat(3,output_red,output_green,output_blue);
    subplot(4,6,[13:14 19:20]); imshow(outpic);
    title('Enhanced Image');
    subplot(4,6,15); plot(cumsum(imhist(output_red)),'red');
    title('CDF Red Channel');
    subplot(4,6,16); plot(cumsum(imhist(output_green)),'green');
    title('CDF Green Channel');
    subplot(4,6,21); plot(cumsum(imhist(output_blue)),'blue');
    title('CDF blue Channel');
    subplot(4,6,17); imhist(output_red);
    title('Red Channel');
```

```
        subplot(4,6,18); imhist(output_green);
        title('Green Channel');
        subplot(4,6,23); imhist(output_blue);
        title('Blue Channel');
    end
end
```

3.3 MATLAB Output

Grayscale images are enhanced using histogram equalization and plotted the histograms and cumulative distributive function as below:

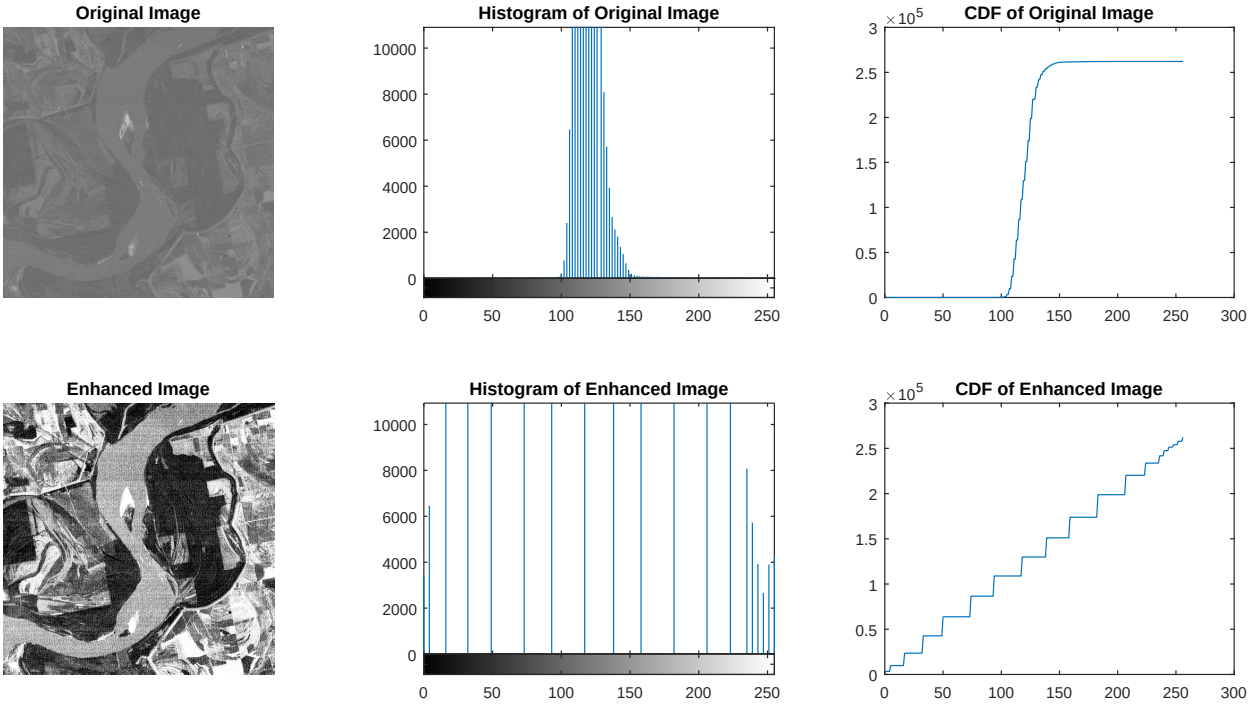

Figure 3.1: Grayscale input image.

RGB images are enhanced using histogram equalization and plotted the histograms and cumulative distributive function as below:

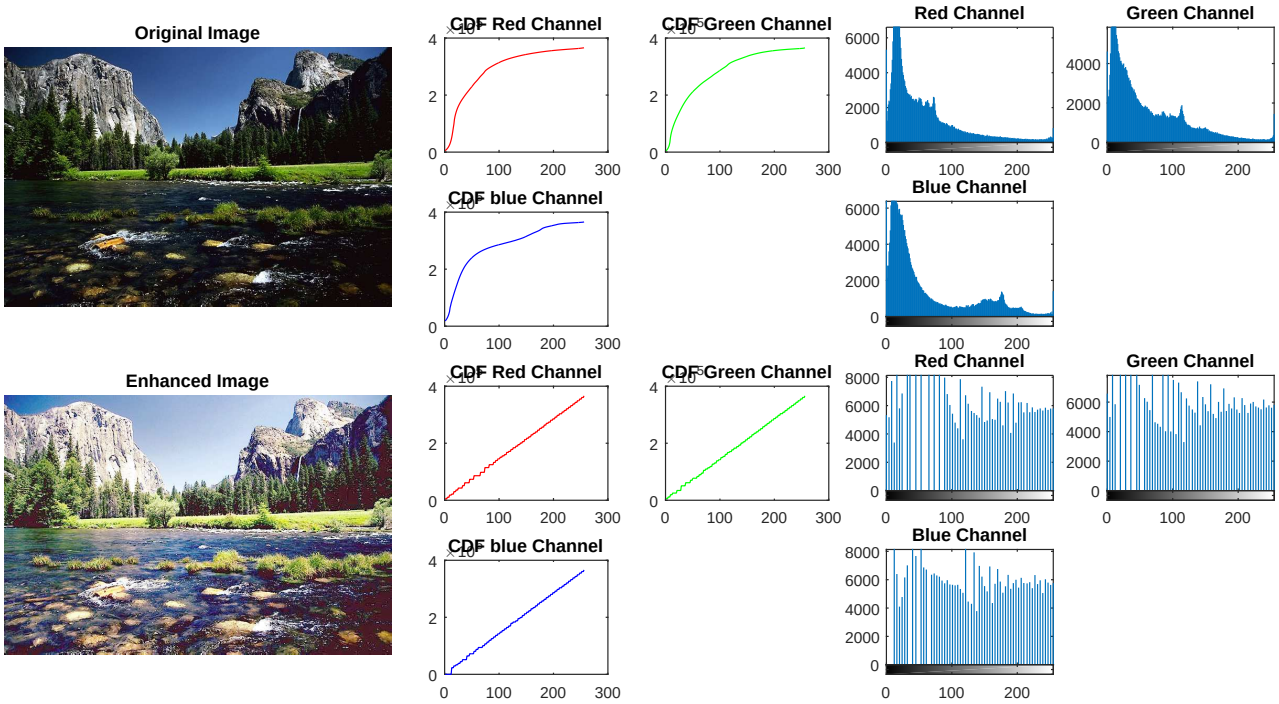

Figure 3.2: RGB input image.

Chapter 4

Part - IV

4.1 Median Filtering

Writing and testing a MATLAB script that perform the following:
Adding pepper and salt noise [6] to the image and using median filtering [7] to remove the noise. Displayed images with and without noise.

4.2 MATLAB Code

