# Machine Vision

## Lecture 4 Part 4

## Image Enhancement.

## Neighbourhood operations

Based on lectures of

Brian Mac Namee

# Contents

In this lecture we will look at spatial filtering techniques:

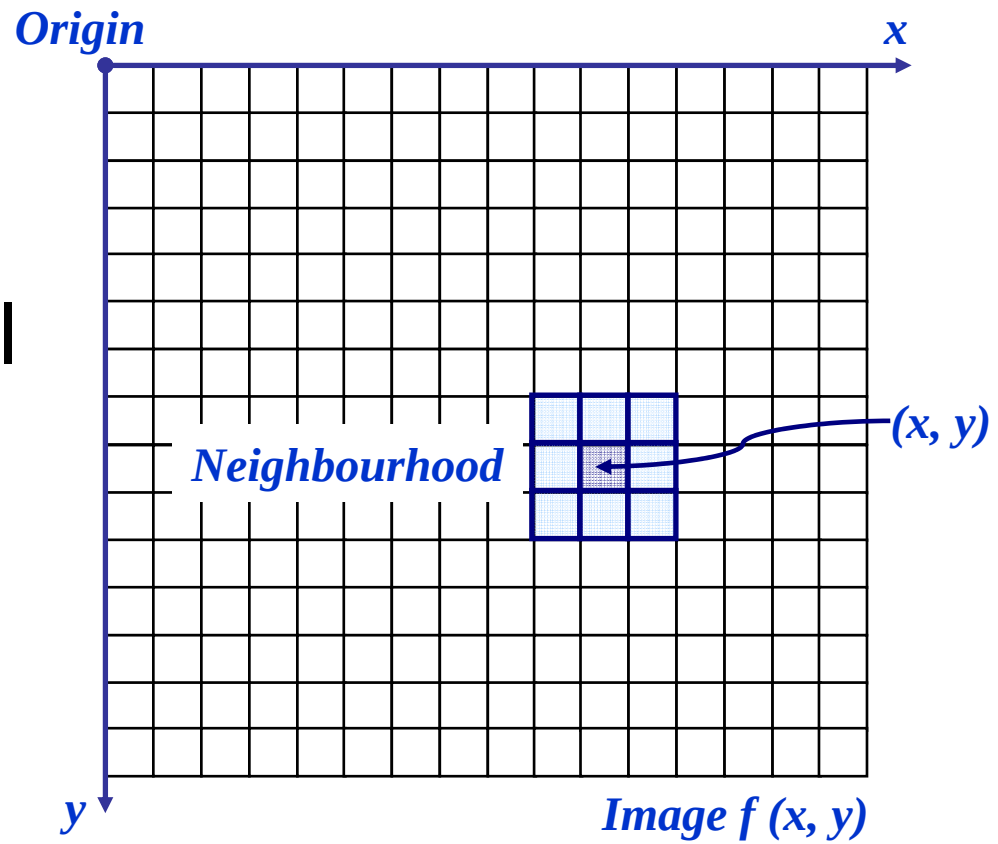
- Neighbourhood operations
- What is spatial filtering?
- Smoothing operations
- What happens at the edges?
- Correlation and convolution

