# Digital Image Processing Lecture 3

Muhammad Usman Ghani Khan

Department of Computer Science & Engineering
UET, Lahore

Spring Semester

# Outline

# Agenda

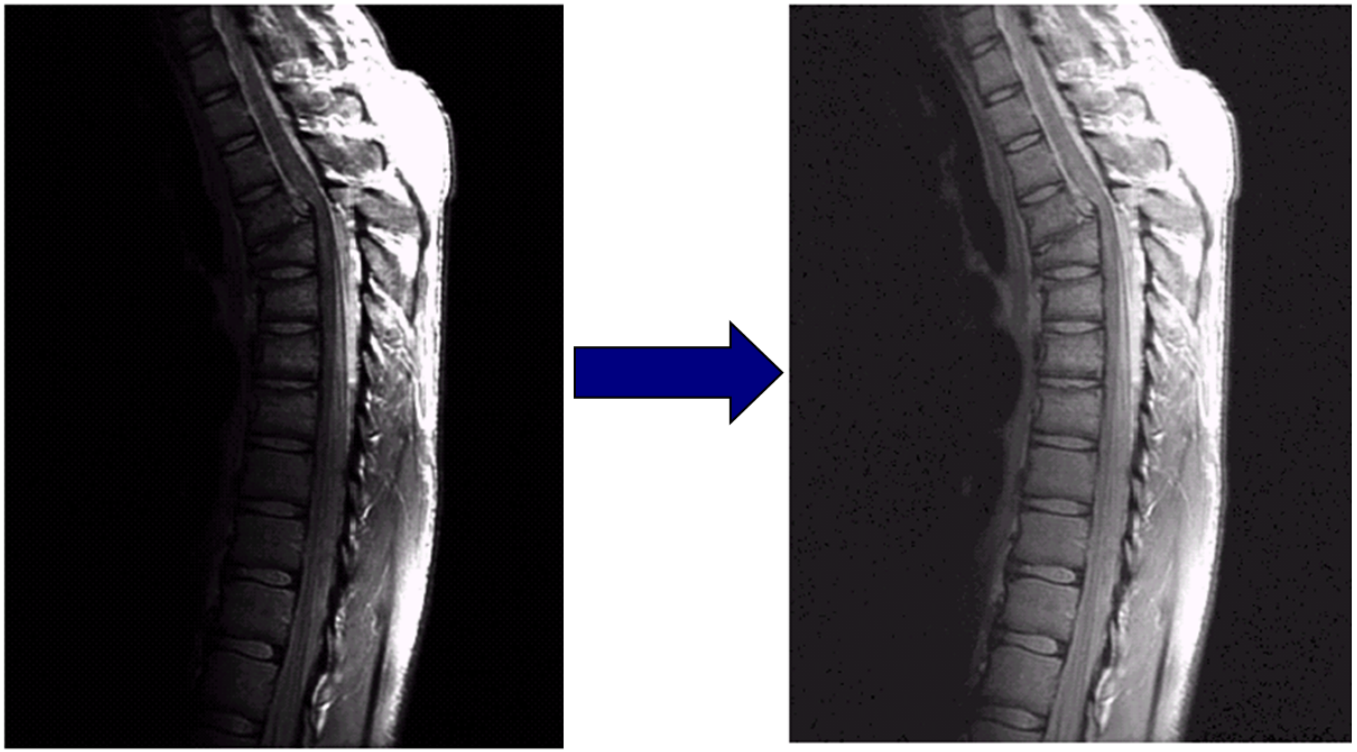
## Today Discussion

- What is image enhancement?
- Different kinds of image enhancement
- Histogram processing
- Point processing
- Neighbourhood operations

# What is image enhancement?

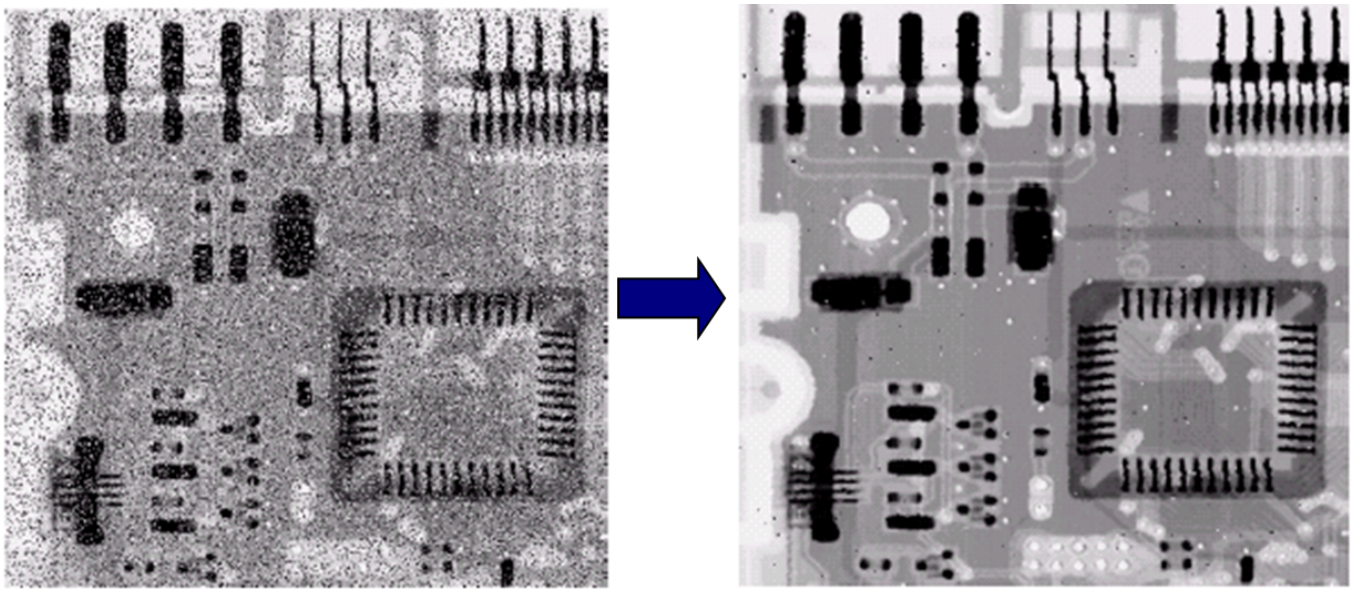
- Image enhancement is the process of making images more useful. The reasons for doing this include:
  - Highlighting interesting detail in images
  - Removing noise from images
  - Making images more visually appealing

# Image Enhancement: Example 1



# Image Enhancement: Example 2

# Image Enhancement: Example 3



# Image Enhancement: Example 4

# Spatial & Frequency Domains

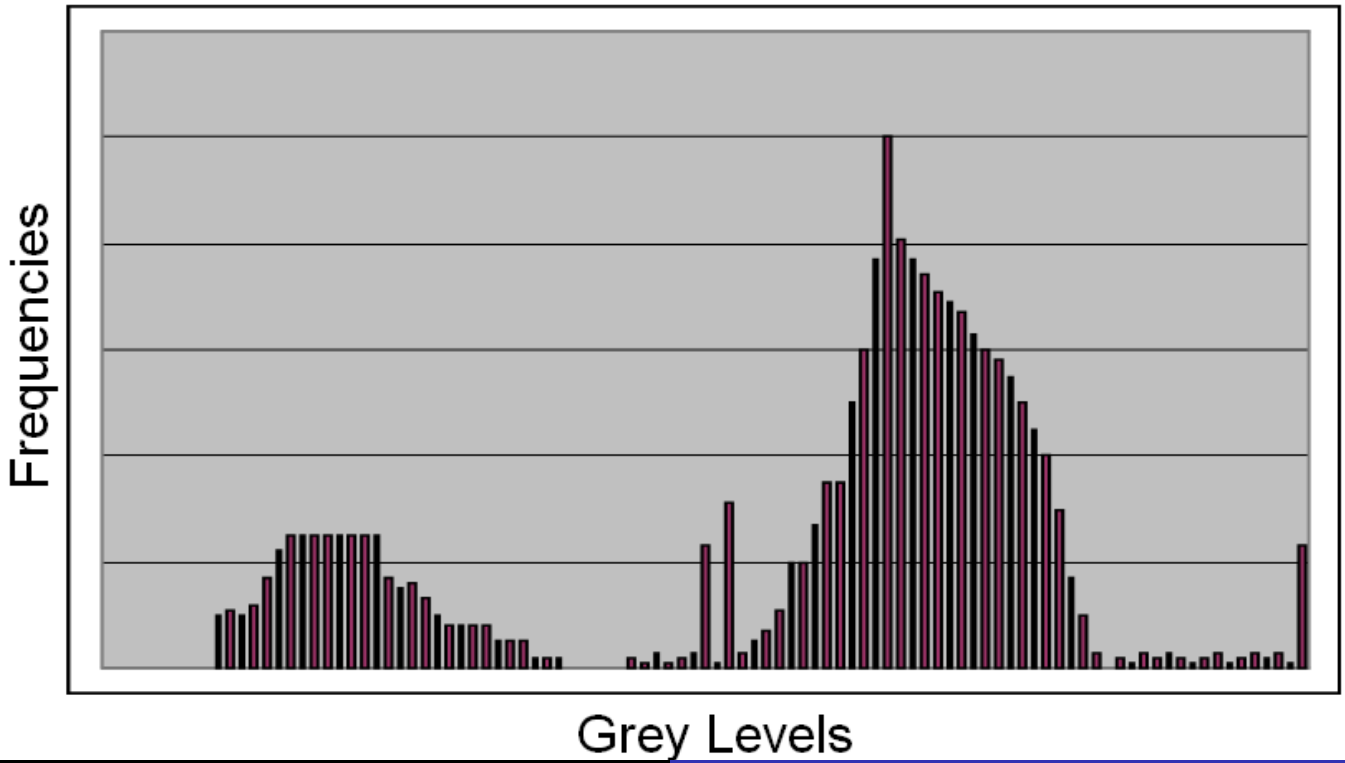
There are two broad categories of image enhancement techniques

1. Spatial domain techniques
   - Direct manipulation of image pixels
2. Frequency domain techniques
   - Manipulation of Fourier transform or wavelet transform of an image

