UNIVERSITY OF GONDAR

FACULTY OF NATURAL AND COMPUTATIONAL SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

PROJECT REPORT

Project Title: Image Filtering

PREPARED BY:

1. **Yemisrach Getinet**

Submitted date: August 12, 2013
Submitted to: Dr. Million

This document is the Project report of the image filtering prepared for Multimedia MSc. Course project.

Table of Contents

1. Introduction 3
2. Problem 4
3. Noise addition and Filtering Techniques and how they work 5
   3.1. Noise Addition 5
      3.1.1. Speckle Noise 5
      3.1.2. Gaussian Noise 5
      3.1.3. Salt & Pepper Noise 6
   3.2. Filtering Techniques 6
      3.2.1. Linear Filtering 9
         3.2.1.1. Average filtering 9
         3.2.1.2. Gaussian filtering 11
      3.2.2. Non-linear Filtering 12
         3.2.2.1. Median Filter 12
         3.2.2.2. Adaptive filtering 14
      3.2.3. Others 15
         3.2.3.1. BWAREAOPEN (Binary area open) 15
         3.2.3.2. conv2 or filter2 15
         3.2.3.3. Minimum Filtering 16
         3.2.3.4. Maximum Filtering 16
      3.2.4. Edge detection 16
4. Performance Evaluation 18
5. Discussion 19
6. Strategy 20
7. Sample Code 23
8. Concluding remark 29
9. Recommendation 29
10. Reference 30

1. Introduction

Images are Discrete Signals! Signal is visible light (the scene radiance). Since Images are Signals, they can be corrupted, Subject to random and additive noise (e.g. from electronics). Image Enhancement is one of the aspects of image processing in which the result is more suitable for a particular application. (Sharpening or de-blurring an out of focus image, highlighting edges, improving image contrast, or brightening an image, removing noise). Therefore, given a noise corrupted image, attenuating the noise using signal processing and then minimizing the impact of the noise (both random and impulsive noise) on the true signal using many filtering techniques and also adding various types of noise to an image to simulate the effects of some of the problems listed below is the main concern of this project.

2. Problem

Digital images are prone to a variety of types of noise. Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created.

For example:

- If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.
- If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise.
- Electronic transmission of image data can introduce noise.
- External noise in Analog-Digital conversion

Common types of Noises

1. **Amplifier noise (Gaussian noise)**

   The standard model of amplifier noise is additive, Gaussian, independent at each pixel and independent of the signal intensity. Amplifier noise is a major part of the "**read noise**" of an image sensor, that is, of the constant noise level in dark areas of the image. In color cameras where more amplification is used in the blue color channel than in the green or red channel, there can be more noise in the blue channel.

2. **Salt and pepper noise:**

   It represents itself as randomly occurring white and black pixels. Salt and pepper noise creeps into images in situations where quick transients, such as faulty switching, take place. The image after distortion from salt and pepper noise looks like the image attached. An image containing salt-and-pepper noise will have dark pixels in bright regions and bright pixels in dark regions. This type of noise can be caused by **analog-to-digital converter errors**, bit errors in transmission, etc. It can be mostly eliminated by using dark frame subtraction and interpolating around dark/bright pixels.

3. **Film grain**

   The grain of photographic film is a signal-dependent noise. If film grains are uniformly distributed (equal number per area), and if each grain has an equal and independent probability of developing to a dark silver grain after absorbing photons, then the number of such dark grains in an area will be random with a binomial distribution.

4. **Speckle noise**

   Speckle noise is a granular noise that inherently exists in and degrades the quality of images. Speckle noise is a multiplicative noise, i.e. it is in direct proportion to the local grey level in any area. The signal and the noise are statistically independent of each other.

3. Noise addition and Filtering Techniques and how they work

3.1. Noise Addition

3.1.1. Speckle Noise

Syntax:

J = imnoise (I,'speckle',V) adds multiplicative noise to the image I, using the equation J = I + n*I, where n is uniformly distributed random noise with mean 0 and variance V. The default for V is 0.04. Speckle Noise can add more noise than salt and pepper noise.

Example:

1. Read in the image and display it.
   c=imread(‘pic_0.bmp');

   imshow(c);

2. Add noise to it and display the result
   k = imnoise(c, 'Speckle', 1);

   imshow(c),title('original'), figure, imshow(k), title(‘with speckle noise');

3.1.2. Gaussian Noise

Syntax:

J = imnoise(I,'gaussian',M,V) adds Gaussian white noise of mean M and variance V to the image I. When unspecified, M and V default to 0 and 0.01 respectively.

Example:

1. Read in the image and display it.
   c=imread(‘pic_0.bmp');

   imshow(c);

2. Add noise to it and display the result

```
GaussianNoise = imnoise(c,'gaussian',0,0.005);

imshow(c), figure, imshow(GaussianNoise);
```

3.1.3. Salt & Pepper Noise

Syntax:

J = imnoise(I,'salt & pepper',D) adds "salt and pepper" noise to the image I, where D is the noise density. This affects approximately D*PROD(SIZE(I)) pixels. The default for D is 0.05.