# 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

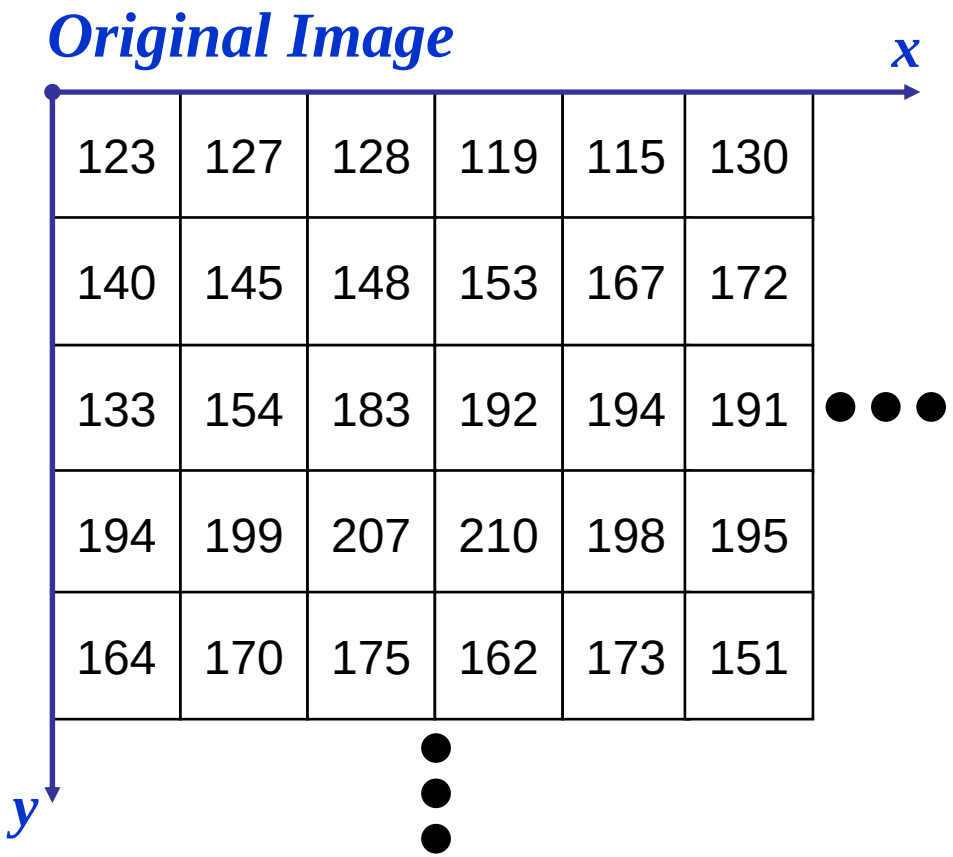


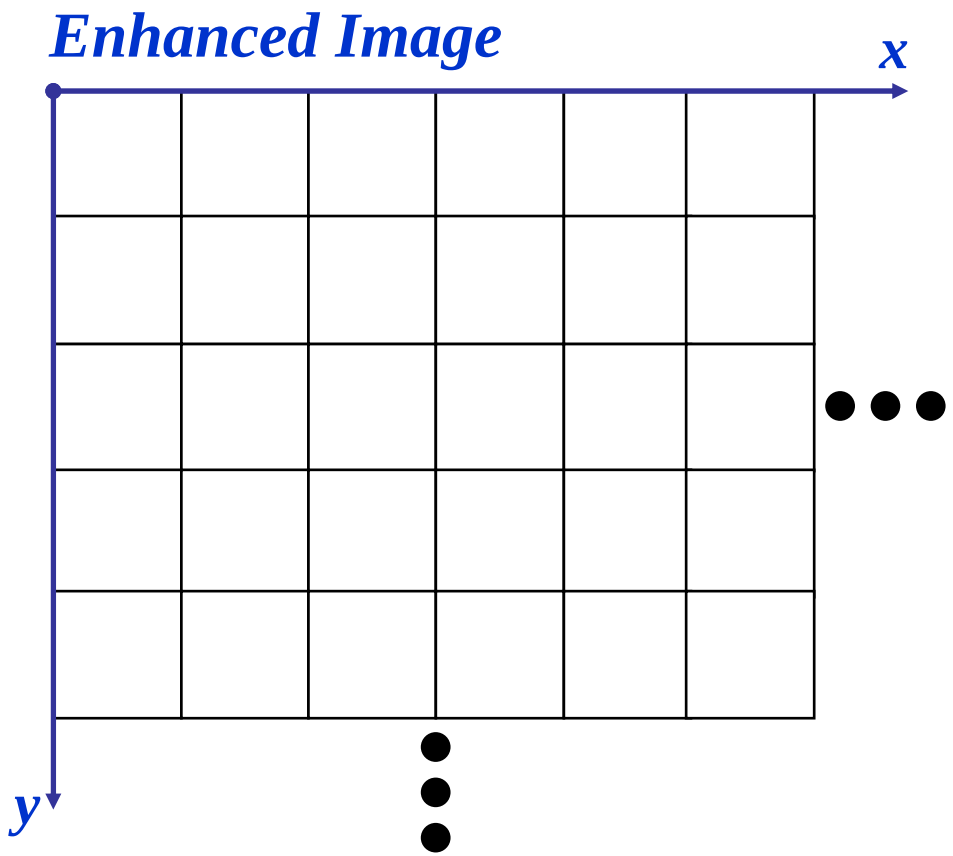
# Simple Neighbourhood Operations

Some simple neighbourhood operations include:

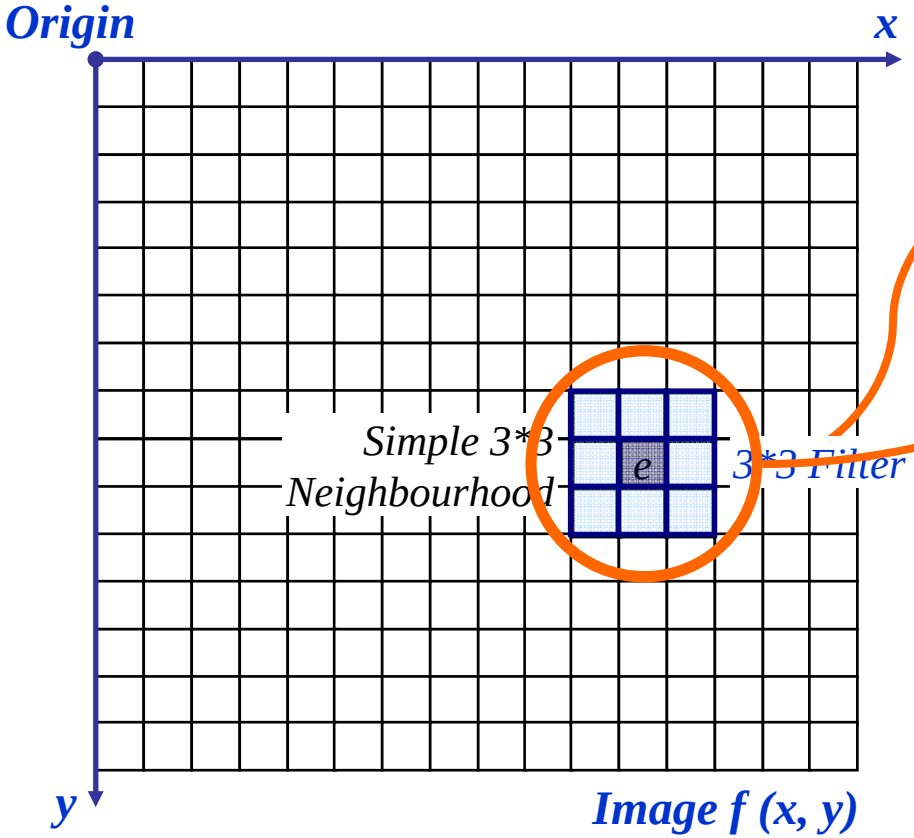
- **Min:** Set the pixel value to the minimum in the neighbourhood
- **Max:** Set the pixel value to the maximum in the neighbourhood
- **Median:** The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average

# Simple Neighbourhood Operations Example



# The Spatial Filtering Process




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

**Original Image Pixels**

*

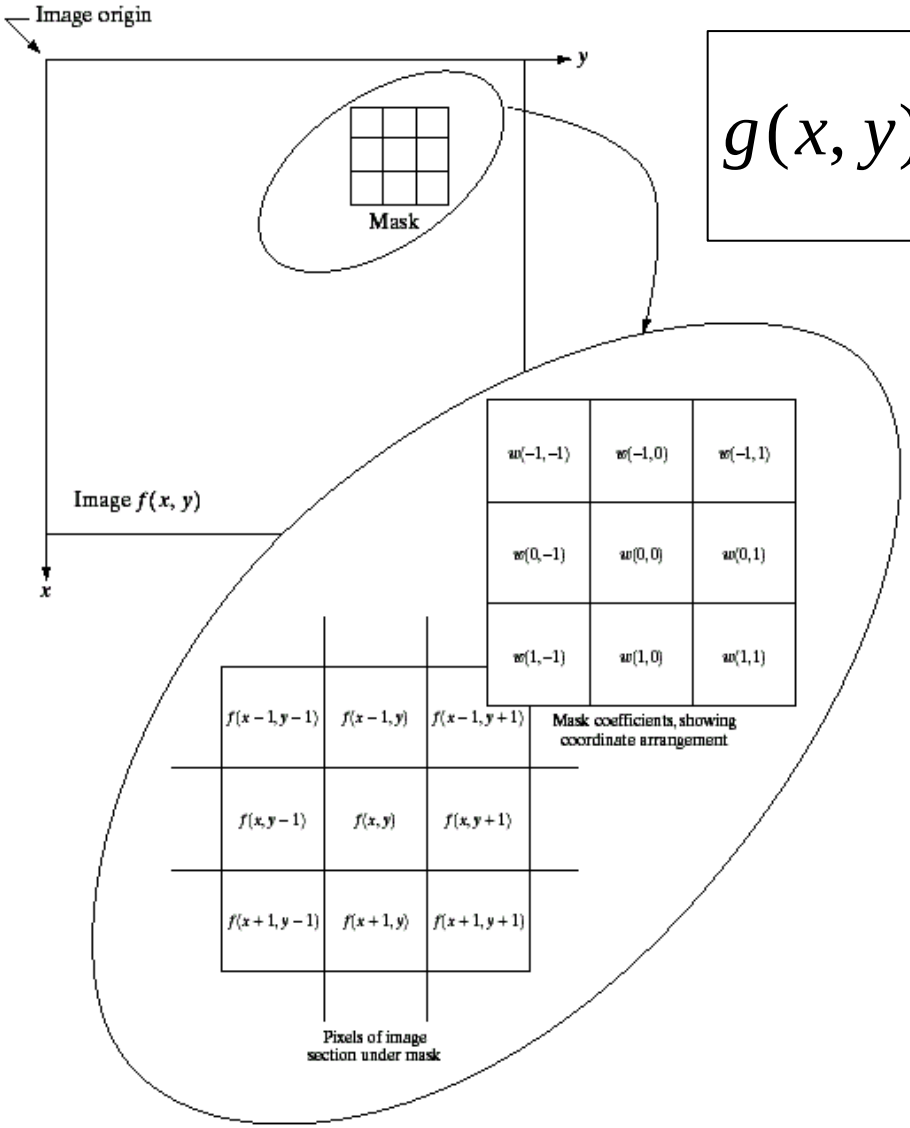
| | | |
|---|---|---|
| *r* | *s* | *t* |
| *u* | *v* | *w* |
| *x* | *y* | *z* |