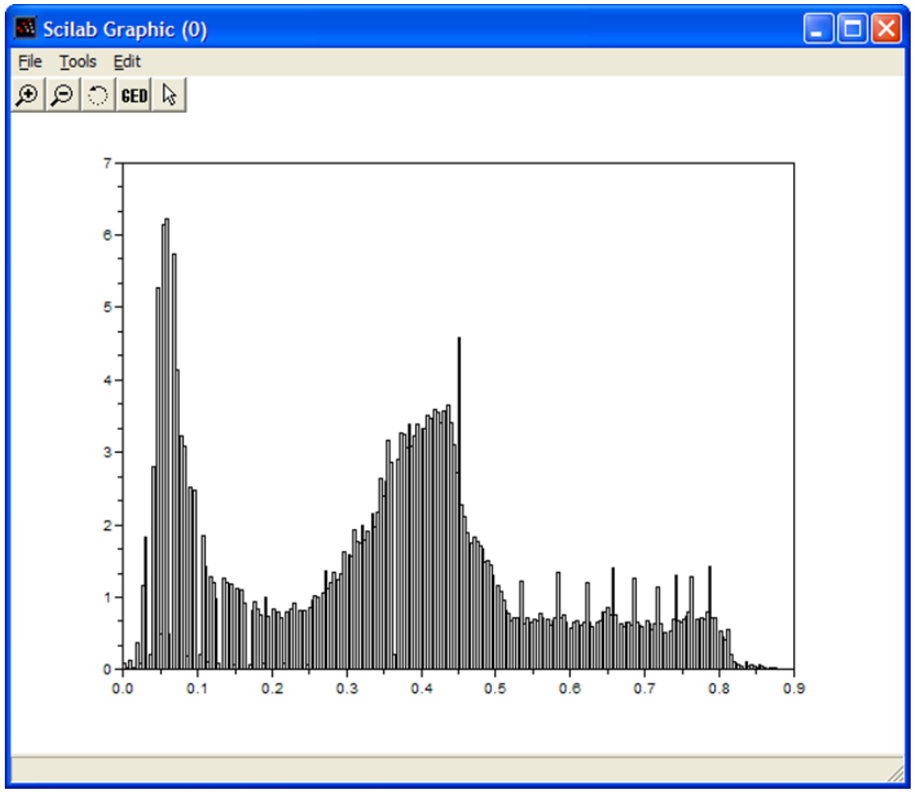
For the moment we will concentrate on techniques that operate in the spatial domain

# Image Histograms

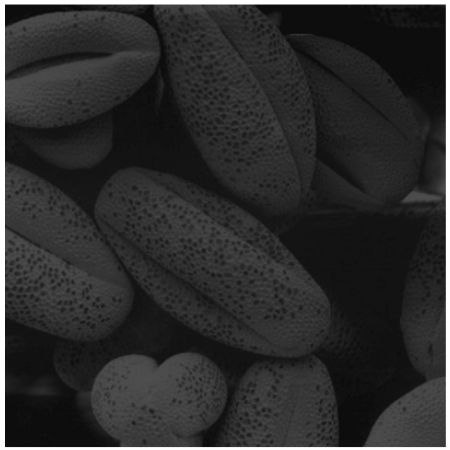
- The histogram of an image shows us the distribution of grey levels in the image
- Massively useful in image processing, especially in segmentation