Example:

3. Read in the image and display it.
```
c=imread('samp_2.bmp');

imshow(c);
```

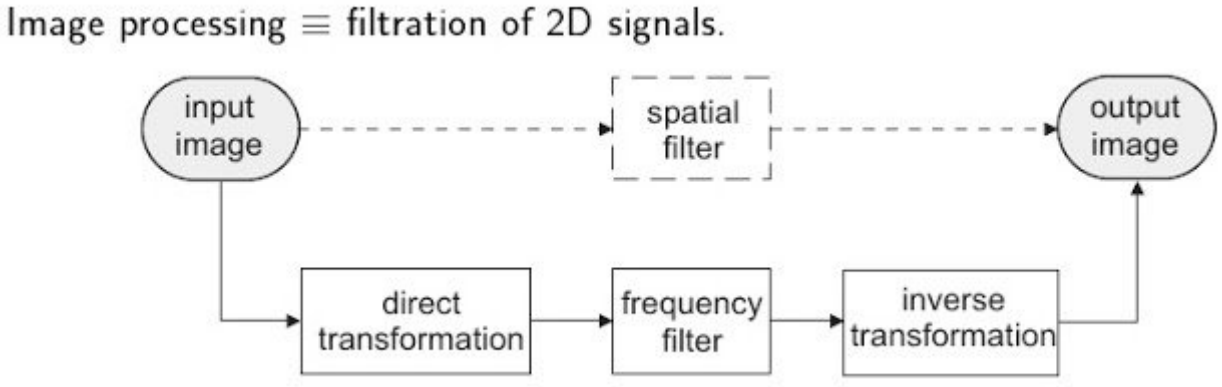
4. Add noise to it and display the result
```
J = imnoise(c,'salt & pepper', 0.02);
imshow(c), figure, imshow(J);
```

3.2. Filtering Techniques

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. It is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel.

Two Approaches for image filtering

1. Spatial domain: direct modification of the pixels in an image.
2. Frequency domain: modification of the Fourier transform of an image.



Spatial domain approaches

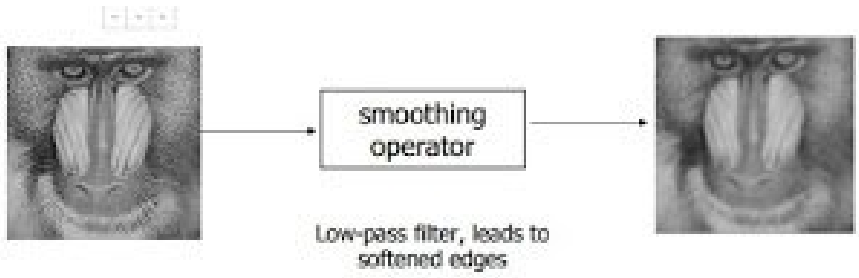
In the spatial domain method, the pixel composing of image details are considered and the various procedures are directly applied on these pixels.

Definition

- It is manipulating or changing an image representing an object in space to enhance the image for a given application.
- Its techniques are based on direct manipulation of pixels in an image.
- It is used for filtering basics, smoothing filters, sharpening filters, unsharp masking and laplacian.

Techniques

Smoothing



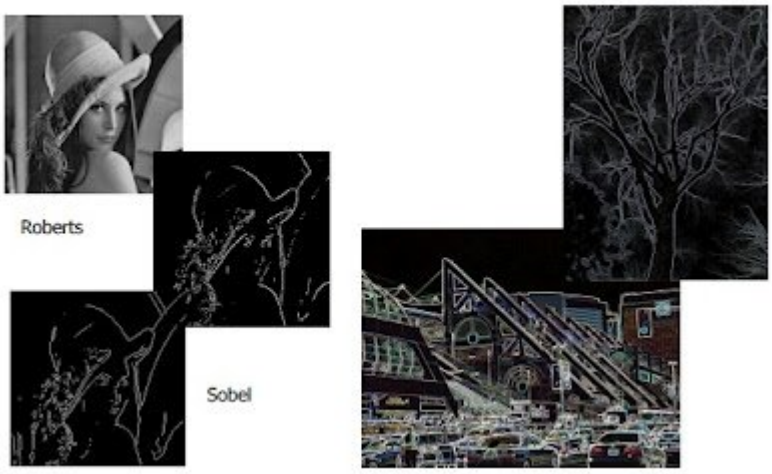
Unsharp Masking

- Is image manipulation technique for increasing the apparent sharpness of photographic images
- Uses a blurred or unsharp, positive to create a mask of the original image.

