Worksheet 2
Spatial Filtering

1. The "Mean" Filter (Part-1)

This filter sweeps an NxN kernel across the image using the correlation operation as explained in class to produce the smoothed output image.

(a) Open up the script "**FilterMean**". There's quite a lot going on here. Noise will be added to the image in two ways, (salt & pepper, then Gaussian), then the mean filter will be applied. Look for the following in the script.

  (i) Two functions which add different sorts of noise to the image
  (ii) The calculation of the NxN smoothing kernel
  (iii) Filtering operation applied to original and two noisy images

(b) Run the script, and when asked **set the kernel size to 3**. In the Command Window type **kernel** and you will see the smoothing kernel displayed. Its values should sum to 1.0. Check this by typing **sum(sum(kernel))** to sum both columns and rows.

(c) Check out the original and smoothed images. What has smoothing done to each image? To what extent has the noise been reduced? Any other visible changes?

(d) Now repeat with a kernel size of 11. Dump the kernel values in Command Window. Now check out the images. Has smoothing worked to reduce the noise? Are there any other visible changes? (Hint – look at the transformation on the *original* image).

(e) Any conclusions on how to choose the size of the kernel?
(f) You may wish to experiment with other kernel sizes.

Purpose	Images may contain noise. When they are transformed (e.g., contrast enhancement) then the noise may become more visible. It is usual to "smooth" images, reducing the amount of noise in the image. This makes them more agreeable to the viewer. This process is called "Image Smoothing"
Files Required	Octave scripts in the zipped folder "Octave_ImProc_Release" on the module site.
ILO Contribution	Position Paper 1 (part 1) Learning outcome 2.
Send to Me	
Homework	Read Chapter 1

2. The "Mean" Filter (Part-2)

Here we shall work with a small 1D synthetic image to try and increase our understanding of what is going on. The synthetic image has a block of low values on the left, and high values on the right, i.e., there is a vertical edge in the image. Noise will play havoc with this edge. You will see a plot of this 1D image, you could think of this as a 'slice' of a 2D image, say one row of pixel values.

(a) Run the script **NumFilerMeanEdgeRev** and when asked set (i) input size = 10, (ii) noise = 0, (iii) Input kernel size 3. What has happened to the pixels near the edge? To the image far left and far right?

(b) Type the following in the Command Window (i) "**I**" which will show the numeric values of the input image (ii) "**I_f**" which will show the numeric values of the filtered image (iii) "**kernel**" which will show the numeric values of the kernel.

(c) Looking at the input image and the kernel, calculate the output values when the kernel is centred at the index 5 in the original image. Now try at index 2.

This should help you understand the operation of correlation. You may wish to take notes and records of these numbers.

(d) Run the script again but now with (i) input size = 100, (ii) noise = 0 and let's change the kernel size. Look at the graphs which show you what is happening at the *middle row* of input and smoothed image.
So try kernel sizes of 3,5,7,11,15 and look at what happens to the edge. Does this agree with your results from Activity 1?

(e) Now let's work with noise. Run some experiments with (i) input size = 100, (ii) noise = 0.01 and change the kernel sizes 3,5,7,9,11,15.
(i) What happens to the noise when you increase the kernel size?
(ii) What happens to the edge when you increase the kernel size?

(f) Do you understand that when you increase the kernel size, one image property improves but another property gets worse. Sound like a *tradeoff*, methinks.

3. Gaussian Filter applied to real images.

Let's run the spatial filter "FilterGauss" on some images.

(a) Open up the script in the Octave Editor, and make sure you understand what is going on, especially where the Gaussian kernel is being set up.

(b) Now investigate the four combinations of values in this table and make some conclusions about which combination works best. Remember you can see the kernel values by typing **kernel** at command line.

image size	Kernel size	sigma
100	11	2
100	11	9
100	5	2
100	5	9