
University of
Nottingham
UK | CHINA | MALAYSIA
COMP-2032
Introduction to
Image Processing
Lecture 9
Hough Transform & Frequency Domain

Learning Outcomes

IDENTIFY

1. How to find lines?
2. The Hough Transform
3. Other parameter spaces
4. What is frequency domain?
5. The frequency domain processing

University of Nottingham
UK | CHINA | MALAYSIA

Finding Lines:
Template Matching

Why Lines?

Geometric primitives make important vision tasks easier

E.g., matching or classifying images

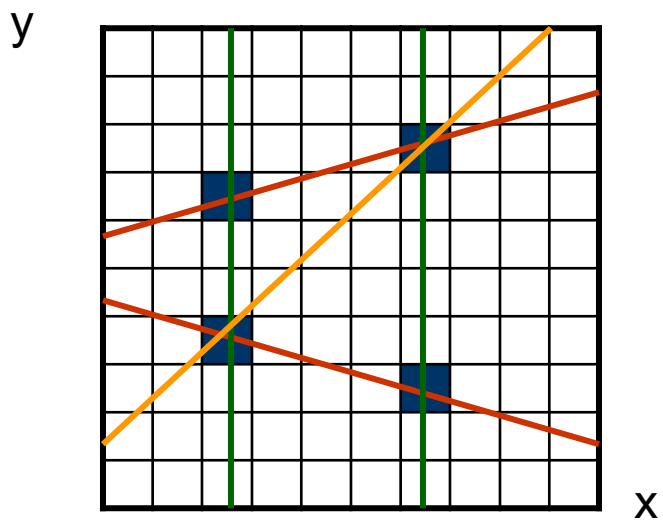
- Some geometric objects are invariant under projection from the (3D) world to (2D) image
- Straight lines in the world create straight lines on the image

A straight line in the image is evidence for a straight line in the world

ACK: Prof. Tony Pridmore, UNUK

The Problem

- Suppose straight lines are important
- Edge detection provides a set of points (x_i, y_i) which are likely to lie on those lines
- But how many lines are there and what are their parameters?



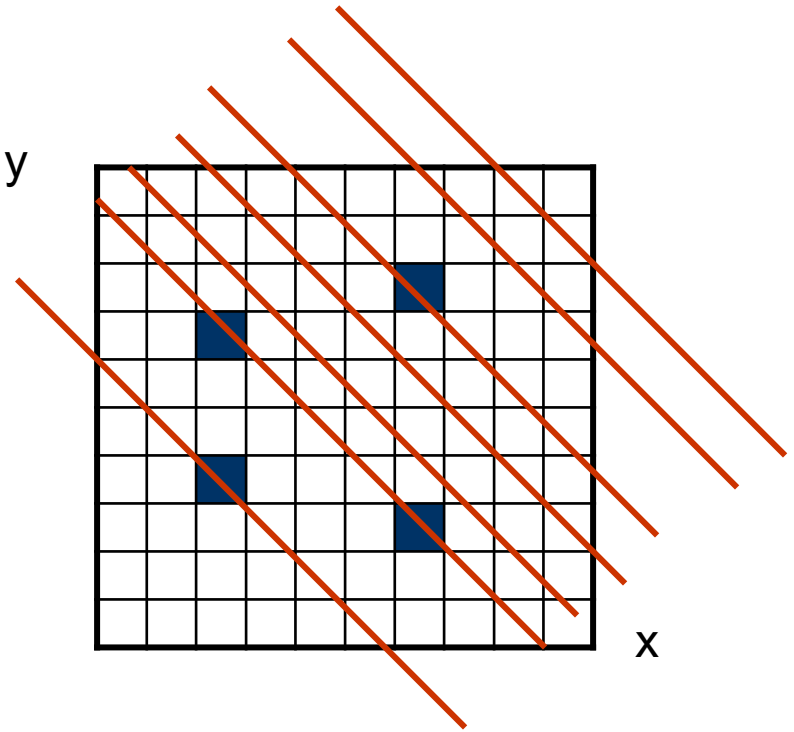
ACK: Prof. Tony Pridmore, UNUK

Template Matching

- One solution is to take a straight line and match it to all possible image positions and orientations
- Compute a measure of fit to the edge data

E.g., count how many edges are under each possible line

Incredibly expensive!