Unsharp Mask Example

Laplician

- Highlight regions of rapid intensity change and is therefore often used for edge detection.
- Often applied to an image that has first been smoothed with something approximating in order to reduce its sensitivity to noise.



Edge Detection Example

2. Frequency Domain approaches

Definition

- Techniques are based on modifying the spectral transform of an image
- Transform the image to its frequency representation
- Perform image processing
- Compute inverse transform back to the spatial domain
  - High frequencies correspond to pixel values that change rapidly across the image (e.g. text, texture, leaves, etc.)
  - Strong low frequency components correspond to large scale features in the image (e.g. a single, homogenous object that dominates the image)

Technique

Fourier Transform

- Function that are not periodic but with finite area under the curve can be expressed as the integral of sines and/ or sines multiplied by a weight function.

3.2.1. Linear Filtering

It is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood. For example, an algorithm that computes a weighted average of the neighborhood pixels is one type of linear filtering operation. It replaces the original pixel by the weighted sum of its neighboring pixel.

3.2.1.1. Average filtering

It is also called mean filtering, which is a simple, intuitive and easy to implement method of smoothing images, i.e. reducing the amount of intensity variation between one pixel and the next. It is windowed filter of linear class, that smoothes signal (image). The basic idea behind mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. An averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (*e.g.* 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar but not identical effect as a single pass with a large kernel.)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 1 3×3 averaging kernel often used in mean filtering

An averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced. Filtering of images, either by correlation or convolution can be performed using the toolbox function **imfilter**.

The function fspecial produces several kinds of predefined filters, in the form of computational molecules. After creating a filter with fspecial, you can apply it directly to your image data using filter2, or you can rotate it 180 degrees and use conv2 or convn. One simple filter fspecial can produce is an averaging filter. **h = fspecial('average', hsize)** returns an averaging filter h of size hsize. The argument hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3].

Example-1:

This example filters an image with a 5-by-5 filter containing equal weights. Such a filter is often called an **averaging filter**.

```
I = imread('pic_0.bmp');
B = imresize(I,[210 210],'nearest');
h = ones(5,5) / 25;
I2 = imfilter(B,h);
imshow(B), title('Original Image'), figure, imshow(I2), title('Filtered Image')
```

Example-2

1. Read in the image and display it.

```
c=imread('pic_0.bmp');

imshow(c);
```

2. Add noise to it and display the result

```
J = imnoise(c,'salt & pepper', 0.02);
imshow(c), figure, imshow(J);
```

3. Filter the noisy image with an averaging filter and display the results.

```
K = filter2(fspecial('average',3),J)/255;
imshow(c), figure, imshow(J), figure, imshow(K);
```

filter2: is the same as imfilter but always converts to double.

Example-3:

```
I = imread('pic_0.bmp');
%% Show current image.
imshow(I);
figure;
G=imnoise(I,'gaussian',0.0005,0.0019);
imshow(G);
figure;
h = fspecial('average', 3);
F=imfilter(G,h);
imshow(F);
imshow(I), figure, imshow(G), figure, imshow(F);
```

Example: 4

```
c=imread('pic_0.bmp');

imshow(c);
```

```
z=fspecial('average',[7,7]);

y=fspecial('average',[5,5]);

k=fspecial('average',[3,3]);

g=imfilter(c,z);

f=imfilter(c,y);

e=imfilter(c,k);

imshow(c), figure, imshow(e), figure, imshow(f), figure, imshow(g);
```

Example-5

```
c=imread('pic_0.bmp');

h = fspecial('average',5);

I2 = uint8(round(filter2(h,c)));

imshow(c), figure, imshow(I2);
```

3.2.1.2. Gaussian filtering

Gaussian filter is windowed filter of linear class; by its nature is weighted mean. It is a smoothing filter with a non-uniform kernel, whose coefficients are derived from a 2D Gaussian function.

Gaussian filter characteristics

- More weight is given to central pixels than to those in the periphery of the neighborhood
- The kernel coefficients diminish in size with increasing distance from the kernels centre
- Large values of produce a wider peak – increased blurring
- As increases the dimensions of the kernel also increase
- The kernel is rotationally symmetric, so there is no directional bias in the amount of smoothing
- The Gaussian kernel is separable.

Example-1

1. Read in the image and display it.

   ```
   c=imread('pic_0.bmp');
   ```

2. h = fspecial('gaussian', hsize, sigma) returns a rotationally symmetric Gaussian lowpass filter of size hsize with standard deviation sigma (positive). hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3]; the default value for sigma is 0.5.