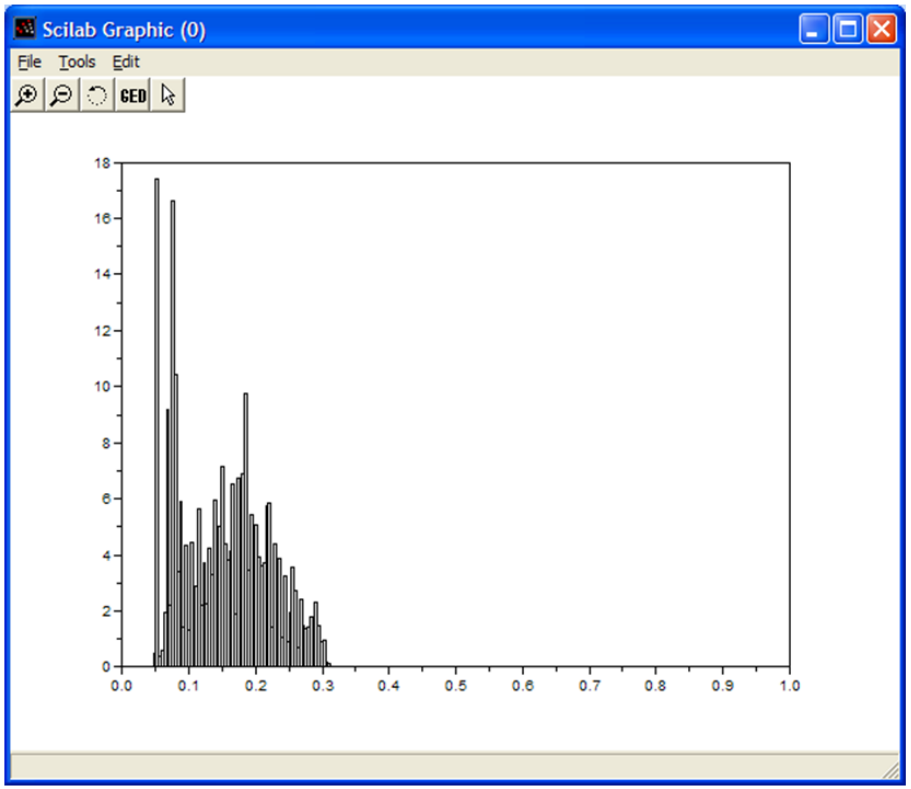


# Histogram: Example 1

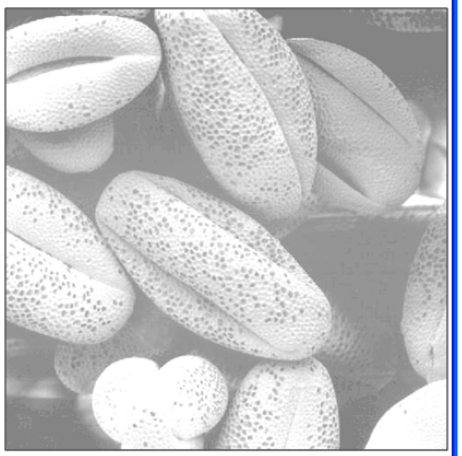


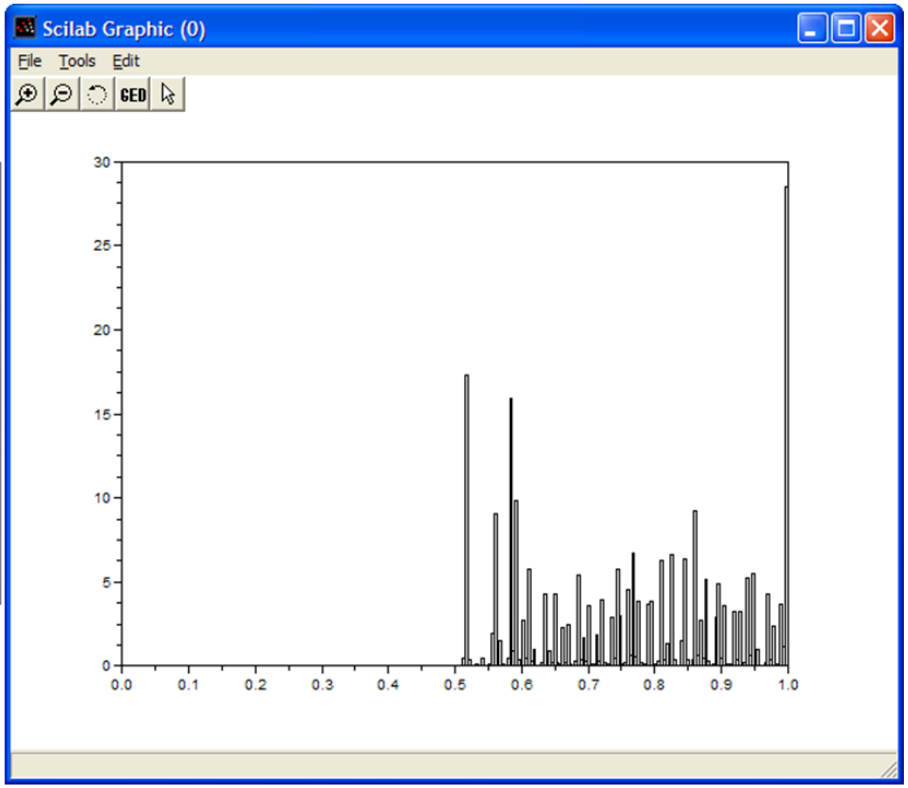
# Histogram: Example 2



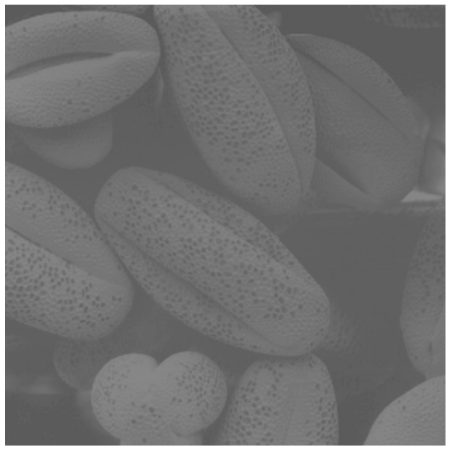


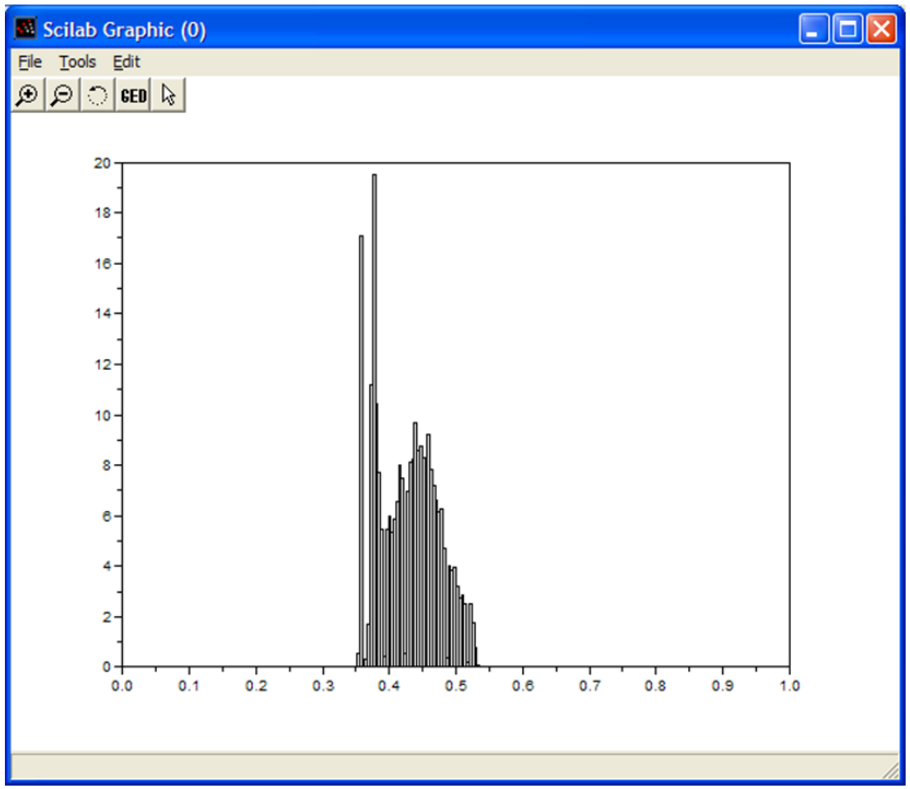
# Histogram: Example 3



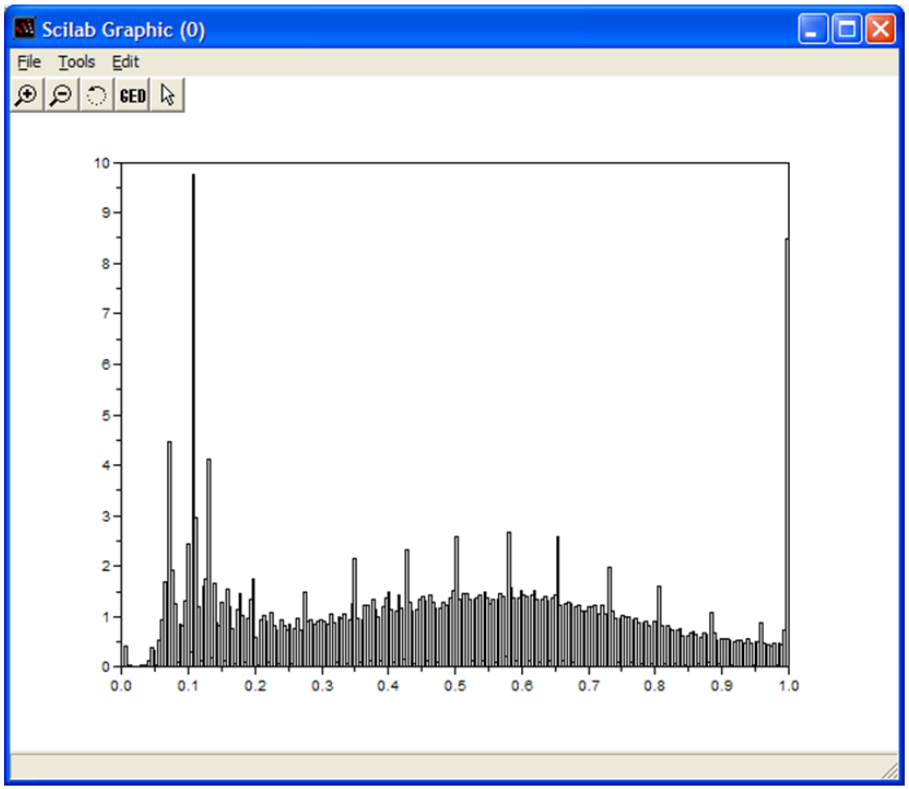


# Histogram: Example 4

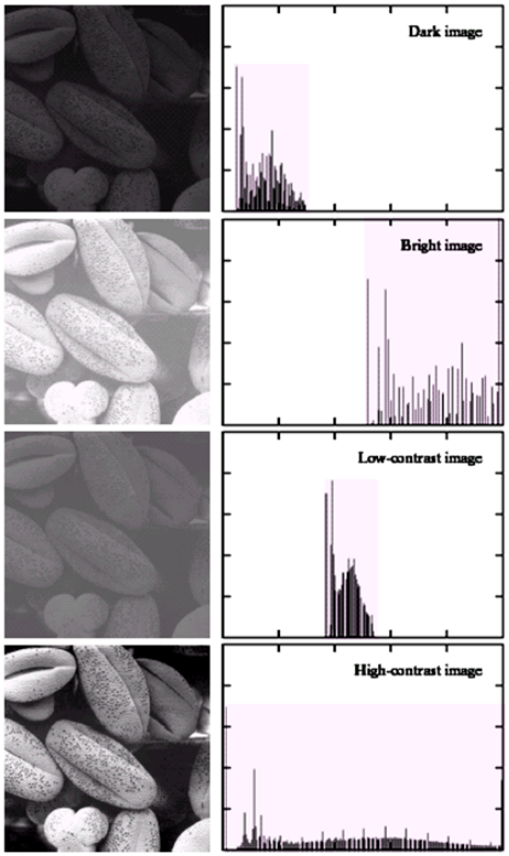




# Histogram: Example 5



# Histograms Examples: Final Note




- A selection of images and their histograms
- Notice the relationships between the images and their histograms
- Note that the high contrast image has the most evenly spaced histogram

# Outline

# Contrast Stretching

- We can fix images that have poor contrast by applying a pretty simple contrast specification
- The interesting part is how do we decide on this transformation function?

# Histogram Equalisation

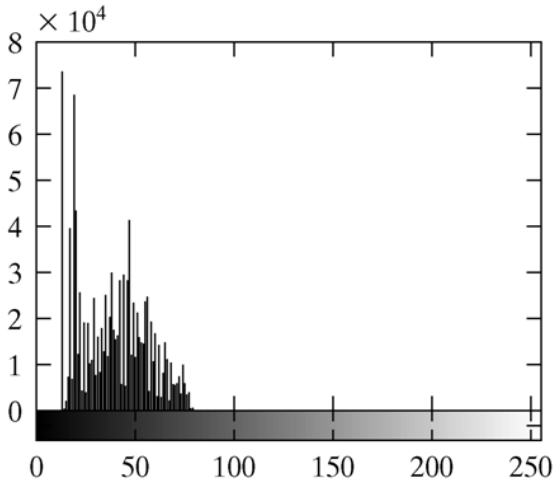
- Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

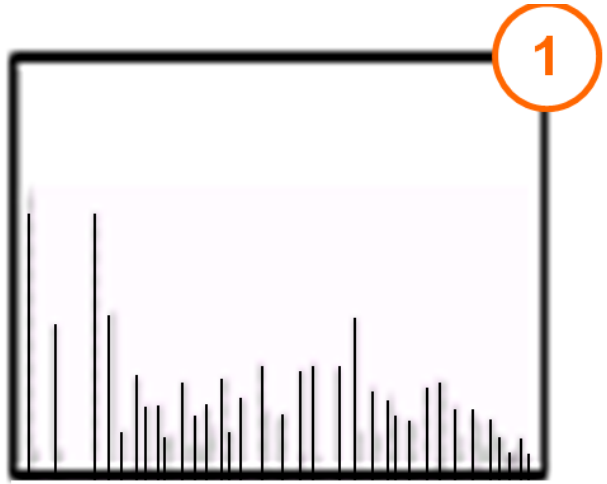
$$S_k = T(r_k) = \sum_{j=1}^{k} P_r(r_j) = \sum_{j=1}^{k} \frac{n_j}{n} \tag{1}$$

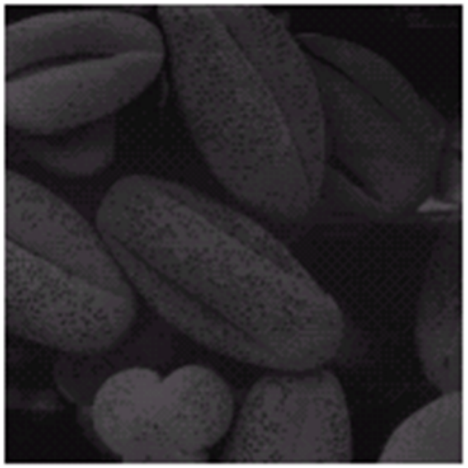
where

- $r_k$ : input intensity
- $s_k$ : processed intensity
- $k$ : the intensity range (*e.g.*, 0.0 to 1.0)
- $n_j$ : the frequency of intensity $j$
- $n$ : the sum of all frequencies

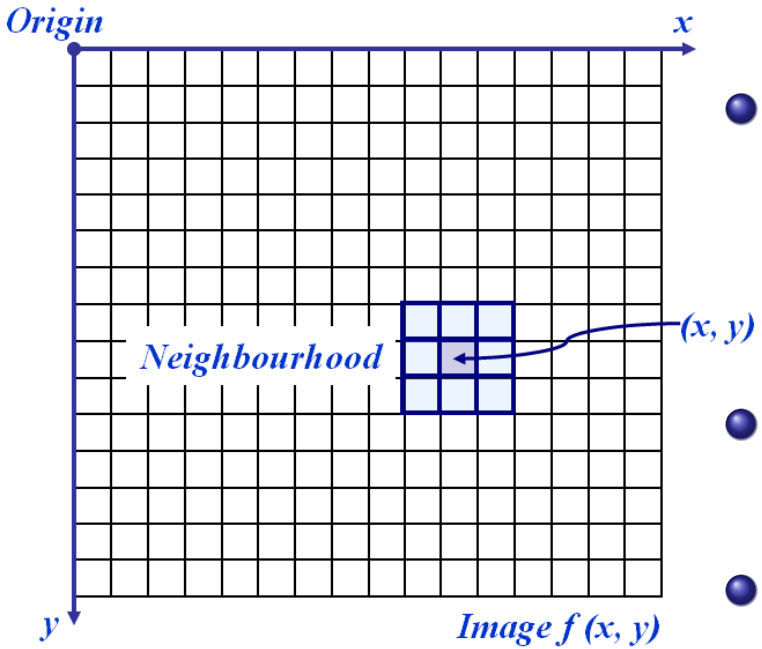
# Equalisation Examples



# Neighbourhood Operations




- Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations
- Neighbourhoods are mostly a rectangle around a central pixel
- Any size rectangle and any shape filter are possible