ACK: Prof. Tony Pridmore, UNUK

Template Matching vs. Hough

Classic template matching takes a line, lies it on the image data and asks:

- *Does it fit here?*
- *Here?*
- *Here?*
- *How about here?*
- *Here then?*
- ……..

The Hough takes each data item (edge) and asks:

What lines could pass through this?

ACK: Prof. Tony Pridmore, UNUK

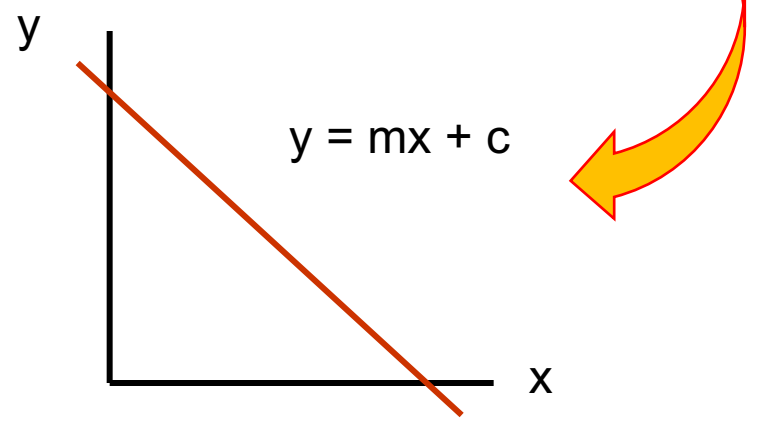
University of Nottingham
UK | CHINA | MALAYSIA

The Hough Transform

Line Parameters

The standard equation of a line is

$$y = mx + c$$



- If we know the line parameters *m, c* - we can vary *x*, compute *y* and draw the line
- This line represents the set of (*x,y*) pairs that satisfy the above equation

But we don't know the line parameters – we just have some data points (x_i, y_i)

ACK: Prof. Tony Pridmore, UNUK

Parameter Space

But if a data point (x_i, y_i) lies on a line then that line's parameters m, c must satisfy

$$y_i = mx_i + c$$

So we can represent all possible lines through (x_i, y_i) by the set of (m,c) pairs that satisfy this equation

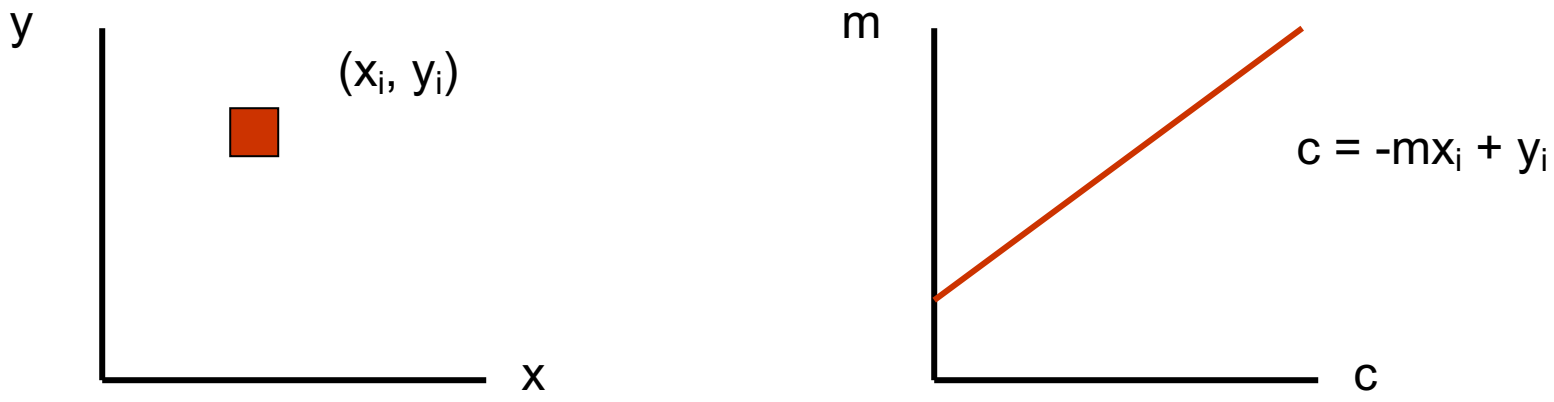
Rearranging the equation gives

$$C = -mx_i + y_i$$

This also describes a straight line, but as x_i and y_i are known and m and c are unknown, that line is in m, c space – a parameter space