```
%% Function where our program starts
function asg1_part_4()
%-------------------------------------------------------------------
% USAGE: asg1_part_4
% Output: Adds and removes noise from the image
%-------------------------------------------------------------------
    % Given the Image filename/path
    pic1_name = 'lena_color.tiff';
    pic2_name = 'lena_gray.pgm';
    % Calls the imagecheck function to check image type
    img1_type = imagecheck(pic2_name);
    % Reads the image specified the image name
    var_gmat = imread(pic2_name,'pgm');
    % Calls the imagecheck function to check image type
    img2_type = imagecheck(pic1_name);
    % Reads the image specified the image name
    var_cmat = imread(pic1_name,'tiff');
    % Performing linear stretching for grayscale/RGB image
    [grayOutput, rgbOutput] = processImg(img1_type,var_gmat);
```

```
    [grayOutput_1, rgbOutput_1] = processImg(img2_type,var_cmat);
    subplot(2,2,1); imshow(grayOutput);
    title('Grayscale - With Noise(salt & pepper)');
    subplot(2,2,2); imshow(rgbOutput);
    title('Grayscale - Without Noise');
    subplot(2,2,3); imshow(grayOutput_1);
    title('RGB Image - With Noise(salt & pepper)');
    subplot(2,2,4); imshow(rgbOutput_1);
    title('RGB Image - Without Noise');
end

%% Function to check the type of image
function[img_check] = imagecheck(var_img)
%-------------------------------------------------------------------
% USAGE: x = imagecheck(image_path);
% Inputs: image_path = Image filename/URL
% Output: x = Type of the Image('GrayScale Image'/'Color Image')
%-------------------------------------------------------------------
    % var_info get information about the image file
    var_info = imfinfo(var_img);
    % Checks if the image is grayscale/color and return image type
    if(strcmp(var_info.ColorType,'grayscale'))
        img_check = 'GrayScale Image';
    elseif(strcmp(var_info.ColorType,'truecolor'))
        img_check = 'Color Image';
    else
        img_check = 'Invalid';
    end
end

%% Function to process the image
function[gOut, cOut] = processImg(var1,var2)
%-------------------------------------------------------------------
% USAGE: processImg(typeOfImage,MatOfImage);
% Inputs: var1 = Type of Image
%         var2 = Matrix of Image
% Output: gOut - Image with noise
%         cOut - Image without noise
%-------------------------------------------------------------------
    if(strcmp(var1,'GrayScale Image'))
        gOut = imnoise(var2,'salt & pepper');
```

```
        cOut = medfilt2(gOut);
    end
    if(strcmp(var1,'Color Image'))
        rchan = var2(:, :, 1);
        gchan = var2(:, :, 2);
        bchan = var2(:, :, 3);
        gOut1 = imnoise(rchan,'salt & pepper');
        gOut2 = imnoise(gchan,'salt & pepper');
        gOut3 = imnoise(bchan,'salt & pepper');
        gOut = cat(3,gOut1,gOut2,gOut3);
        cOut1 = medfilt2(gOut1);
        cOut2 = medfilt2(gOut2);
        cOut3 = medfilt2(gOut3);
        cOut = cat(3,cOut1,cOut2,cOut3);
    end
end
```

4.3 MATLAB Output

Grayscale and RGB images are added with with noise and removed the output is as below:

Figure 4.1: Noise added by salt & pepper and removed by median filtering.

References

[1] Documentation. (2016) Matlab - read image from graphics file. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/matlab/ref/imread.html

[2] Documentation. (2016) Matlab - display image. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/matlab/ref/imshow.html

[3] Documentation. (2016) Matlab - adjust image intensity values or colormap. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/images/ref/imadjust.html

[4] Documentation. (2016) Matlab - enhance contrast using histogram equalization. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/images/ref/histeq.html

[5] Documentation. (2016) Matlab - cumulative sum. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/matlab/ref/cumsum.html

[6] Documentation. (2016) Matlab - add noise to image. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/images/ref/imnoise.html

[7] Documentation. (2016) Matlab - 2-d median filtering. [Online; accessed 16-June-2016]. [Online]. Available: http://www.mathworks.com/help/images/ref/medfilt2.html