**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 smoothed 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

# Smoothing Spatial Filters

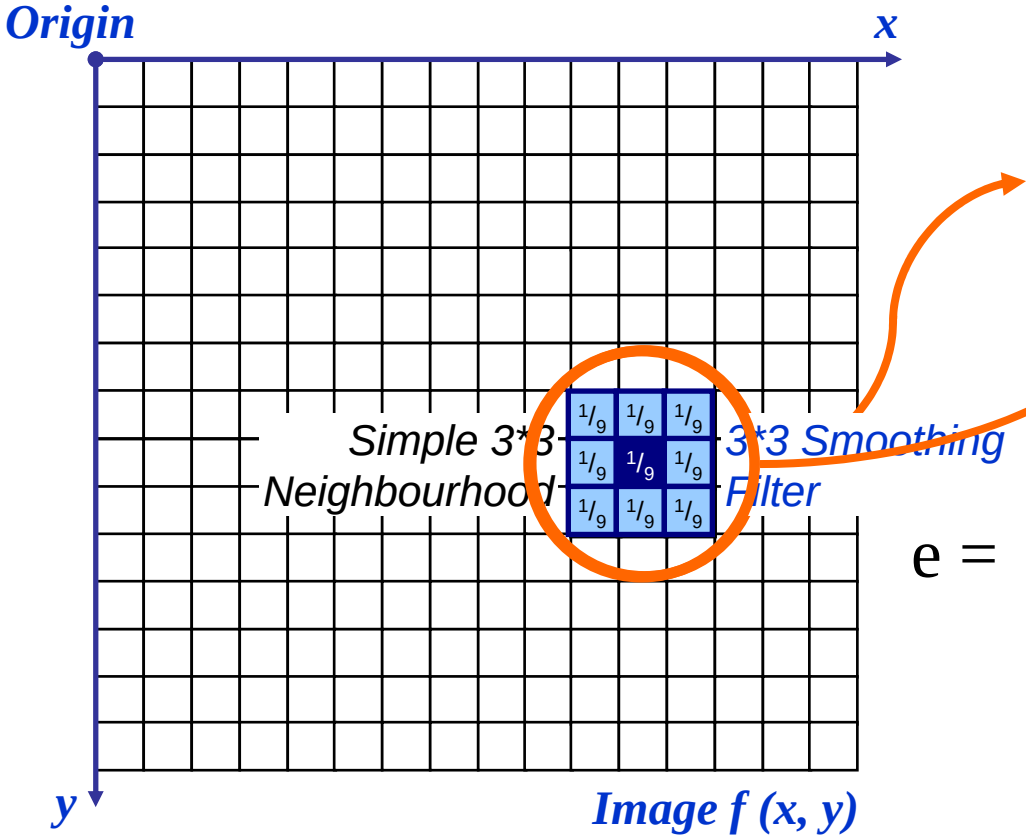
One of the simplest spatial filtering operations we can perform is a smoothing operation

- 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

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

Simple averaging filter

# 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**

$$e = \tfrac{1}{9}*106 + \tfrac{1}{9}*104 + \tfrac{1}{9}*100 + \tfrac{1}{9}*108 + \tfrac{1}{9}*99 + \tfrac{1}{9}*98 + \tfrac{1}{9}*95 + \tfrac{1}{9}*90 + \tfrac{1}{9}*85$$

