

## LAB 4 - Modelling Noise & 2D Filters - Chapter 6- MATLAB for Image Processing

Lab Report Due: - March 5, 2008

### 1. Background

Real world signals usually contain departures from the ideal signal that would be produced by our model of the signal production process. Such departures are referred to as noise. Noise may be caused by a wide range of sources, e.g. variations in the detector sensitivity, environmental variations, the discrete nature of radiation, transmission or quantization errors, etc.

Noise can generally be grouped into two classes:

(i) Independent Noise.

Image independent noise can often be described by an *additive* noise model, where the recorded image  $f(i, j)$  is the sum of the true image  $s(i, j)$  and the noise  $n(i, j)$ :

$$f(i, j) = s(i, j) + n(i, j)$$

The stationary noise  $n(i, j)$  is often zero-mean and described by its variance  $\sigma_n^2$ . The impact of the noise on the image is often described by the signal-to-noise ratio (SNR)

$$SNR = \frac{\sigma_s}{\sigma_n} = \sqrt{\frac{\sigma_f^2}{\sigma_n^2} - 1}$$

where  $\sigma_s^2$  and  $\sigma_f^2$  are the variances of the actual image and the recorded image respectively. An image often contains low frequency information whereas noise ( $\mathfrak{R}_n(\tau) \leftrightarrow S_n(\omega)$ ) is high frequency. Filtering has to be done in a *statistical sense*, such as by using the so-called Wiener filter.

(ii) Data-Dependent Noise

Data-dependent noise can arise when monochromatic radiation is scattered from a surface, causing wave interference which results in image *speckle*. It is possible to model this noise with a multiplicative or a non-linear model.

We also have so-called Salt and Pepper Noise. This is often manifested as intensity spikes. Here, the noise may be caused by errors in the data transmission. The corrupted pixels are either set to the maximum value (which looks like snow in the image) or have single bits flipped over. In some cases, single pixels are set alternatively to zero or to the maximum value, giving the image a ‘salt and pepper’ like appearance.

### 2. Modelling Noise

There are many types of noise: Thermal noise, Poisson (shot) noise, white noise and  $1/f$  noise. White noise, by definition, has a constant power spectral density  $S_n(\omega)$ .  $1/f$  noise is often found in natural phenomena such as nuclear radiation, electron flow through a conductor, or even in the environment. In electrical engineering, it is called also Flicker noise.

Do all the experiments on pages 6-6 to 6-12, MATLAB for Image Processing. They are described in items **1** → **5** below.

1. Use MATLAB to generate (i) white additive 'gaussian noise', (ii) 'poisson noise' , (iii) 'salt and pepper noise' and (iv) 'speckle' noise. Just generate this noise and get a sense of it. The function to use is the following:

```
>> J = imnoise(I, type, parameters)
where the variables are as follows:
I - image
type- gaussian, localvar, poisson, salt & pepper, speckle
parameters - these are used depending on the type of noise used.
```

Examples:

`J=imnoise(I, 'gaussian', m ,v)` adds Gaussian white noise of mean  $m$  and variance  $v$  to image  $I$ . Default values are zero mean and variance 0.01.

`J=imnoise(I, 'poisson')` generates Poisson noise. Poisson noise is *signal-dependent* which means that the variance of the noise changes, depending upon pixel intensities. This type of noise can be large, so standard techniques for 'small noise' removal can be ineffective. Sometimes the noise is converted to Gaussian and then denoised using standard Gaussian denoising algorithms.

`J = imnoise(I, 'salt & pepper', d)` adds salt and pepper noise to image  $I$ .  $d$  is noise density. This affects approximately  $d \times \text{prod}(\text{size}(I))$  pixels. Default value of  $d$  is 0.05 noise density.

`J = imnoise(I, 'speckle', v)` adds multiplicative noise to image  $I$ , according to the formula  $J = I + n \times I$ , where  $n$  is uniformly distributed random noise, mean 0 and variance  $v$ . Default value for  $v$  is 0.04.