# Basics of Spatial Filtering

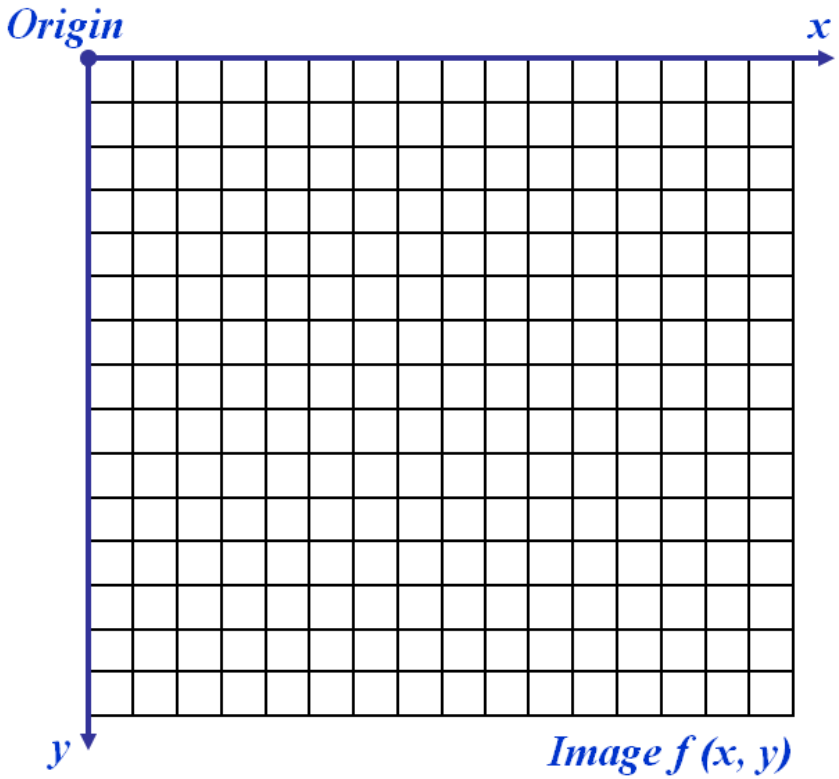
1. Image enhancement operations can also work with the values of the image pixels in the neighborhood and the corresponding values of a subimage that has the same dimensions as the neighborhood.
2. The subimage is called a filter, mask, kernel, template, or window.
3. The values in a filter subimage are referred to as coefficients, rather than pixels.
4. The process consists simply of moving the filter mask from point to point in an image.
5. For linear spatial filtering, the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask

# Simple Neighbourhood Operations

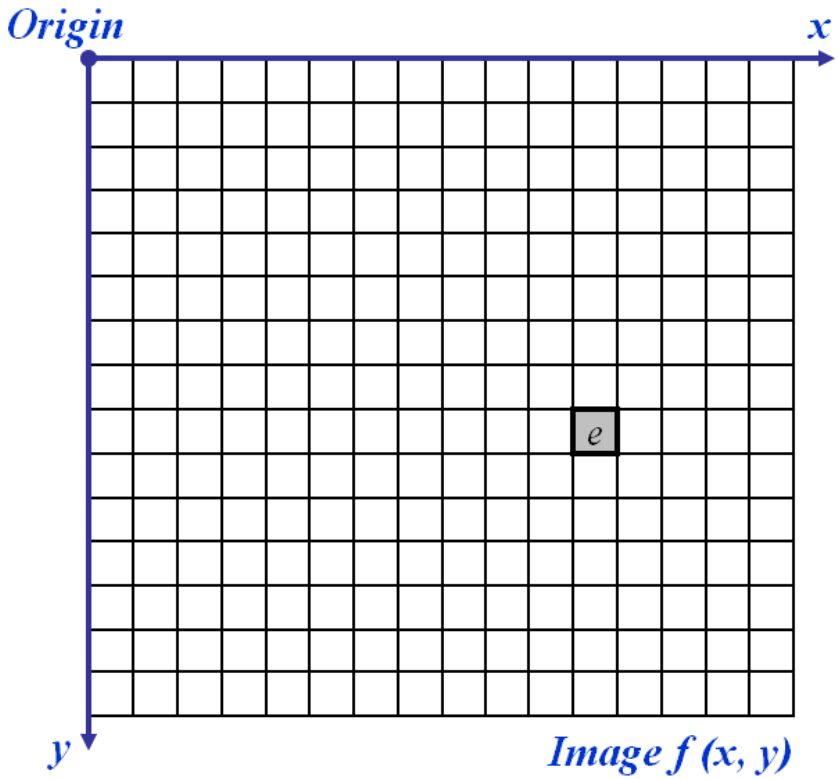
Some simple neighbourhood operations include:

1. Min: Set the pixel value to the minimum in the neighbourhood
2. Max: Set the pixel value to the maximum in the neighbourhood
3. Median: The median value of a set of numbers is the midpoint value in that set