$$= 98.3333$$

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

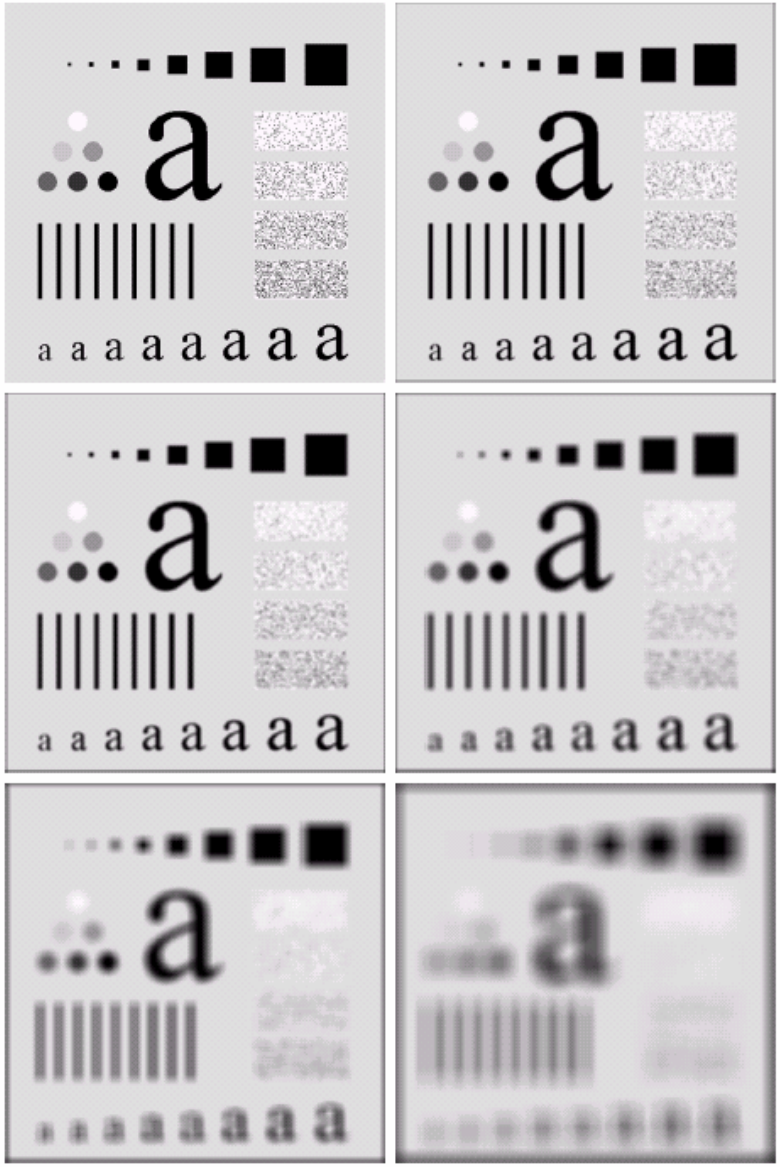
# Image Smoothing Example

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

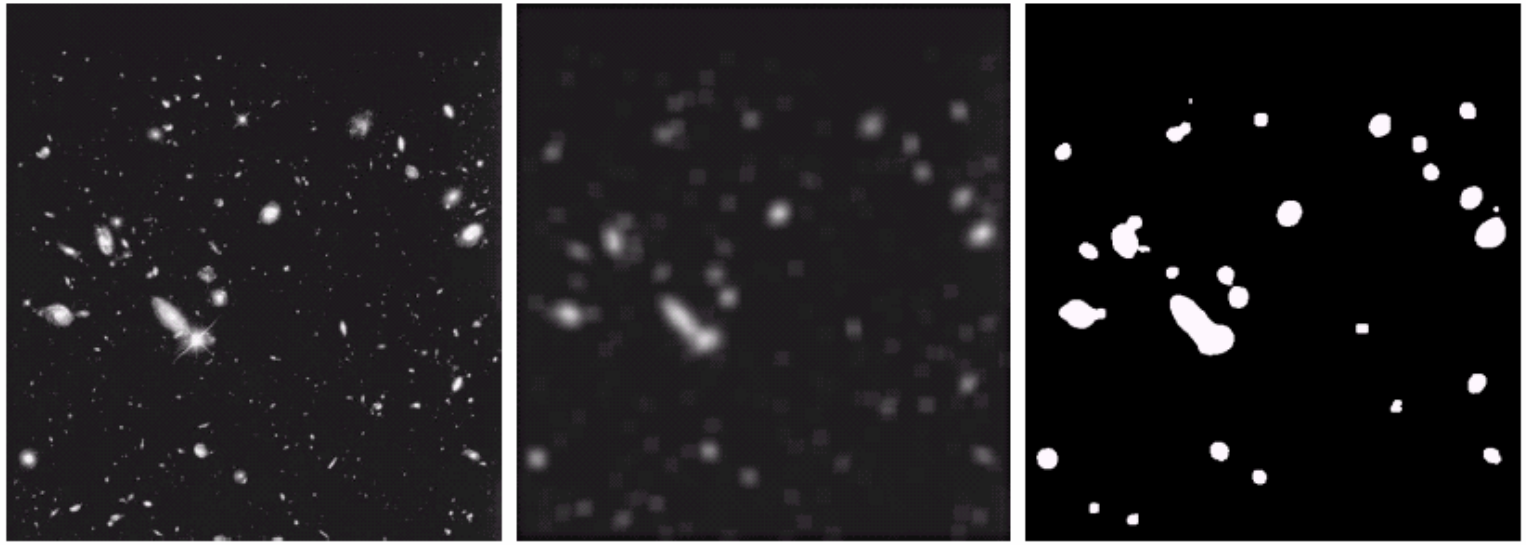
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