We will focus a bit more in depth here, on additive white gaussian noise (awgn). Let us examine the nature of awgn. We can start with an image of all zeros so that we can get to the noise part. Use *imnoise* to add wgn with zero mean and variance of, say, 1. Examining this, can you figure out how you can show that the wgn process indeed has zero mean and variance 1?

As a related issue, let us generate awgn noise directly, using the function *wgn*. Use `w=wgn(100,1,1)` to generate a 100 x 1 matrix  $w$  of wgn with a power of 1 db ( as in watts when applied to a 1 ohm resistor).

- i. Use *hist* to show that  $w$  is gaussian.
- ii. Find the mean of  $w$ .
- iii. Look at *xcorr*( $w$ ) to show that it is uncorrelated.
- iv. Look at *xcov*( $w$ ) to show that it is uncorrelated.
- v. Obtain a zero mean realization (generated from using *xcorr*),  $W = w - \text{mean}(w)$ .
- vi. Do  $W' * W$  to find total power. Divide by 100 to find average power.
- vii. Do  $W' * W$  to find total power. Divide by 99 to find average power.
- viii. Now find *var*( $w$ ). Does it agree with 6 or 7? Why?
- ix. Is this 1 db?

Note that you could have the done the same analysis for *awgn*(100, 100, 1).

2. We use a simple averaging filter to filter speckle noise. Try the following:

```
I =imread('cameraman.tif');
I_noise = imnoise(I, 'specvkle', 0.01);
h = fspecial('average', [3 3])
I2 = imfilter(I_noise, h);
imshow(I_noise), title('Original image')
figure
imshow(I2), title('Filtered image')
```

Note the loss of resolution. Determine the frequency spectrum of the 2-D filter  $h$ .

3. When the noise is so-called 'salt & pepper' noise, the image pixel intensity values vary vastly from the original values. Linear filters such as averaging filters do not have much effect. Median filters are typically used. The syntax for this is:

```
J = medfilt2(I, [m n]);
```

where

I - input image

J - output image

[m n] - the block size used to calculate the median value.

Example:

```
I = imread('eight.tif');
I_noise = imnoise(I, 'salt & pepper');
h=fspecial('average', [ 3 3]);
I_avg = imfilter(I_noise, h);
I_md= medfilt2(I_noise, [3 3]);

subplot(2,1,1)
imshow(I_noise), title('Quarters with Salt & Pepper Noise');
subplot(2,2,3)
imshow(I_avg), title('Averaging Filter')
subplot(2,2,4)
imshow(I_md), title('Median Filter')
```

4. We use the Wiener filter here. (We will cover it in class). The Wiener filter does low pass filtering on an intensity image that has been degraded by constant power additive noise. It estimates statistics from local neighborhoods of each pixel.

Syntax is as follows:

```
[J, noise] = wiener2(I, [m n], noise)
where
I - input image
[m n] - size of local neighborhood
J - output image
noise - Noise variance
```

Try the following:

```
I = imread('saturn.tif');  
J = imnoise(I,'gaussian',0,0.005);  
K = wiener2(J,[5 5]);  
imshow(J), figure, imshow(K)
```

5. Under Filter Demonstration , use *nrfiltdemo* to get familiar with this GUI for filtering.

## REPORT

1. Comment on 1.
2. Comment on 2.
3. Comment on 3.
4. Comment on 4.
5. Comment on 5.

References: (1) 'MATLAB for Image Processing ©2007, The MathWorks, Inc. Image Restoration Techniques Ch.6. Note that we only have *one* copy in the Lab. It is copyrighted by MathWorks. Although the notes here are probably sufficient for this experiment, you may need to refer to the Ch.6 notes for additional details.

(2.) <http://homepages.inf.ed.ac.uk/rbf/HIPR2/noise.htm>

Class notes:mirchand/ee295