   ```
   h = fspecial('gaussian', [3,3], 0.5);
   K = imfilter(c,h);
   ```

3. Display the result
   imshow(c), figure, imshow(K);

3.2.2. Non-linear Filtering

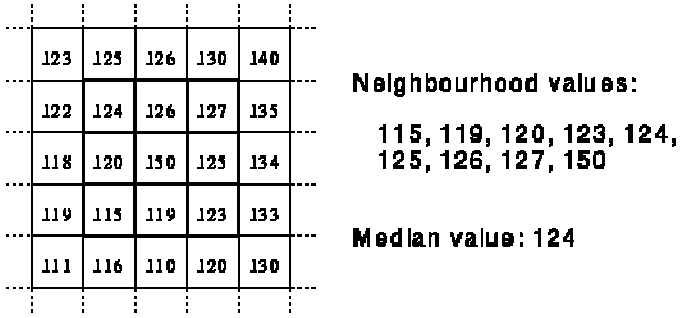
Nonlinear filters have quite different behavior compared to linear filters. For nonlinear filters, the filter output or response of the filter does not obey the principles outlined earlier, particularly scaling and shift invariance. Moreover, a nonlinear filter can produce results that vary in a non-intuitive manner. In signal processing, a filter is said to be a **nonlinear** (or **non-linear**) if its output is not a linear function of its input. That is, if the filter outputs signals R and S for two input signals r and s separately, but does not always output $\alpha R + \beta S$ when the input is a linear combination $\alpha r + \beta s$.

Non-linear filters have many applications, especially in the removal of certain types of noise that are not additive. For example, the median filter is widely used to remove spike noise that affects only a small percentage of the samples, possibly by very large amounts. Indeed all radio receivers use non-linear filters to convert kilo to gigahertz signals to the audio frequency range; and all digital signal processing depends on non-linear filters (analog-to-digital converters) to transform analog signals to binary numbers.

However, nonlinear filters are considerably harder to use and design than linear ones, because the most powerful mathematical tools of signal analysis (such as the impulse response and the frequency response) cannot be used on them. Thus, for example, linear filters are often used to remove noise and distortion that was created by nonlinear processes, simply because the proper non-linear filter would be too hard to design and construct. In many applications linear filtering techniques do not provide satisfactory results and nonlinear filters must be used.

3.2.2.1. Median Filter

Median filtering is similar to using an averaging filter, in that each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the *median* of the neighborhood pixels, rather than the mean. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used.) Figure 1 illustrates an example calculation.



As can be seen, the central pixel value of 150 is rather unrepresentative of the surrounding pixels and is replaced with the median value: 124. A 3×3 square neighborhood is used here --- larger neighborhoods will produce more severe smoothing.

The **medfilt2** function implements median filtering. It does a better job of removing noise, with less blurring of edges than average filtering. Median filtering can preserve discontinuities in a step function and can smooth a few pixels whose values differ significantly from their surroundings without affecting the other pixels. It is also useful in preserving edges in an image while reducing random noise.

The median filter, when applied to grayscale images, is a neighborhood **brightness-ranking** algorithm that works by first placing the brightness values of the pixels from each neighborhood in ascending order. The median or middle value of this ordered sequence is then selected as the representative brightness value for that neighborhood. Subsequently, each pixel of the filtered image is defined as the median brightness value of its corresponding neighborhood in the original image.

The **median filter** for color images operates differently from the **grayscale median filter**. Since each pixel in an RGB color image is composed of three components (red, green, and blue), it is not useful to rank the pixels in the neighborhood according to brightness. Instead, the color median filter works by comparing each pixel's color to that of every other pixel in the neighborhood. The pixel whose red, green, and blue components have the smallest sum of squared differences from the color coordinates of its neighbors is then chosen to replace the central pixel of the neighborhood.

Example: 1

```
%% read the image from the file system to Matrix I
I = imread('samp_1.bmp');

 %% create a new figure to show the image .
 figure(1);
 %% show the loaded image.
 imshow(I);

 %Add noise to it and display the result
J = imnoise(I,'salt & pepper', 0.02);
imshow(c), figure, imshow(J);

%% apply median filter using the internal matlab function medfilt2 .
 Y=medfilt2(J,[5 5]);

 figure(2);
 %% show image after applying the filter
```

```
imshow(Y);

%% write the new image to the file system .
imwrite(Y,'newimage.gif','GIF');
```

Example-2: (medfilt2 with imadjust)

```
c=imread('samp_2.bmp');

L = medfilt2(c,[5  5]);figure, imshow(L); title('median  filtering');

Z = imadjust(L,[0.3 0.7],[]);

figure, imshow(Z)
```

3.2.2.2. Adaptive filtering

The wiener2 function applies a Wiener filter (a type of linear filter) to an image *adaptively,* tailoring itself to the local image variance. Where the variance is large, wiener2 performs little smoothing. Where the variance is small, wiener2 performs more smoothing. This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image. In addition, there are no design tasks; the wiener2 function handles all preliminary computations and implements the filter for an input image. wiener2, however, does require more computation time than linear filtering. Wiener2 works best when the noise is constant-power ("white") additive noise, such as Gaussian noise. The example below applieswiener2 to an image of Saturn that has had Gaussian noise added.