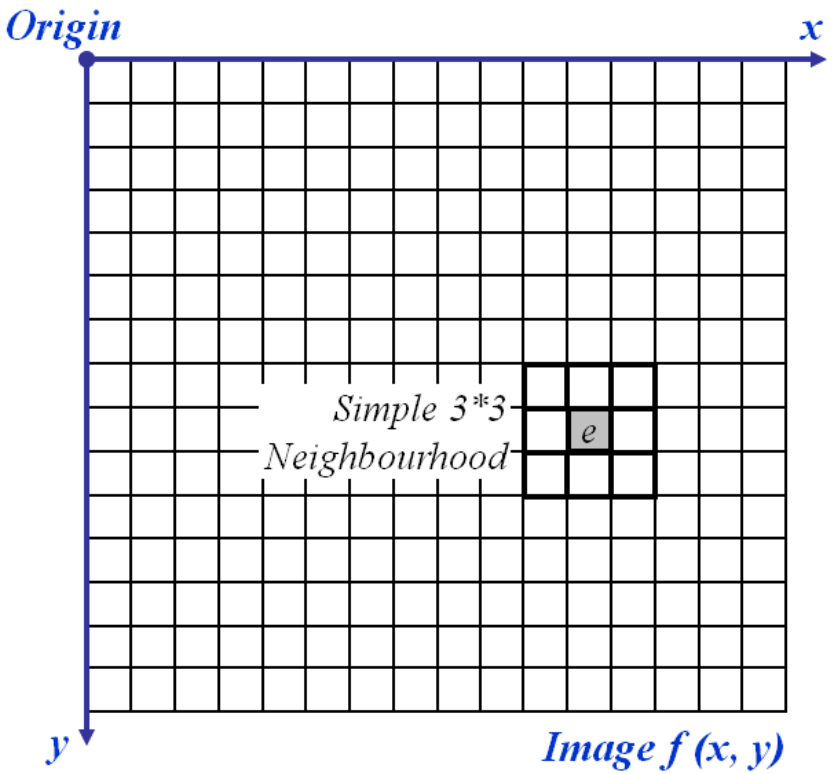
# The Spatial Filtering Process



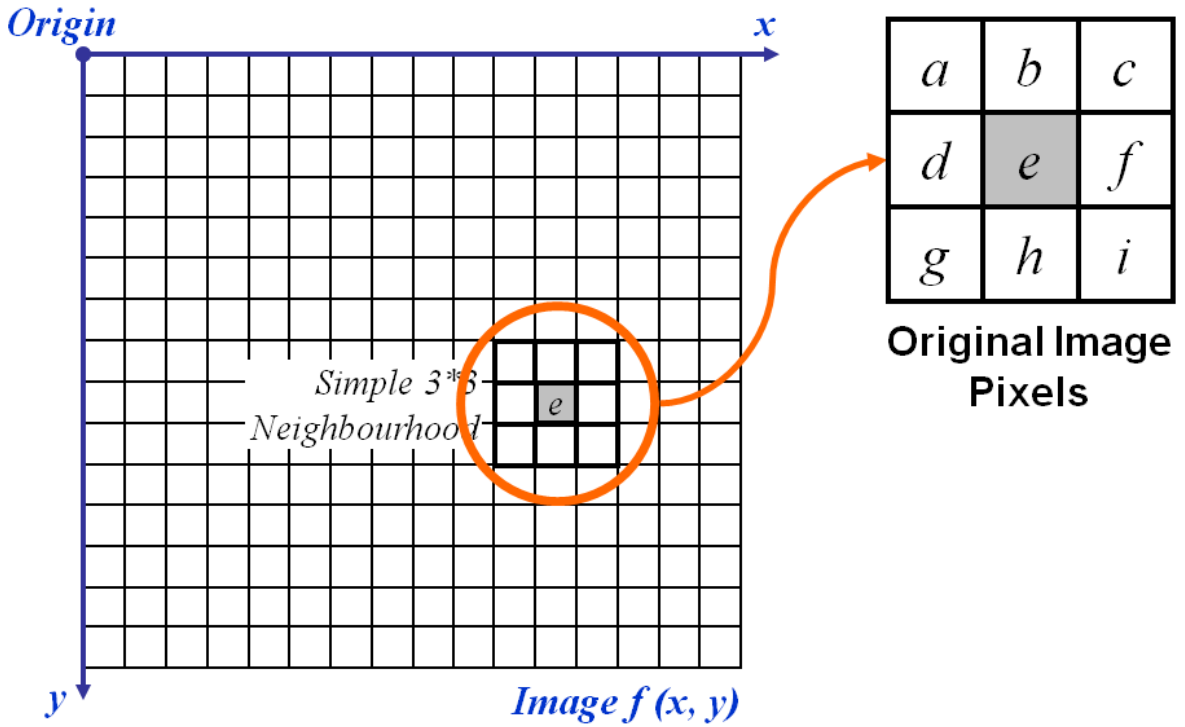
# The Spatial Filtering Process



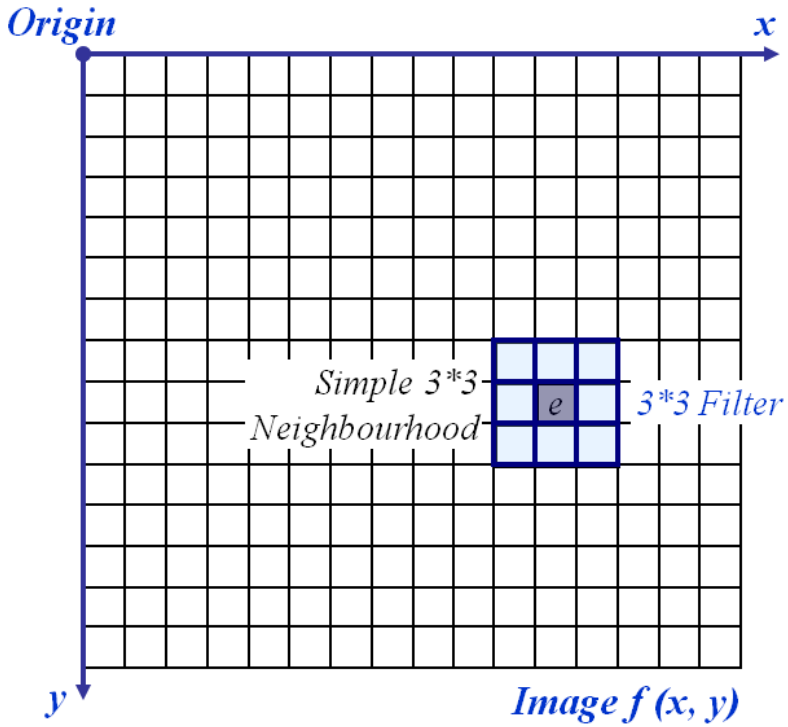
# The Spatial Filtering Process



# The Spatial Filtering Process



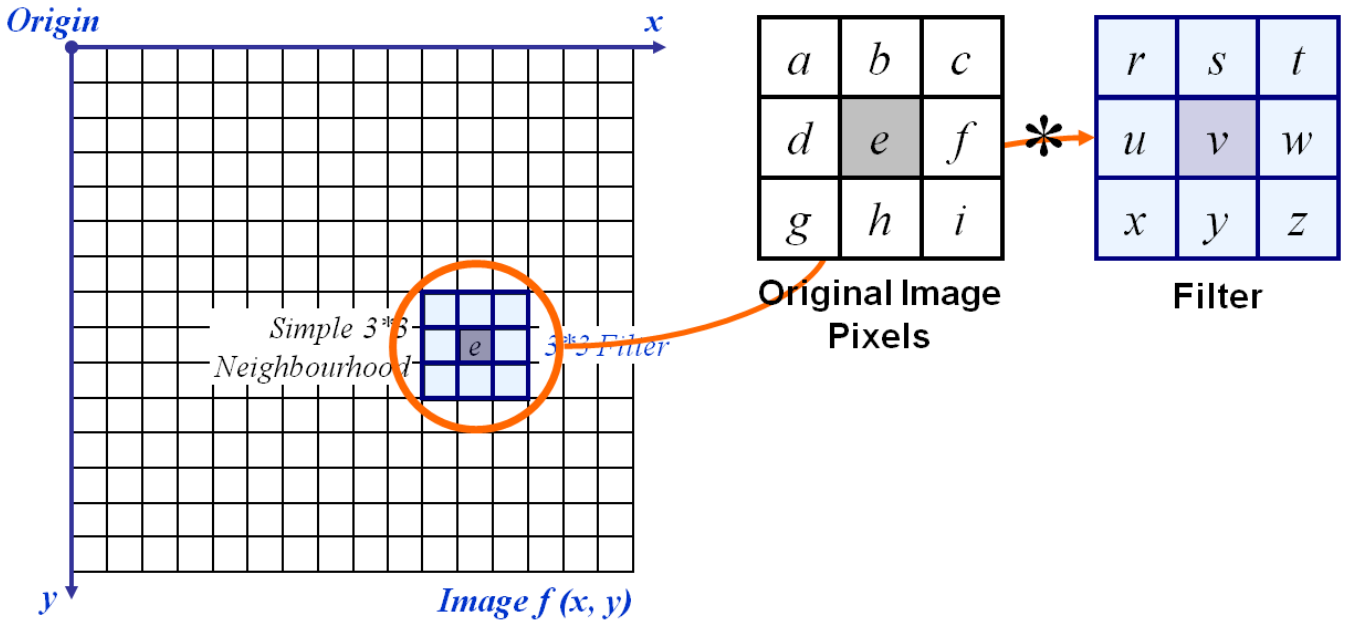
# The Spatial Filtering Process



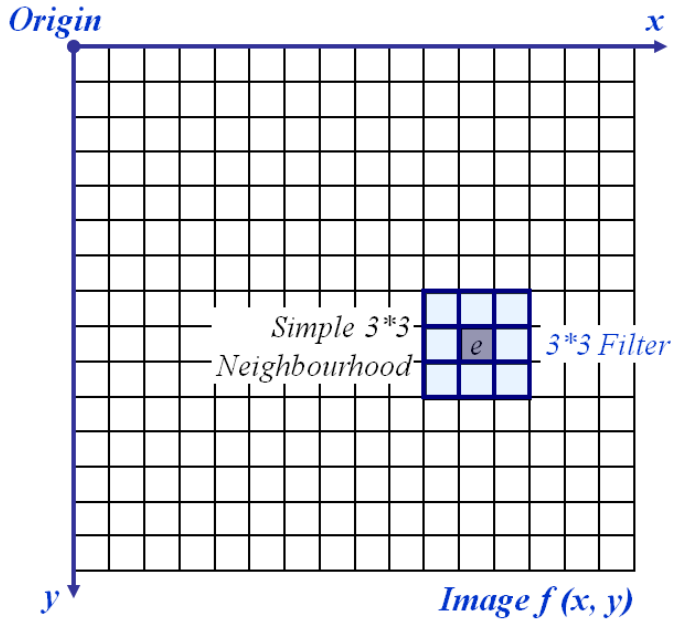

| | | |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $e$ | $f$ |
| $g$ | $h$ | $i$ |

**Original Image Pixels**

# The Spatial Filtering Process



# The Spatial Filtering Process




| | | |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $e$ | $f$ |
| $g$ | $h$ | $i$ |

Original Image Pixels

$*$

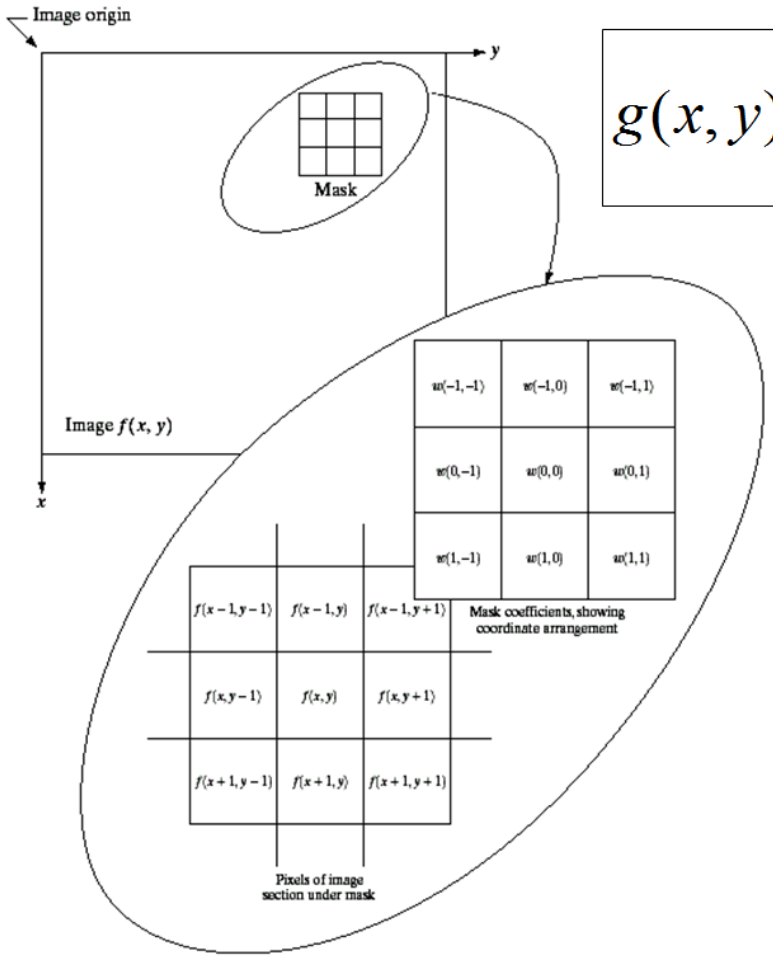
| | | |
|---|---|---|
| $r$ | $s$ | $t$ |
| $u$ | $v$ | $w$ |
| $x$ | $y$ | $z$ |

Filter

$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the filtered image

# Spatial Filtering: Equation Form




$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

# Outline

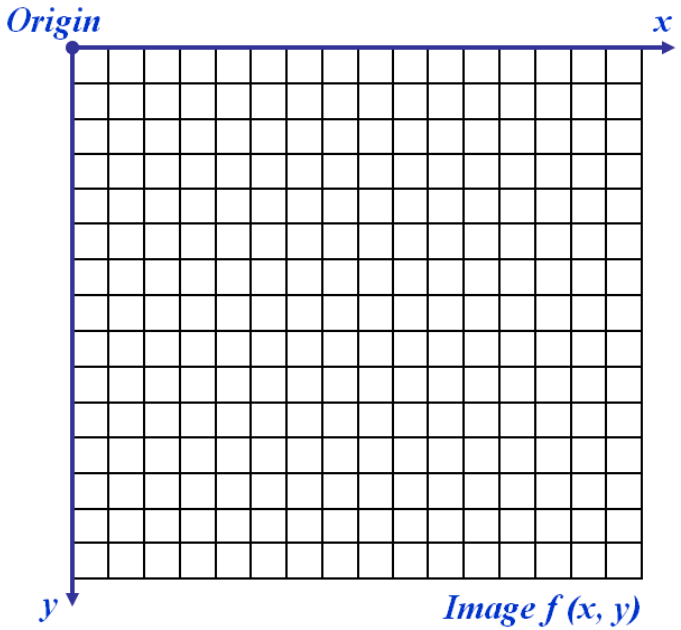
# Smoothing Spatial Filters

One of the simplest spatial filtering operations we can perform is a smoothing operation