ACK: Prof. Tony Pridmore, UNUK

Parameter Space

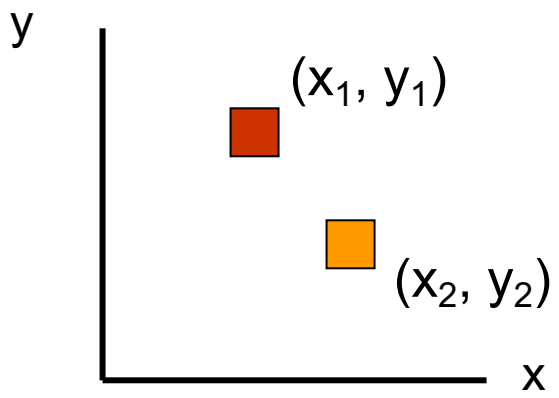


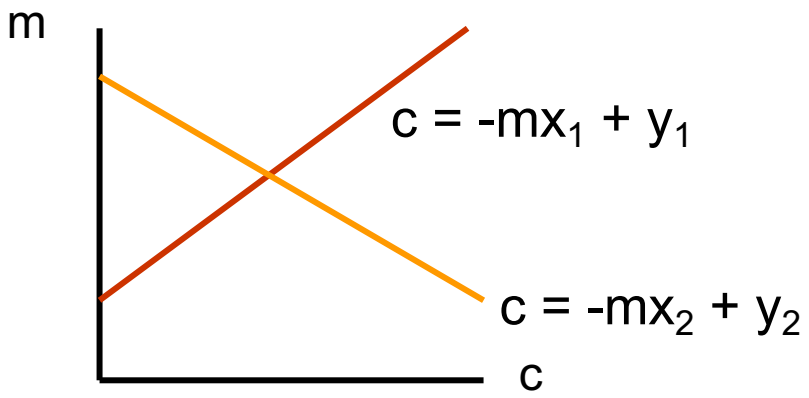
This straight line in *m*, *c* space represents all the values of (*m, c*) that satisfy $y_i = mx_i + c$, and so the parameters of all the lines that could pass through (x_i, y_i)

ACK: Prof. Tony Pridmore, UNUK

Parameter Space

Suppose there are 2 edges in our image (x_1, y_1) and (x_2, y_2)



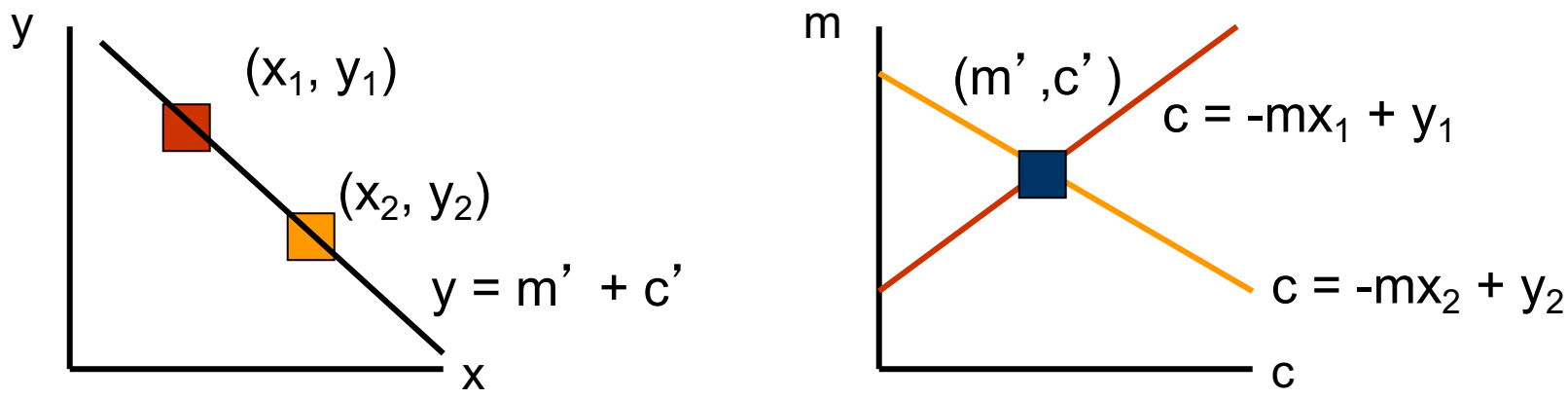


They will each generate a line in (m,c) space representing the set of lines that could pass through them