Example-1:

1. Read in an image.

```
c=imread('samp_2.bmp');
```

2. Then add salt & pepper noise to the image.

```
J = imnoise(c,'salt & pepper',0.02);
```

3. Then add Gaussian noise to the image.

```
z = imnoise(c,'gaussian',0,0.025);
```

4. Remove the noise, using the wiener2 function.

```
K = wiener2(J,[5 5]);

p = wiener2(z,[5 5]);
```

Display the original image, the

```
imshow(c), figure, imshow(K), figure, imshow(p);
```

Example -2

```
c=imread('Ministri.jpg');
```

```
k=imnoise(c,'Speckle',1);

x=rgb2gray(k);

g = wiener2(x,[5 5]);

imshow(c),title('mini'), figure, imshow(k), title('mini with noise'), figure, imshow(ix), title('gray'), figure, imshow(g), title('adaptive filter');

imwrite(g,'C:\MATLAB6p1\work\a.jpg','jpg');
```

Example-3

```
c=imread('Ministri.jpg');

B = IMRESIZE(a,[210 410],'bicubic'); % OR >> B = IMRESIZE(a,[210 410],'bilinear');

e=rgb2gray(B);

Q = wiener2(e,[7 7]);

figure,imshow(B), figure,imshow(Q);
```

3.2.3. Others

3.2.3.1. BWAREAOPEN (Binary area open)

Syntax

BW2 = BWAREAOPEN(BW,P) removes from a binary image all connected components (objects) that have fewer than P pixels, producing another binary image BW2.

Example: Remove all objects in the image samp_2.bmp containing fewer than 40 pixels.

```
bw=imread('samp_2.bmp');

bw2=bwareaopen(bw, 40);

imshow(bw2);
```

NB: Binary images not supported by imadjust.

3.2.3.2. conv2 or filter2

To perform two-dimensional convolution, you use conv2 or filter2. To perform higher-dimensional convolution, you use the convn function. convn takes as arguments a data array and a convolution kernel, both of which can be of any dimension, and returns an array whose dimension is the higher of the two input arrays' dimensions.

```
c=imread('samp_2.bmp');

B = convn(c,3);
```

imshow(c), figure, imshow(B);

3.2.3.3. Minimum Filtering

```
c=imread('three.jpeg');

I = rgb2gray(c);

B=ordfilt2(I,1,ones(3, 3), symmetric);

imshow(I), figure, imshow(B);
```

3.2.3.4. Maximum Filtering

```
c=imread('three.jpeg');

I = rgb2gray(c);

zmax = ordfilt2(I, 3*3, ones(3,3), 'symmetric');

imshow(I), figure, imshow(B);
```

3.2.4. Edge detection

There are many ways to perform edge detection. However, the most may be grouped into two categories: Gradient (Roberts, Prewitt, Sobel) and Laplacian (Marr-Hildreth). The **gradient method** detects the edges by looking for the maximum and minimum in the first derivative of the image. The **Laplacian method** searches for zero crossings in the second derivative of the image to find edges. Edges in an image are those points which show sudden change of intensity.

Canny edge detector is a complex but accurate detector as compared to **Marr-Hilderth detector**. And it is based on these 3 following objectives: **Low Error Rate:** It says that all edges should be found and these detected edges should be as much close to true edges as possible. **Edge points should be well localized:** It states that the edges located should be as close to true edges as possible. In other words, the distance between a point detected or marked as an edge by canny detector and the centre of the true edge should be minimum. **Single Edge Point Response:** The detector must output one edge point for each true edge point. This figure shows the edges of an image detected using the gradient and Laplacian methods.

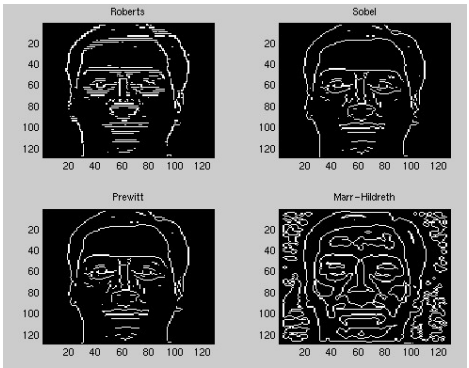


Fig 1. Various Edge Detection Filters

Notice that the facial features (eyes, nose, and mouth) have very sharp edges. These also happen to be the best reference points for morphing between two images. Notice also that the Marr-Hildreth not only has a lot more noise than the other methods, the low-pass filtering it uses distorts the actual position of the facial features.

