HERIOT-WATT UNIVERSITY

DEPARTMENT OF COMPUTING AND ELECTRICAL ENGINEERING

B35SD2– Matlab tutorial 5
Filtering

Objectives:

We will demonstrate various aspects of filtering, including spatial and frequency domains and we will analyse the frequency response of various filters and their interpretation.

During this session, you will learn & practice:

1- Filter synthesis in space and frequency domain
2- Filter response analysis
3- Various examples of commonly used filters
4- Noise removal using various filters

Ressources required:

In order to carry out this session, you will need to download images and matlab files from the following location:

http://www.ece.eps.hw.ac.uk/~ceeyrp/WWW/Teaching/B39SD2/B39SD2.html

Please download the following images:

- Image of Lena
- Landsat image

We will also use the circuit and flowers images (circuit.tif, peppers.png).
You are of course encouraged to try these programs out on images of your choice.

Please download the following functions and script:

- SimpleFiltering.m
- gene_filters.m
- Analyse_Filter.m
- Deconvolution.m

SimpleFiltering:

The simplest form of filtering are high and low pass filters applied directly in the frequency domain. This is illustrated in the SimpleFiltering program as you might have time to see in tutorial 2.
It is interesting to note that a finite frequency response (as it is the case for ideal low pass and band-pass filtering) lead to an infinite space signal (image) and that these filters are in general not realizable are they are not causal.

Load an image of your choice (grey level)
Display the image in a figure;
Use the SimpleFiltering function with various settings (see help SimpleFiltering).

Comments?

This type of filtering is more of the type *feature extraction* as we define the response of the filter in function of the type of characteristics of the image we want to extract. More efficient filters (both in terms of response and computation time) have been proposed over the years to perform these type of operations.

Space domain filtering

Space domain filters are generally defined as small windows (or images) that are convolved with the original image (see your notes on filtering). In the space domain, the action of a filter can be seen looking at the structure of the filter. For instance a mean filter will just replace the value of the current pixel by the mean of its neighbour. This is typically a low pass filtering operation as we will show shortly. Conversely, a Prewitt or Sobel filter acts as a derivation and tends to locate edges. This is typically a high-pass / band-pass filtering operation.

Low pass filtering:

Using the gene_filters function, you will generate the following filters:

- fmean3 : [3x3] mean filter
- fmean5 : [5x5] mean filter
- fmean11 : [11x11] mean filter
- fgaussian5_1 :[5x5] Gaussian filter with standard deviation 1. See help fspecial.
- fgaussian7_05 :[7x7] Gaussian filter with standard deviation 0.5.
- fgaussian7_3 :[7x7] Gaussian filter with standard deviation 3.

These filters are all low pass filters. The interest of the Gaussian filter lies in the fact that the Fourier transform of a Gaussian is a Gaussian and therefore the frequency response of the filter is very smooth with no virtually no sidelobes.

Use the function AnalyseFilter(image,filter) to see the effect of each filter on the image and their associated frequency response. This will only work on grey level images but can be easily extended to color images.

High / Band-Pass pass filtering and edge detection:

Using the gene_filters function, you will also have generated the following filters:

- flog : [5x5] Laplacian of Gaussian filter
- fprewittx : [3x3] Prewitt filter in the x direction
- fsobelx : [3x3] Sobel filter in the x direction

These filters are all band-pass or high-pass filters used to perform edge detection
Use the function AnalyseFilter(image,filter) to see the effect of each filter on the image and their associated frequency response. This will only work on grey level images but can be easily extended to color images. You can see that these filters are not always ideal and have a fairly smooth frequency drop-off while you would prefer a quick one for better performance. The last filter (ftop) is designed using an optimal design technique (remez) mainly used in1D signal processing using FIR filtering theory. The cut-off frequencies can be accurately chosen and the response of the filter is more controllable. Try it on various images.

All these filters do not take into account noise in the images and are not adapted to the spectral content of the specific image we are dealing with. This is a major drawback in some applications and can be dealt with using adaptive filtering (wiener for instance). This is a vast area of filtering that will be covered in one of the assignments.

All the filters considered here are linear filters. Some more advanced filters, usually non-linear, have also been developed to perform feature extraction. It is useful to note filters such as canny for edge detection.

Non-linear filters and Noise removal

So far we have used filters merely as feature extraction tools (edges, low frequencies, …). We have also only considered *linear filters* where the resulting image is obtained by convolution of the input image with the filter, making the analysis simple.
But filters are also used to perform noise removal on images corrupted by various sources of noise. The noise can be additive (gaussian noise, slat and pepper noise) or multiplicative (Speckle noise). In each case an filter adapted to the problem must be selected and we will see a few examples of well adapted filters for each case.

Gaussian additive noise:

Using the imnoise function, it is possible to add a specific type of noise to an image.
For example you could do:

```
ima = imread('peppers.png');
ima = rgb2gray(ima);
ima1 = imnoise(ima,'gaussian',0.2,0.01);
```

Note that the mean and variance are in [0,1] as a percentage of the max gray level.

It can be shown that gaussian or mean filters are best used to remove this type of noise.
Try the Analyse_Filter function again to see how you can denoise the image. However, there is not theoretical backing to this assertion and adaptive filtering is a better solution in most case. Try using wiener2 and compare the results. You should learn more about adaptive filtering later in the term.

Salt and Pepper noise:

Using the imnoise function, it is possible to add a specific type of noise to an image.
For example you could do:

```
ima = imread('peppers.png');
ima = rgb2gray(ima);
ima1 = imnoise(ima,'salt & pepper',0.02);
```

It can be shown that median filters are best used to remove this type of noise.
Try the medfilt2 function (help medfilt2) and see how you can denoise the image.

Speckle noise:

This is a multiplicative noise, very frequent in Radar and Satellite imagery. It is very difficult to remove as it is multiplicative and very often correlated to the signal. Try:

```
ima = imread('peppers.png');
```

ima = rgb2gray(ima);
ima1 = imnoise(ima,'speckle',0.02);
And try various filters on it to remove noise…
Comments?

Deconvolution:

A very quick word on deconvolution…
Imagine that the real scene that you are after has been distorted by the imaging system that you are using (out of focus lens, atmospheric perturbation,…). What you see is the result of the convolution of the real scene by a known (or unknown) transfer function. When the transfer function is known, you can, in the frequency domain, divide the spectrum of the observed image by the spectrum of the transfer function and recover the original image. To illustrate this, try the deconvolution program. Edit the program and change the level of noise to see the effect of instabilities in the filter.
However, direct deconvolution is very sensitive to noise as you might not have access to the exact transfer function and also because you are dividing by the spectrum of the transfer function (inverse problem). For frequencies where the signal in the image is small and the noise signal is important, you are going to amplify the noise…
Better techniques can be used such as again adaptive filtering (wiener).
If the convolution kernel (transfer function) is not known, then we are confronted to a problem called blind deconvolution where we are trying to find the convolution kernel from the data by imposing constraints on the type of images we should observe. This is an on-going area of research and is outside the scope of this course. The matlab demo gives examples of blind deconvolution using the classical Lucy-Richardson algorithm. You are welcome to investigate in your spare time.