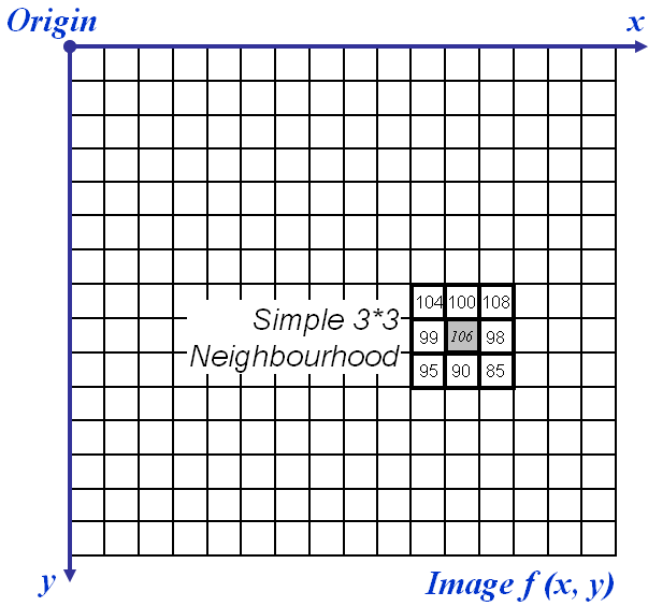
- Simply average all of the pixels in a neighbourhood around a central value
- Especially useful in removing noise from images
- Also useful for highlighting gross detail