ACK: Prof. Tony Pridmore, UNUK

Parameter Space

The intersection of our 2 lines is the intersection of 2 sets of line parameters



The point of intersection therefore gives the parameters of a line through both (x_1, y_1) and (x_2, y_2)

ACK: Prof. Tony Pridmore, UNUK

Parameter Space

Taking this one step further

- All pixels which lie on a line is (x,y) space are represented by lines which all pass through a single point in (m,c) space
- The point through which they all pass gives the values of m' and c' in the equation of the line: $y = m'x + c'$

To detect lines all we need to do is transform each edge point into m,c space and look for places where lots of lines intersect

This is what Hough Transform do

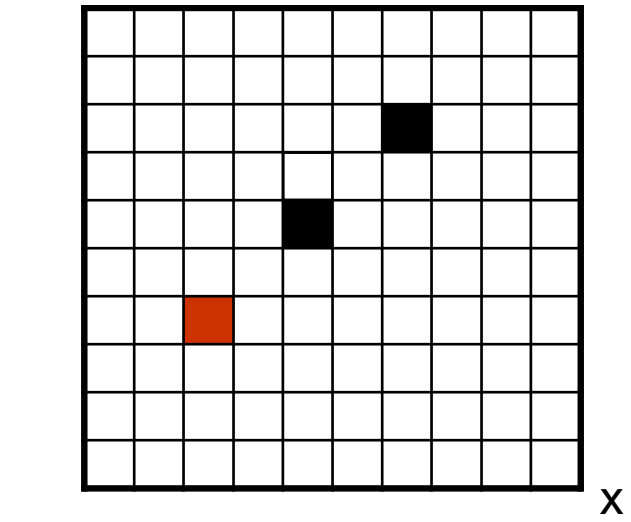
ACK: Prof. Tony Pridmore, UNUK

The Hough Transform

1. Quantise (m,c) space into a two-dimensional array A for appropriate steps of m and c
2. Initialise all elements of $A(m,c)$ *to zero*
3. For each pixel (x_i, y_i) which lies on some edge in the image, we add 1 to all elements of $A(m, c)$ whose indices m and c satisfy $y_i = mx_i + c$
4. Search for elements of $A(m, c)$ which have large values – each one found corresponds to a line in the original image

ACK: Prof. Tony Pridmore, UNUK

The Hough Transform



m

m'

c

c'

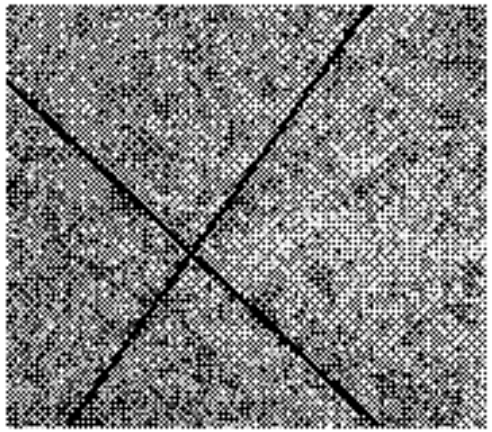
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	1	0	1	0	0
1	1	1	1	1	1	3	1	1	1
0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0

There is one line: $y = m'x + c'$

ACK: Prof. Tony Pridmore, UNUK

A (More) Real Example

Image

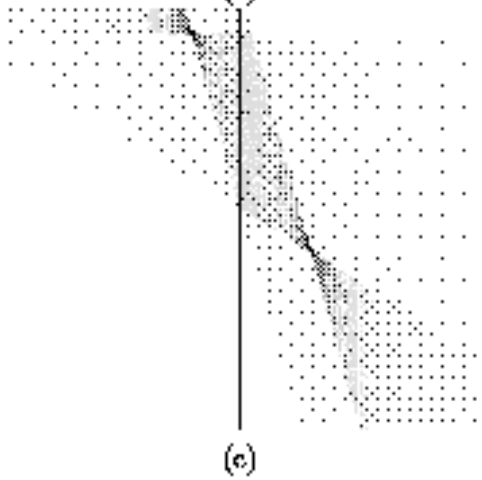


(a)