Example: 1

```
a=imread('pic_0.bmp');

 i=edge(a,'sobel');

imshow(i), title('sobel edge detection');
```

Example: 2

```
a=imread('pic_0.bmp');

 i=edge(a,'LoG');

imshow(i), title('LoG edge detection');
```

Example: 3

```
a=imread('pic_0.bmp');

i=edge(a,'canny');

imshow(i), title('canny edge detection');
```

Example: 4

```
a=imread('pic_0.bmp');

i=edge(a,'prewitt');

imshow(i), title('perwitt edge detection');
```

Example: 5

```
a=imread('pic_0.bmp');

i=edge(a,'roberts');

imshow(i), title('Roberts edge detection');
```

4. Performance Evaluation

Signal to noise ratio

Different image filtering techniques are explored in order to remove noise in an image and it is required to identify which filtering technique is best. In order to do that, **Signal to noise ratio** is used. The signal to noise ratio(SNR) is a useful and universal way of comparing the relative amounts of signal and noise for any electronic system; high ratios will have very little visible noise whereas the opposite is true for low ratios. In other words, High SNR is used to clearly separate the image information from background noise. A low SNR would produce an image where the "signal" and noise are more comparable and thus harder to discrin from one another. Signal-to-noise ratio (SNR), often expressed in *decibel.* $SNR = \sigma s / \sigma n$

The MSE represents the cumulative squared error between the compressed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE is the lower the error. The PSNR block computes the peak signal-to-noise ratio, in decibel (dB), between two images. This ratio is often used as a quality measurement and the higher the PSNR, the better the quality of the reconstructed image.

Example

```
c = imread(pic_0.bmp);
grayImage = rgb2gray(c);
Median = MedianFilter(grayImage, 3);
NiblackThresholdedImage = NiblackThresholding(Median);
OtsuThresholdedImage = OtsuThresholding(Median);
SauvolaThresholdedImage = SauvolaThresholding(Median);
[MSE1, PSNR1] = calc_MSE_PSNR(grayImage, NiblackThresholdedImage)
[MSE2, PSNR2] = calc_MSE_PSNR(grayImage, OtsuThresholdedImage)
[MSE3, PSNR3] = calc_MSE_PSNR(grayImage, SauvolaThresholdedImage)
disp(c, 'Original Noisy Image');
disp(grayImage, 'Gray Noisy Image');
disp(Median, 'Filtered Image');
disp(NiblackThresholdedImage, 'NiblackThresholding');
disp(OtsuThresholdedImage, 'OtsuThresholding');
disp(SauvolaThresholdedImage, 'SauvolaThresholding');
disp(PSNR1, 'PSNR');
disp(PSNR2, 'PSNR');
disp(PSNR3, 'PSNR');
```

5. Discussion

Based on the experimental result, Median performs better than other linear and non-linear filtering techniques. For example, the median filter removes noise, while the average filter just spreads it around evenly. The performance of median filter is particularly better for removing impulse noise than average filter. By calculating the median value of a neighborhood rather than the mean filter, the median filter has some advantages over the mean filter:

- The median is a more robust average than the mean and so a single very unrepresentative pixel in a neighborhood will not affect the median value significantly.
- Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason the median filter is much better at preserving sharp edges than the mean filter.
- The median is a stronger "central indicator" than the average. In particular, the median is hardly affected by a small number of discrepant values among the pixels in the neighborhood. Consequently, median filtering is very **effective** at removing various kinds of noise.

Although median filter is a useful non-linear image smoothing and enhancement technique, it also has some disadvantages because, utilization of larger neighborhood masks or multiple applications of the median filter (with smaller neighborhood masks) improve noise suppression at the expense of a loss in image details. Anything relatively small in size compared to the size of the neighborhood will have minimal effect on the value of the median, and will be filtered out. In other words, the median filter can't distinguish fine detail from noise.

In addition, if the impulsive values are part of the signal, the result of using the median filter will distort the signal. The median filter is less effective in removing **Gaussian** or random-intensity noise, because the noisy pixels in this case are less likely to differ in brightness from the pixels in the neighborhoods they occupy. Also, the median filter can remove impulse noise from a neighborhood only if the noisy pixels occupy less than one half of the neighborhood area. The median filter may change the contours of objects in the image. With repeated applications of the filter (**3 x 3** and **5 x 5** median filtering), a contouring effect can occur, where pixel brightness values are leveled across regions (a **region** in this sense is a group of pixels having similar brightness values).

Besides, the median is much less sensitive than the mean to extreme values (called *outliers*). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. It generate an n x n neighborhood (n: 3-5 are common) around each pixel in the original image. Median filtering is a nonlinear process useful in reducing impulsive or salt-and-pepper noise which can occur due to a random bit error in a communication channel.

imadjust also perform better than other filtering techniques without a loss in image details but it can’t remove the noise totally and whatever the back ground color is, it makes white. This means it is not advisable when we want to give emphasis for the background color. So when Median and

imadjust are mixed, their performance is better. In addition, after the image is converted in to binary, Adaptive, Median and Adaptive-Median are better as compared to before the image is converted in to binary. In addition, when linear or non linear filtering is mixed with imadjust, they perform much better than the hybrid of linear and non linear filtering with each other.