| | | |
|---|---|---|
| $^1/_{16}$ | $^2/_{16}$ | $^1/_{16}$ |
| $^2/_{16}$ | $^4/_{16}$ | $^2/_{16}$ |
| $^1/_{16}$ | $^2/_{16}$ | $^1/_{16}$ |

Weighted averaging filter

# Another Smoothing Example

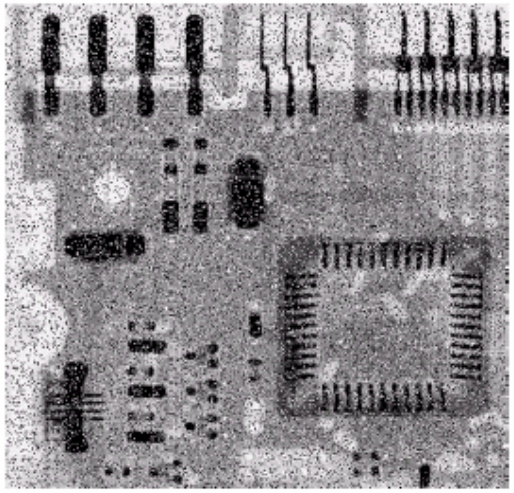
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



**Original Image** **Smoothed Image** **Thresholded Image**

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Averaging Filter Vs. Median Filter Example