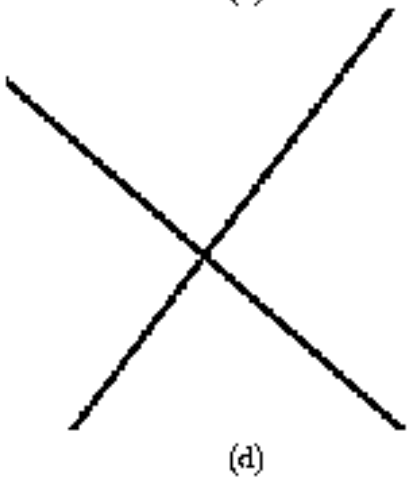
(b)

Edges

Accumulators



(c)



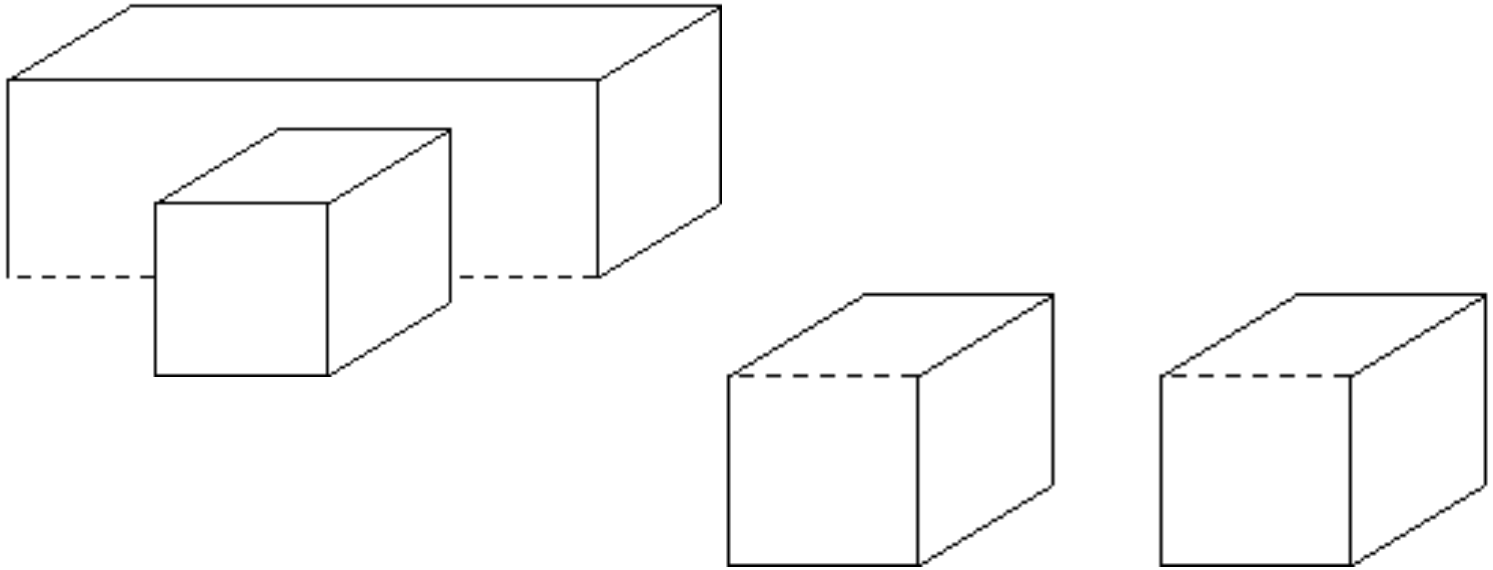
(d)

Result

ACK: Prof. Tony Pridmore, UNUK

Line Parameters

The Hough Transform only supplies the parameters of the lines it detects; this can be an advantage or a disadvantage