Besides, if the level of the noise added on the image is less, it is difficult to differentiate the best filtering techniques (See experimental result) and, canny and the hybrid of Adaptive and Canny are the best from edge detections.

Assumption Taken

- In Median, minimum and Maximum filtering, the image must be a 2-D real uint8, uint16, or double array.
- In Average and BINARYAREAOPEN filtering, the image must be full double.
- In adaptive filtering, WIENER2 does not support 3D true color images as an input.
- To convert an image in to binary, the intensity image I has to be a two-dimensional array.
- Colored image can work before conversion to gray and binary and then works when it converts in to binary.
- Non colored images can't work before and after conversion to binary but not gray.
- Speckle noise reduces the image contrast and has a negative effect on texture based analysis. Moreover, as speckle noise changes the spatial statistics of the images, it makes the classification process a difficult task to do.
- The median filter does a better job of removing 'salt-and-pepper' noise, with less blurring of edges.
- Binary Image doesn't work in Average, Gaussian, Average and Gaussian, Average and Median, Average and Adaptive, Gaussian and Median, Gaussian and Adaptive Filtering.

6. Strategy

The strategies used in this paper for image filtering are:

Strategy 1:

1. Read Original image
2. Observe/Measure Signal to Noise ratio
3. Filter using any possible filtering techniques
4. Observe/Measure Signal to Noise ratio

5. Compare signal to noise ratio before and after applying filtering technique
6. Identify to what extent each filtering techniques remove the noise
7. Recommend the best filtering technique accordingly

Strategy 2:

1. Read Original image
2. Convert the image in to gray
3. Observe/Measure Signal to Noise ratio
4. Filter using any possible filtering techniques
5. Observe/Measure Signal to Noise ratio
6. Compare signal to noise ratio before and after applying filtering technique
7. Identify to what extent each filtering techniques remove the noise
8. Recommend the best filtering technique accordingly

Strategy 3:

1. Read Original image
2. Convert the image in to binary
3. Observe/Measure Signal to Noise ratio
4. Filter using any possible filtering techniques
5. Observe/Measure Signal to Noise ratio
6. Compare signal to noise ratio before and after applying filtering technique
7. Identify to what extent each filtering techniques remove the noise
8. Recommend the best filtering technique accordingly

Strategy 4:

1. Read Original image
2. Add Salt & Pepper Noise
3. Observe/Measure Signal to Noise ratio
4. Filter using any possible filtering techniques

5. Observe/Measure Signal to Noise ratio
6. Compare signal to noise ratio before and after applying filtering technique
7. Identify to what extent each filtering techniques remove the noise
8. Recommend the best filtering technique accordingly

Strategy 5:

1. Read Original image
2. Add Speckle Noise
3. Observe/Measure Signal to Noise ratio
4. Filter using any possible filtering techniques
5. Observe/Measure Signal to Noise ratio
6. Compare signal to noise ratio before and after applying filtering technique
7. Identify to what extent each filtering techniques remove the noise
8. Recommend the best filtering technique accordingly

Strategy 6:

1. Read Original image
2. Add Gaussian Noise
3. Observe/Measure Signal to Noise ratio
4. Filter using any possible filtering techniques
5. Observe/Measure Signal to Noise ratio
6. Compare signal to noise ratio before and after applying filtering technique
7. Identify to what extent each filtering techniques remove the noise
8. Recommend the best filtering technique accordingly