**Original Image With Noise**

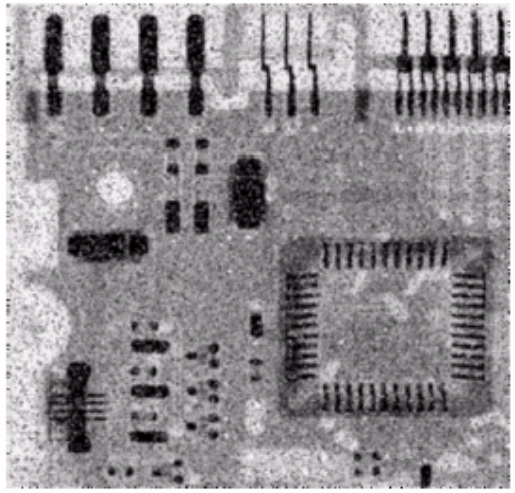


**Image After Averaging Filter**

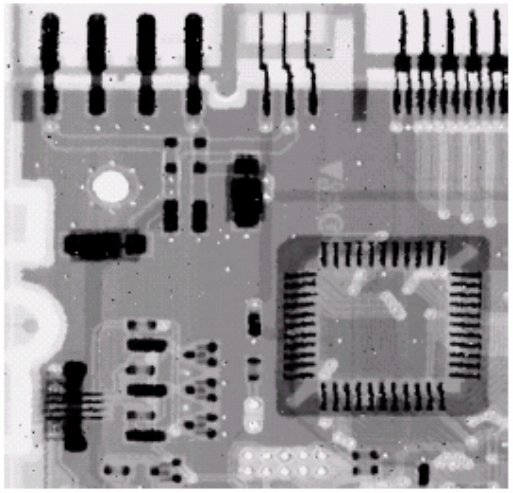


**Image After Median Filter**

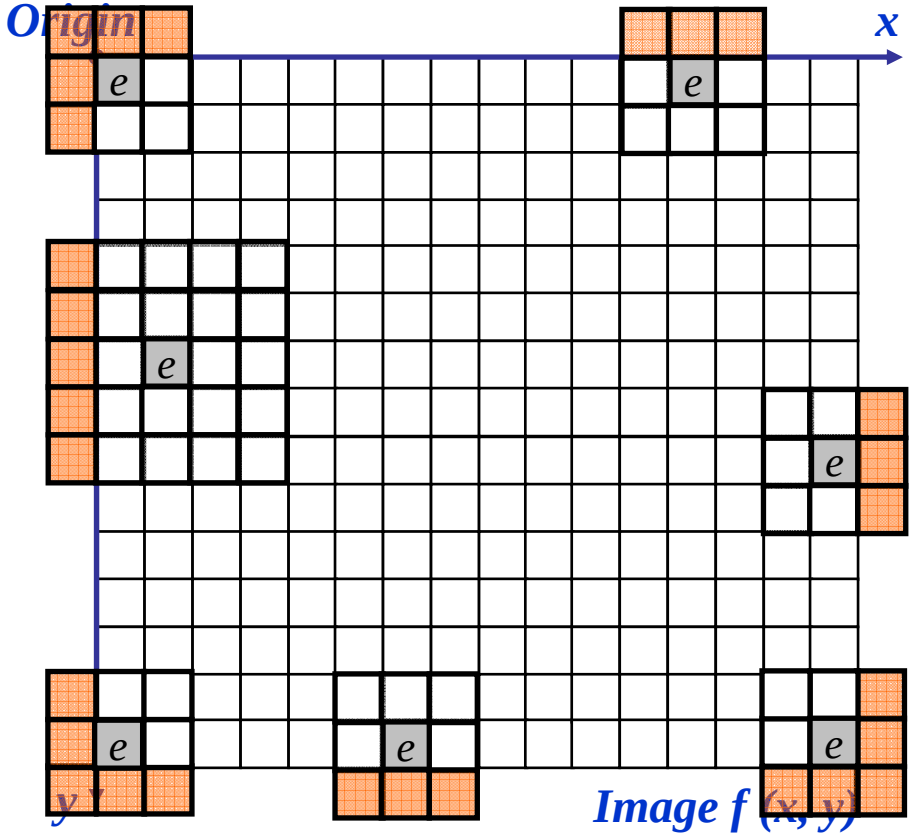
Filtering is often used to remove noise from images

Sometimes a median filter works better than an averaging filter

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Strange Things Happen At The Edges!