| | | |
|---|---|---|
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |

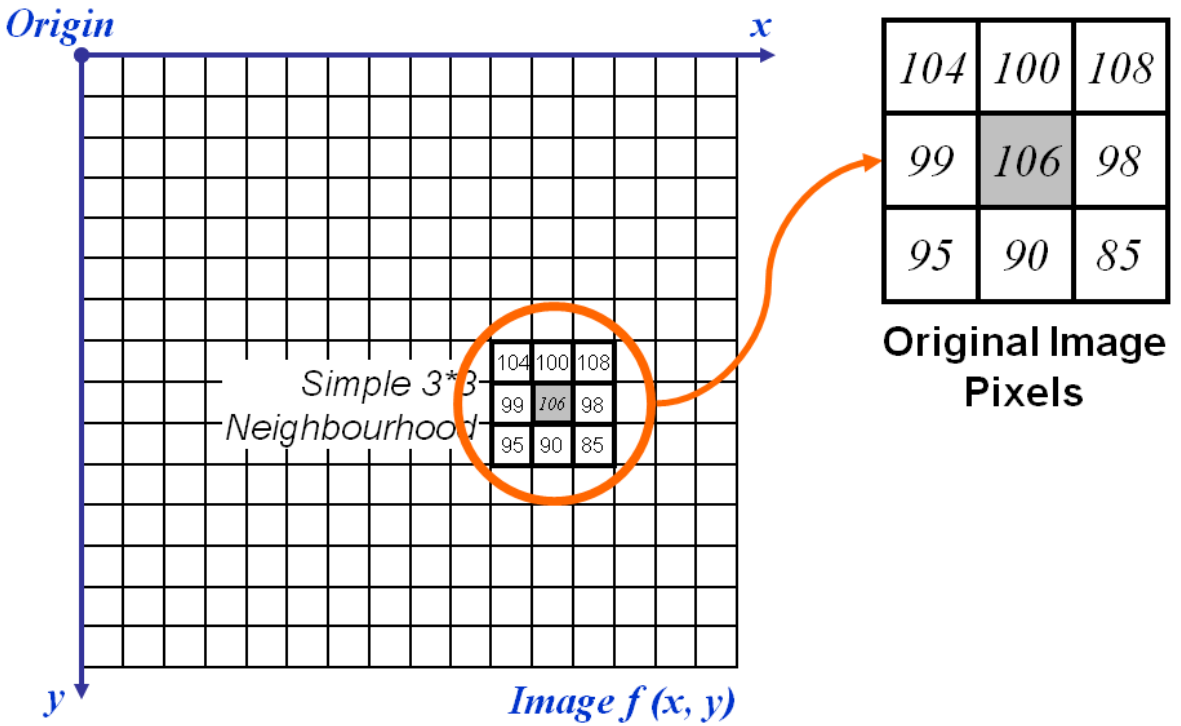
# Smoothing Spatial Filtering



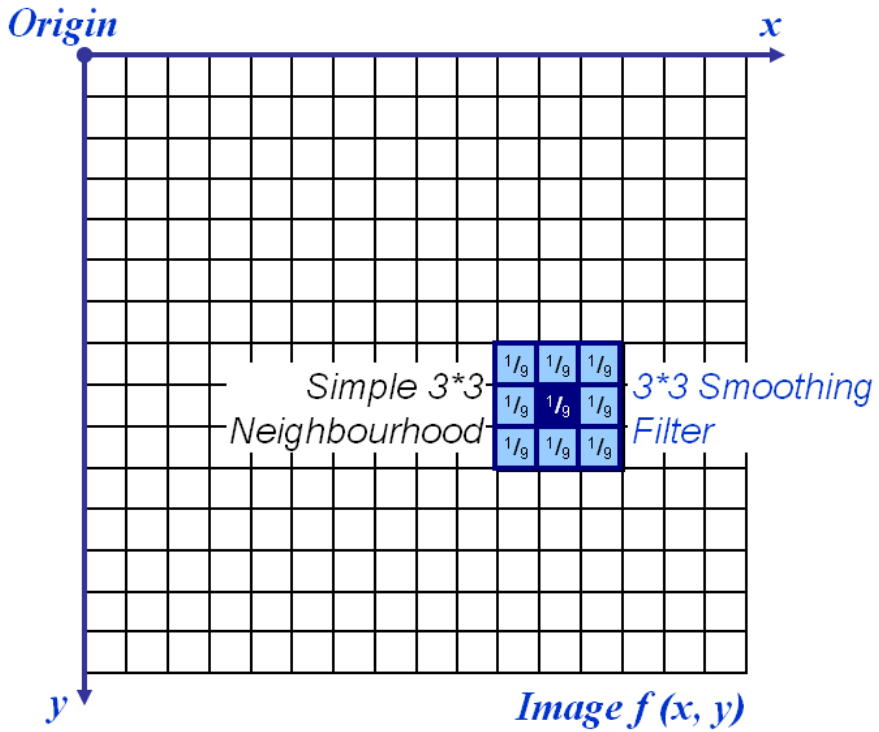
# Smoothing Spatial Filtering



# Smoothing Spatial Filtering



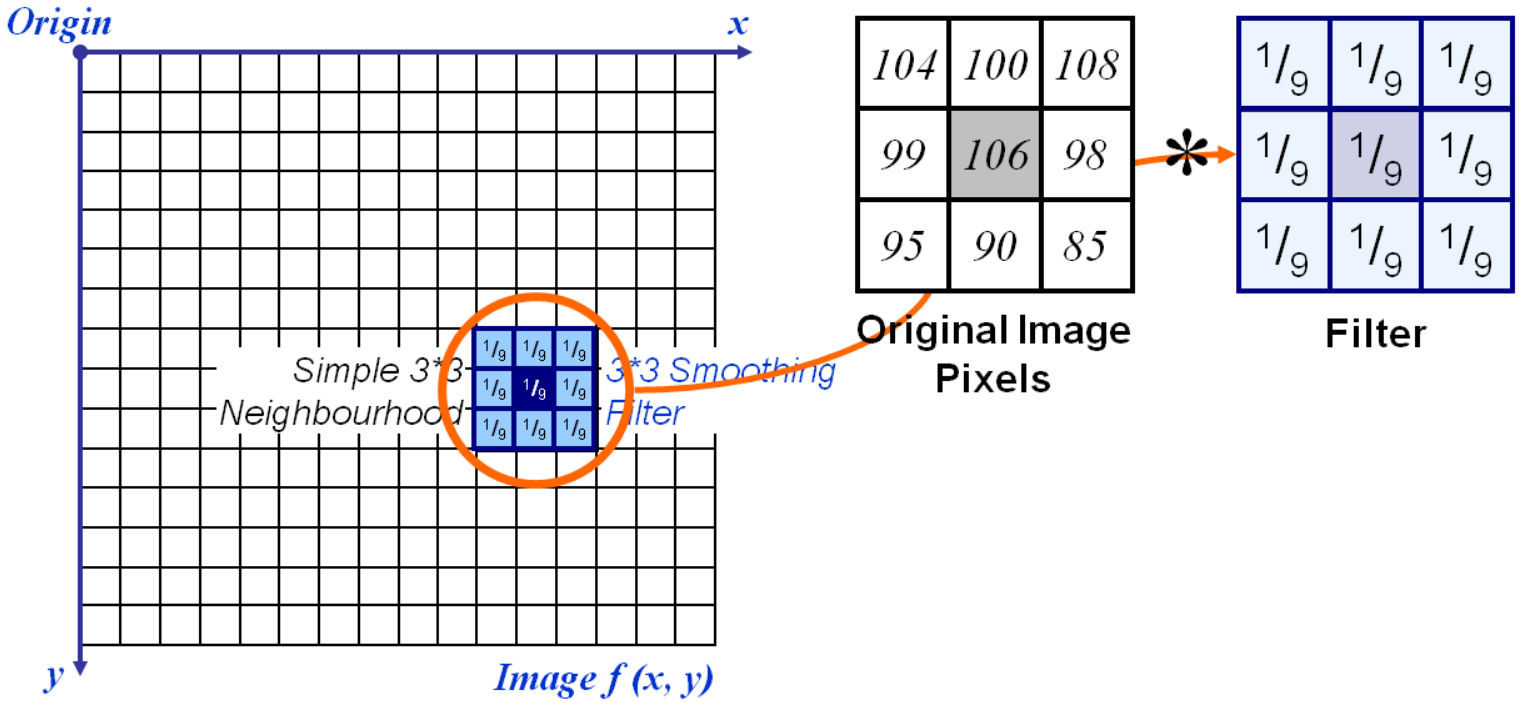
# Smoothing Spatial Filtering



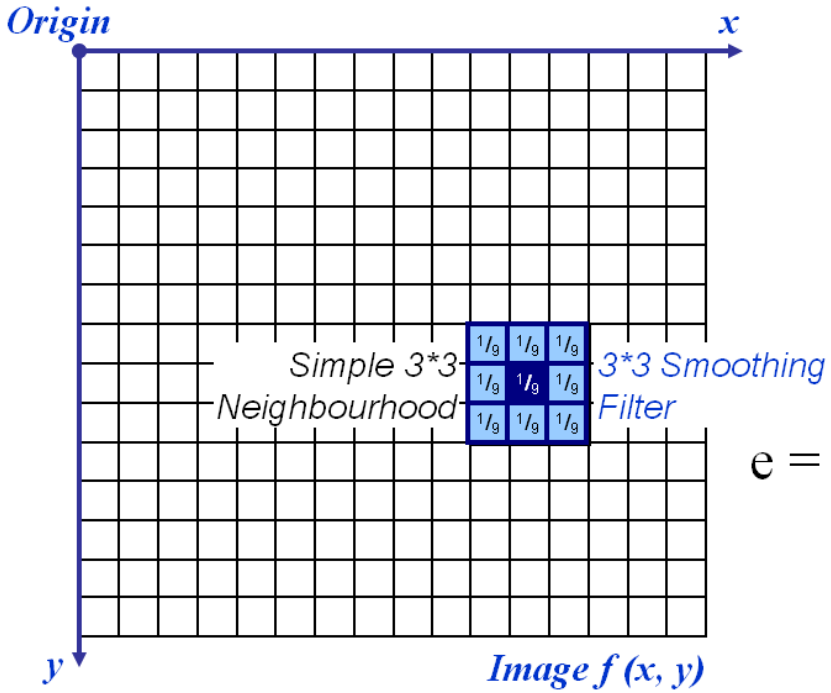

| | | |
|---|---|---|
| 104 | 100 | 108 |
| 99 | 106 | 98 |
| 95 | 90 | 85 |

**Original Image Pixels**

# Smoothing Spatial Filtering



# Smoothing Spatial Filtering




| | | |
|---|---|---|
| 104 | 100 | 108 |
| 99 | 106 | 98 |
| 95 | 90 | 85 |

**Original Image Pixels**

$*$

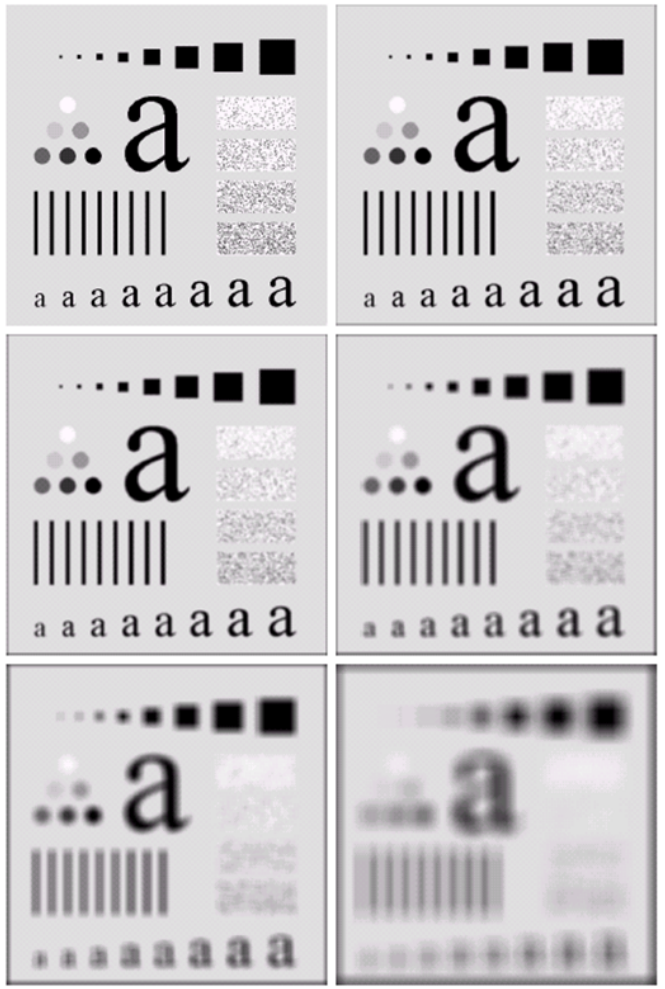
| | | |
|---|---|---|
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |

**Filter**

$$
\begin{aligned}
e = {} & 1/9*106 + \\
& 1/9*104 + 1/9*100 + 1/9*108 + \\
& 1/9*99 + 1/9*98 + \\
& 1/9*95 + 1/9*90 + 1/9*85 \\
= {} & 98.3333
\end{aligned}
$$

# Image Smoothing Example

- The image at the top left is an original image of size $500 * 500$ pixels
- The subsequent images show the image after filtering with an averaging filter of increasing sizes $3, 5, 9, 15$ and $35$
- Notice how detail begins to disappear



# Weighted Smoothing Filters

More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function

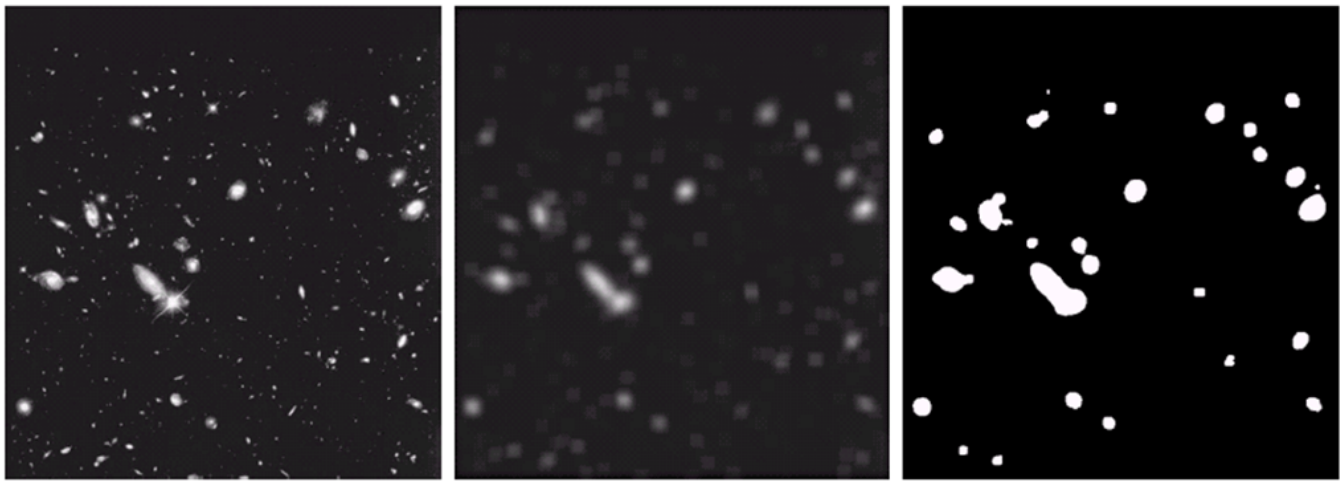
- Pixels closer to the central pixel are more important
- Often referred to as a weighted averaging

| | | |
|---|---|---|
| $1/16$ | $2/16$ | $1/16$ |
| $2/16$ | $4/16$ | $2/16$ |
| $1/16$ | $2/16$ | $1/16$ |

Weighted averaging filter

# Another Smoothing Example

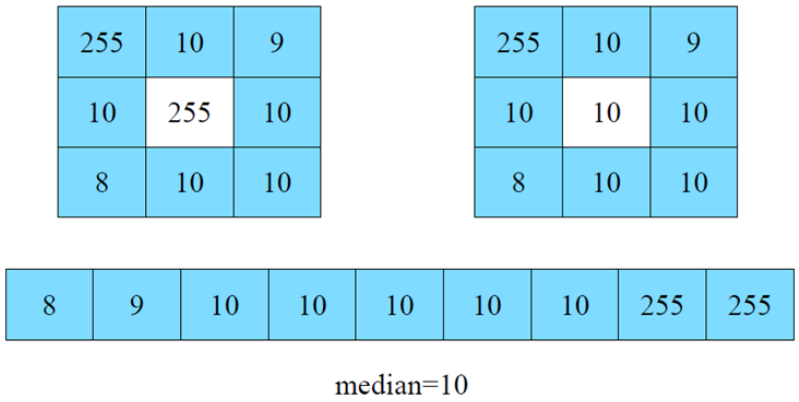
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



Original Image | Smoothed Image | Thresholded Image

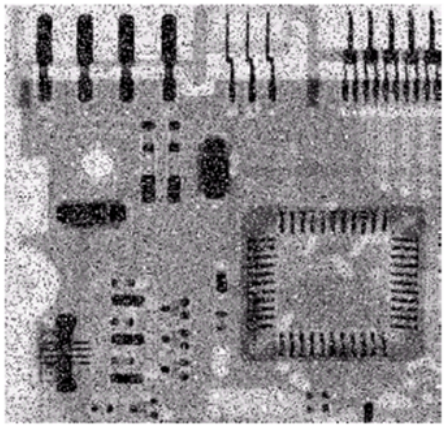
# Median Filter

- Provides good noise reduction for certain types of noise such as impulse noise
- Considerably less blurring than weighted averaging filter
- Forces a pixel to be like its neighbors
- Steps
  1. Order pixels within an area
  2. Replace value of center pixel with median value (half of all pixels have intensities greater than or equal to the median value)



# Averaging Filter vs Median Filter Example

- Filtering is often used to remove noise from images
- Sometimes a median filter works better than an averaging filter



