## EE 5356: Project 4 Hint

## Types of Noise

1. Gaussian noise - Gaussian noise is statistical noise that has a probability density function of the normal distribution (also known as Gaussian distribution). In other words, the values that the noise can take on are Gaussian-distributed. It is most commonly used as additive white noise to yield additive white Gaussian noise (AWGN).
2. Poisson noise - Poisson noise has a probability density function of a Poisson distribution.
3. Salt & pepper noise - It represents itself as randomly occurring white and black pixels. An effective noise reduction method for this type of noise involves the usage of a median filter. 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.
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.

## Adding noise to images

MATLAB provides a function ‘imnoise’ to conveniently add the desired type of noise to an image. For e.g.

m = 0;

v = 0.01;

J1 = imnoise(I,'gaussian',m,v);

Adds Gaussian noise with mean=0 and variance=0.01 to the image. See MATLAB’s documentation for ‘imnoise’. Use MATLAB’s default parameters when none have been specified here.

## Filter descriptions

Corrupted image Filtered image

Sxy = set of coordinates in a rectangular subimage window (neighborhood) of size (m x n) centered at point (x,y).

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

n

Center

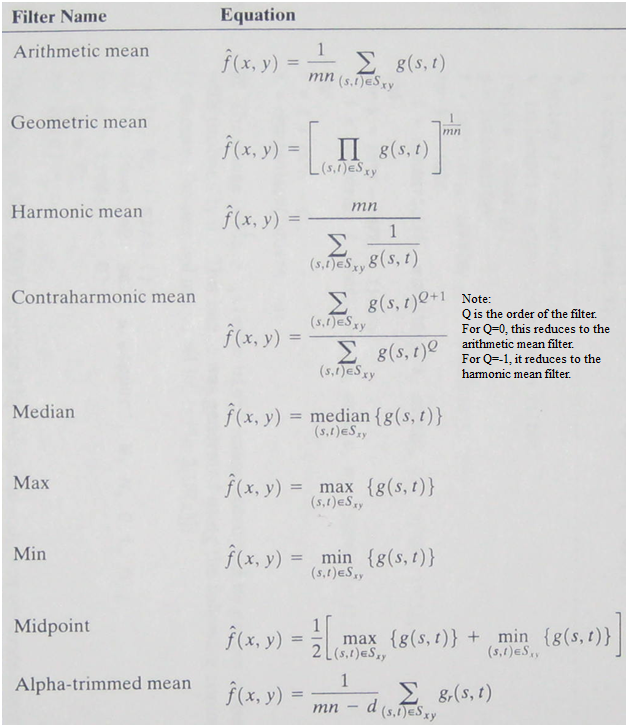
(x,y)

m

(3 x 3) window

1. Arithmetic mean filter: This filter computes the average value of the pixels intensity values in the sub-image (of size m x n).
2. Geometric mean filter: It is similar to an arithmetic mean filter, but it tends to lose less detail in the process.
3. Harmonic mean filter: This filter computes the geometric mean of the pixels intensity values.
4. Contraharmonic mean filter: This filter computes the geometric mean of the pixels intensity values. Note that the contraharmonic filter reduces to the mean filter for Q = 0, and to the harmonic mean filter for Q = -1.
5. Median filter: Replaces the value of the pixel by the median of the pixels in the sub-image.
6. Max filter: For d = 1, replaces the value of the pixel by the maximum of the pixel intensity values in the sub-image. For d>1, uses the mean of the top d values.
7. Min filter: For d = 1, replaces the value of the pixel by the minimum of the pixel intensity values in the sub-image. For d>1, uses the mean of the lowest d values.
8. Mid-point filter: Replaces the value of the pixel by the mid-point of the pixels in the sub-image.
9. Alpha trimmed mean filter: Replaces the value of the pixel by the mean of the remaining pixel intensity values after discarding the top d/2 and lowest d/2 intensity values.

Reference: Pages 322-325 of textbook.



Taken from: Gonzalez, Woods and Eddins, “Digital Image Processing with MATLB”, I edition, Prentice Hall, 2009, page 160.

## Filtering the image

Filtering the image is a neighborhood operation. The 3x3 neighborhood of a pixel (x,y) is shown:

(source: http://www.comp.dit.ie/bmacnamee/materials/dip/lectures/ImageProcessing5-SpatialFiltering1.ppt)

Linear filters can be implemented by multiplying the pixel neighborhood with the filter kernel. This is done conveniently using the ‘imfilter’ function. But in this project, with the exception of the averaging filter, all other filters are non-linear filtering operations and cannot be implemented using ‘imfilter’.

There are several other ways to implement non-linear filtering in MATLAB:

1. ‘nlfilter’ – general sliding neighborhood operations.
2. ‘colfilt’ – column wise neighborhood operations. This is a bit more complex to implement than ‘nlfilt’, but is optimized for speed.
3. ‘ordfilt2’ – performs order statistic filtering. It is only applicable to median, mid-point, min, and max filters.

Please see the MATLAB documentation for these functions. They have several examples to illustrate their usage.

Typical usage of one of these functions is:

result = nlfilter(im, [M N], @fun);

Where im in the input image, [M N] is the neighborhood size, typically [3 3], and @fun is a function handle. The output image is stored in result. This function will extract the neighborhood for every pixel in the image and call fun with the neighborhood as the parameter. ‘fun’ will process these pixels (for e.g. computes the geometric mean of the values) and produce a resulting pixel value. ‘fun’ can be a MATLAB built-in function or a function you wrote.

A simple (but inefficient) example of a geometric mean filter:

% in main

gmean\_result = nlfilter(im, [M N], @gmean);

% in gmean.m

function v = gmean(A)

[M N] = size(A);

prod = 1;

for i=1:M

for j=1:N

prod = prod \* A(i,j);

end

end

v = prod ^ (1/MN);

References:

1. Rafael C. Gonzalez and Richard E. Woods, “Digital Image Processing”, III edition, Prentice Hall, pages 322-325, 2008.
2. Gonzalez, Woods and Eddins, “Digital Image Processing with MATLB”, I edition, Prentice Hall, pages 160-164, 2009.
3. Practical image and video processing using MATLAB by Marques, Oge