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

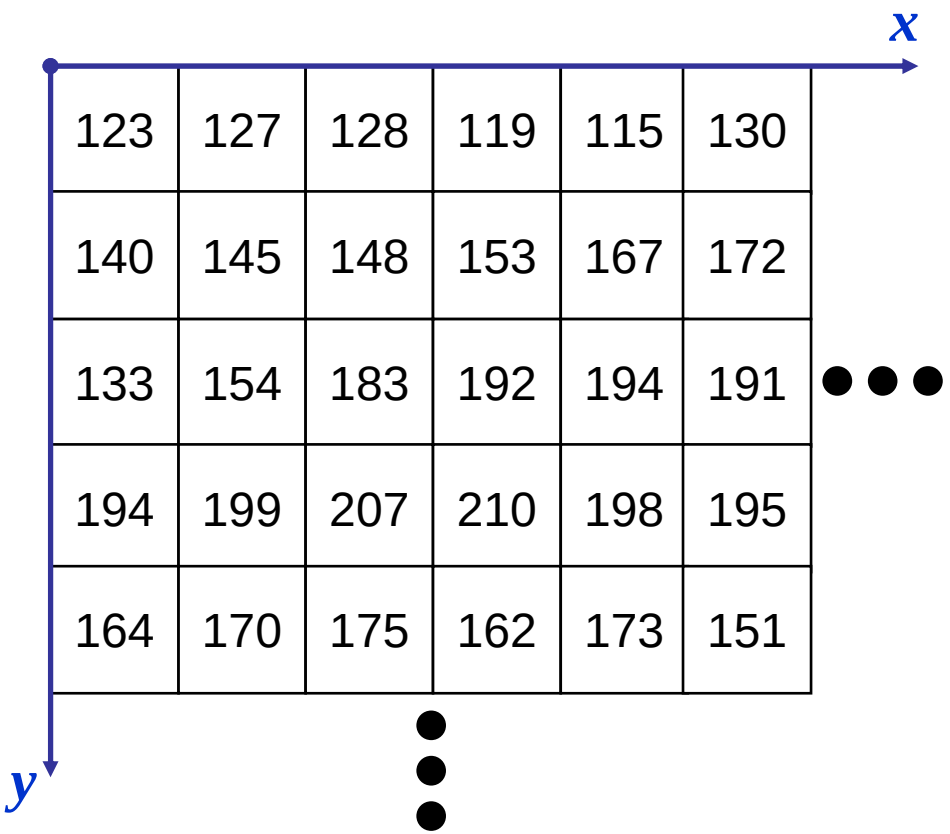


# Strange Things Happen At The Edges! (cont…)

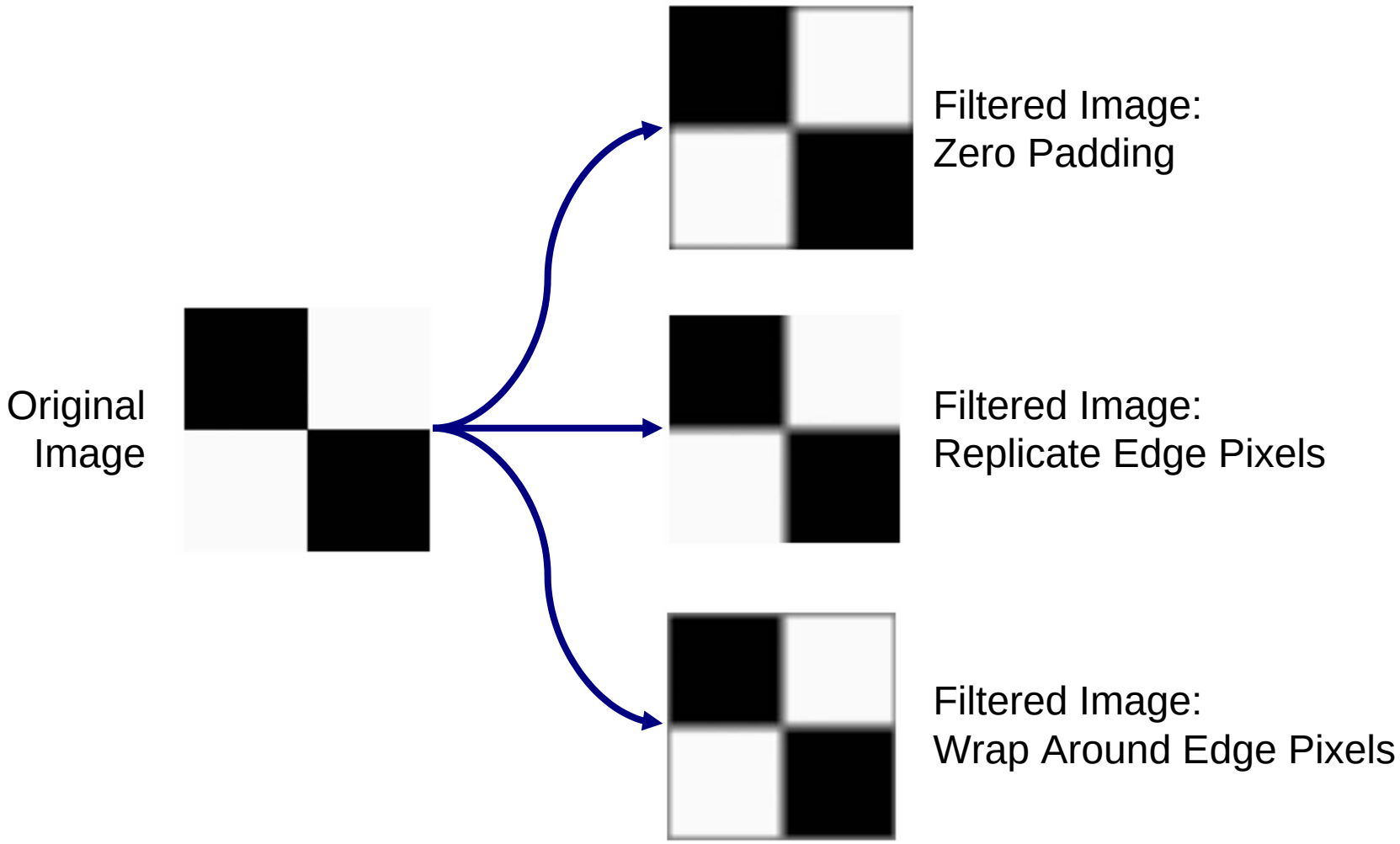
There are a few approaches to dealing with missing edge pixels:

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

# Simple Neighbourhood Operations Example



# Strange Things Happen At The Edges! (cont…)

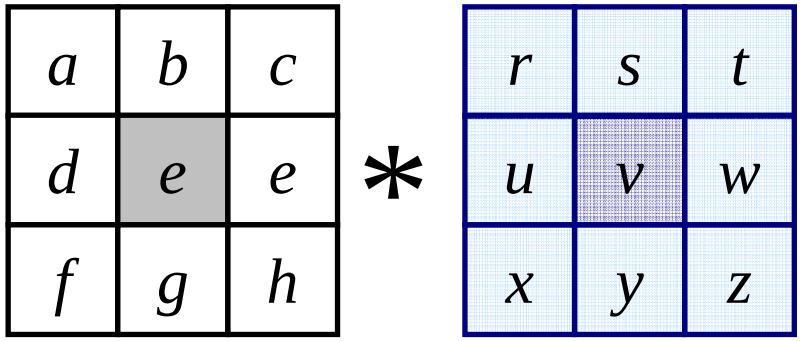



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# 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




**Original Image Pixels** **Filter**

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

For symmetric filters it makes no difference