7. Sample Code

```
 k=input('Enter menu number: ');
switch k
case 1
    disp('[3,3]Average filtering');
    if noise==1
        img=img2;
        noise=0;
    end
    Average=filter2(fspecial('average',[3,3]),img)/255;
    figure, imshow(img), title 'Original Image';
    figure, imshow(Average), title 'Average Filtering';
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
case 4
    disp('Gaussian filtering');
    if noise==1
        img=img2;
        noise=0;
    end
    h=fspecial('gaussian', [3 3]);
    Gaussian=imfilter(img,h);
    figure, imshow(img), title 'Original Image';
    figure, imshow(Gaussian), title 'Gaussian Filtering ';
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
```

```
case 7
    disp('[3,3]Median filtering');
    if noise==1
        img=img2;
        noise=0;
    end
    median=medfilt2(img,[3 3]);
    figure, imshow(img), title 'Original Image';
    figure('name','Median Filtering');imshow(median);
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
case 10
    disp('[3,3]Adaptive filtering');
    if noise==1
        img=img2;
        noise=0;
    end
    Adaptive= wiener2(img,[3 3]);
    figure, imshow(img), title 'Original Image';
    figure, imshow(Adaptive), title 'Adaptive Filtering';
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
 case 13
    disp('BWAREAOPEN filtering');
    if noise==1
        img=img2;
```

```
    noise=0;
  end
  BW=bwareaopen(img, 40);
  filter2(fspecial('average',3),img)/255;
  figure, imshow(img), title 'Original Image';
  imshow(img), figure, imshow(BW), title 'BWAREAOPEN filtering';
  disp('Do you want to continue [Y]/[N]?');
  s=input('-->','s');
case 14
  disp('Minimum filtering');
  if noise==1
    img=img2;
    noise=0;
  end
  ordfilter=ordfilt2(img,1,ones(3, 3), 'symmetric');
  figure, imshow(img), title 'Original Image';
  figure('name','Minimum filter');imshow(ordfilter);
  disp('Do you want to continue [Y]/[N]?');
  s=input('-->','s');
case 15
  disp('Maximum filtering');
  if noise==1
    img=img2;
    noise=0;
  end
  ordfilter=ordfilt2(img,3*3,ones(3, 3), 'symmetric');
```

```
    figure, imshow(img), title 'Original Image';
    figure('name','Maximum filter');imshow(ordfilter);
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
case 16
    disp('imadjust');
    if noise==1
       img=img2;
       noise=0;
    end
    Adjust= imadjust(img,[0.3 0.7],[]);
    figure, imshow(img), title 'Original Image';
    figure('name','imageAdjust');imshow(Adjust);
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
case 17
     disp('sobel');
    if noise==1
       img=img2;
       noise=0;
    end
    sobel=edge(img,'sobel');
    figure, imshow(img), title 'Original Image';
    imshow(sobel), title('sobel edge detection');
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
```

```
case 22
    disp(' Average and Gaussian ');
    if noise==1
       img=img2;
       noise=0;
    end
    figure('name','Original image');imshow(img);
    Average=filter2(fspecial('average',[3,3]),img)/255;
    h=fspecial('gaussian');
    Gaussian=imfilter(Average,h);
    figure, imshow(img), title 'Original Image';
    figure('name',' Average and Gaussian filtering');imshow(Gaussian);
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
  case 28
    disp(' Median and Adjust');
    if noise==1
       img=img2;
       noise=0;
    end
    figure('name','Original image');imshow(img);
    median=medfilt2(img,[3 3]);
    Adjust= imadjust(median,[0.3 0.7],[]);
    figure, imshow(img), title 'Original Image';
    figure('name','Adjust and Median filtering');imshow(Adjust);
    disp('Do you want to continue [Y]/[N]?');
```

```
    s=input('-->','s');
case 32
     disp('sobel and Adaptive');
    if noise==1
        img=img2;
        noise=0;
    end
    sobel=edge(img,'sobel');
    Adaptive=wiener2(sobel);
    figure, imshow(img), title 'Original Image';
    imshow(sobel), title('sobel edge detection');
    imshow(Adaptive), title('sobel edge detection and Adaptive filter');
    disp('Do you want to continue [Y]/[N]?');
    s=input('-->','s');
```

8. Concluding remark

Images are discrete digital signals subjected to additive, random and impulse noise. We can attenuate this noise through a variety of filters (Average, Gaussian, Median, Adaptive, imadjust Minimum Maximum and others). Each filtering techniques has different performance one another. Their difference can be visible but there are also situations which require standard performance Evaluation method (signal to noise ratio) to identify which is better. Generally, what is done in this project is applying many possible filtering techniques (before and after converting the image in to gray and binary, before and after adding common noise types and mix different filtering techniques), observe/measure signal to noise ratio and come up with a best filtering techniques accordingly.

9. Recommendation

Image adjusts and Median has better performance than other filtering techniques discussed. But they are not always good for all types of noises. So hybrid (Using two or more filtering techniques together) is better than them. There is also difference between different hybrids. When linear or non linear filtering is mixed with imadjust, they perform much better than the hybrid of linear and non linear filtering with each other. In addition, after the image is converted in to binary, Adaptive, Median and Adaptive-Median are better as compared to before the image is converted in to binary.

10. Reference

1. Documentation center, "Image Processing Toolbox" Available: http://www.mathworks.com/help/images/index.html [Accessed Date: June 29, 2013].

2. Paulos Gnogno, "የኢትዮጵያ እና የኢጣሊያ ጦርነት", September 1988, Addiss Abeba, print media, Addis abeba.

3. Roger Claypoole, et.al. "Other Methods of Edge Detection" Internet: http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/moredge.html, [Accessed Date: August 5, 2013

4. R. Fisher, et.al. "Median filtering", Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm, Accessed Date: [August 7, 2013], ©2003.

5. Biniam asnake, "Retrieval from real-life Amharic document images", June 2012, Addis ababa university .
6. የሰሜን ኮከብ ስነጽሁፍ ክለብ, እምርታ, [16, 17, 25, 28, 29] 1987.

7. የፋሲለደስ አጠ/2ኛ ደረጃ ት/ቤት የህዝብ ግንኙነትና የትምህርት ዜና አገልግሎት ክፍል, ፍኖተ ጥበብ , June 1990, Gondar, [3, 4, 5, 6, 11, 15, 17].