Original Image
With Noise

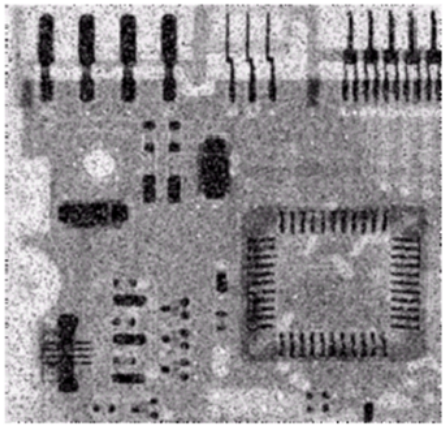


Image After
Averaging Filter

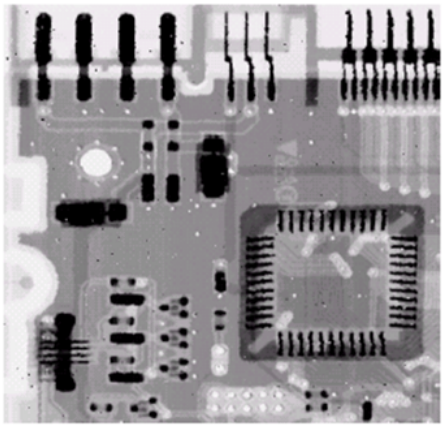
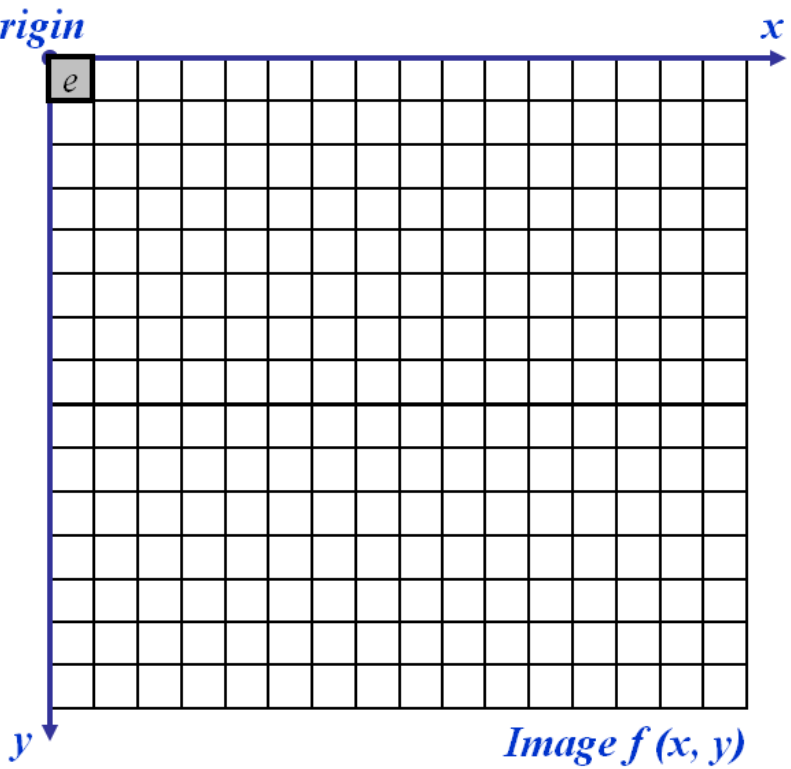


Image After
Median Filter

# Outline

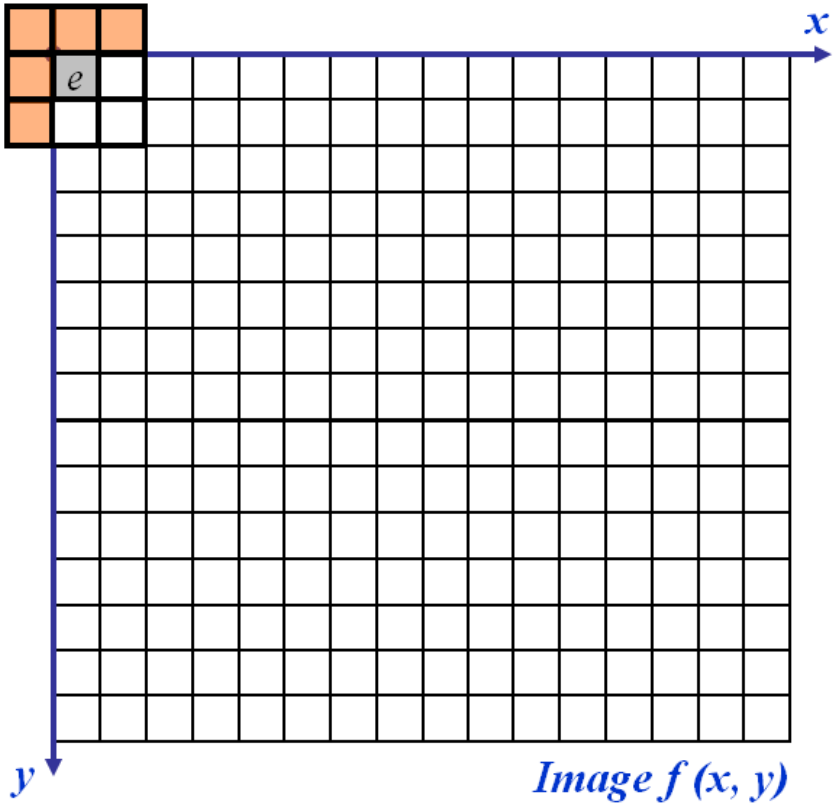
# Strange Things Happen At The Edges

At the edges of an image we are missing pixels to form a neighbourhood



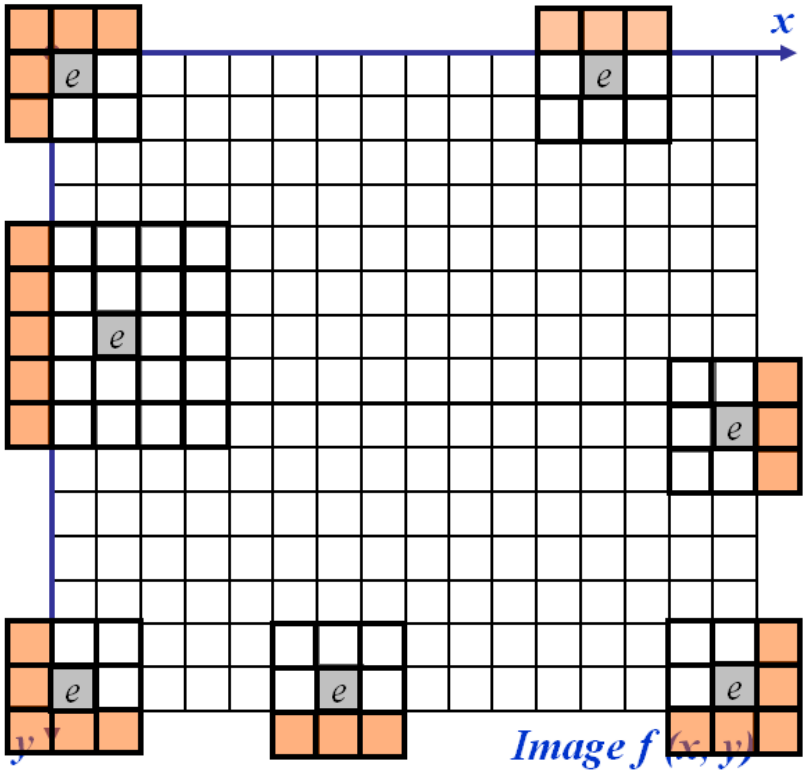
# Strange Things Happen At The Edges

At the edges of an image we are missing pixels to form a neighbourhood



# Strange Things Happen At The Edges

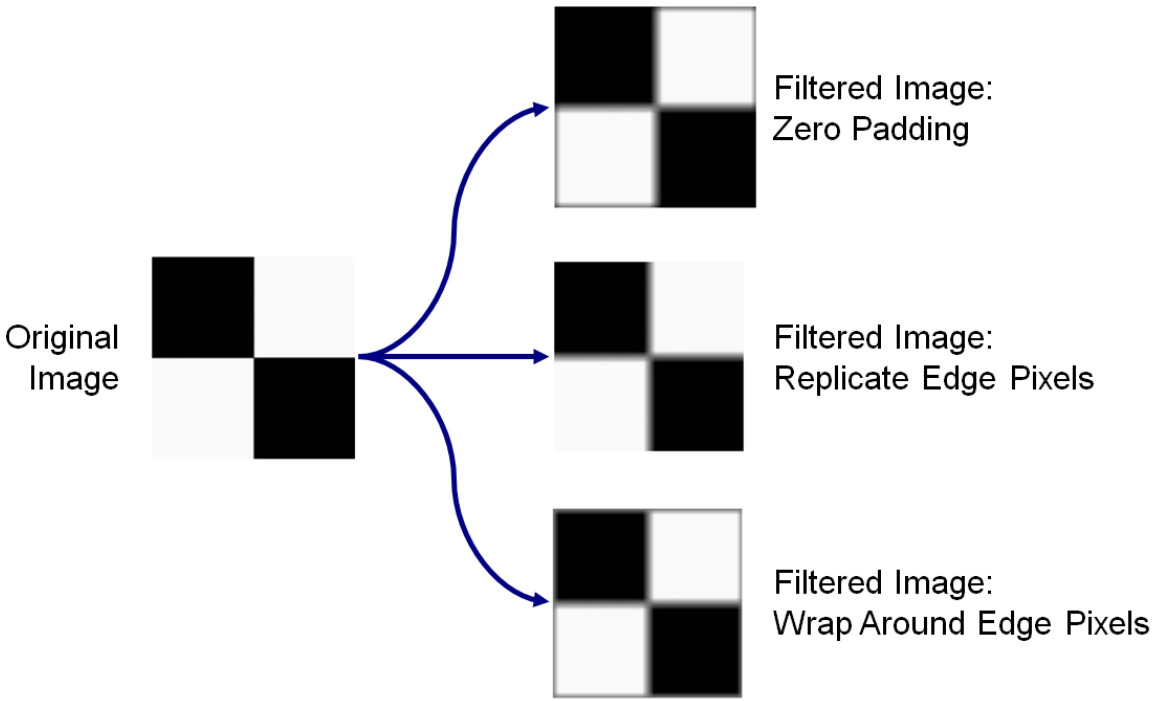
At the edges of an image we are missing pixels to form a neighbourhood



# Dealing with Edges

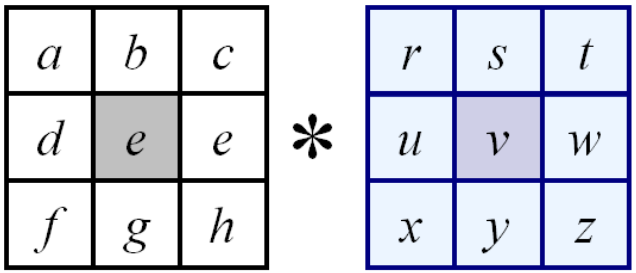
1. Omit missing pixels
   - Only works with some filters
   - Can add extra code and slow down processing
2. Pad the image
   - Typically with either all white or all black pixels
3. Replicate border pixels
4. Truncate the image
5. Allow pixels wrap around the image
   - Can cause some strange image artefacts

# Dealing with Edges — Examples



# Correlation & Convolution

- The filtering we have been talking about so far is referred to as correlation with the filter itself referred to as the correlation kernel
- Convolution is a similar operation, with just one subtle difference




$$e_{processed} = v*e + z*a + y*b + x*c + w*d + u*e + t*f + s*g + r*h$$