ACK: Prof. Tony Pridmore, UNUK

University of Nottingham
UK | CHINA | MALAYSIA

Other Parameter Spaces

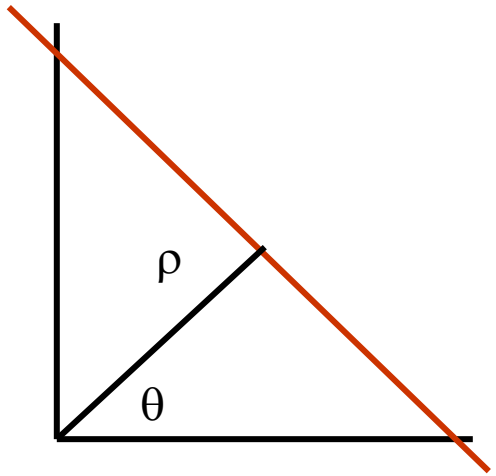
Straight Lines AGAIN

Another equation for a straight line

$$\rho = x.\cos(\theta) + y.\sin(\theta)$$

For an n x n image

- ρ is in range [0, 2n]
- θ is in range [0, 2π]



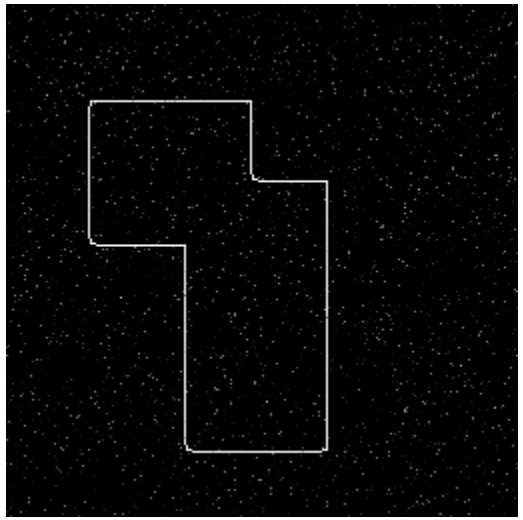
Each x_i, y_i generates a sinusoid in ρ, θ space

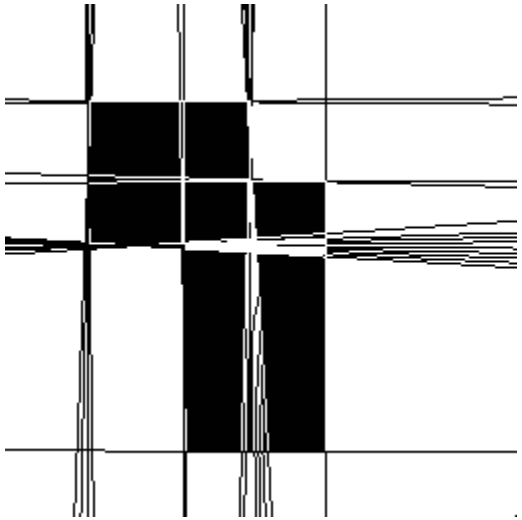
ACK: Prof. Tony Pridmore, UNUK

A ρ, θ Example

ACK: Prof. Tony Pridmore, UNUK

A ρ, θ Example

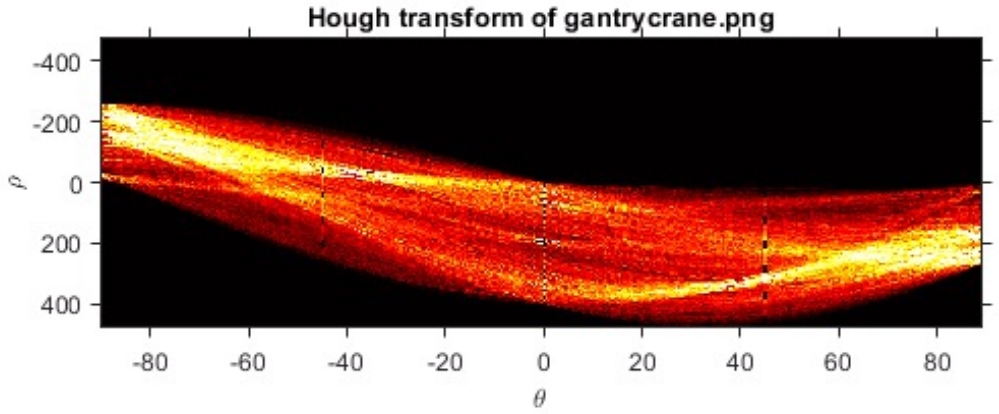




A key question is what happens to peaks in parameter space when noise or occlusion arise?

ACK: Prof. Tony Pridmore, UNUK

The Hough Transform in Matlab



Like many implementations, Matlab's Hough allows you to specify which part of the parameter space you're interested in…

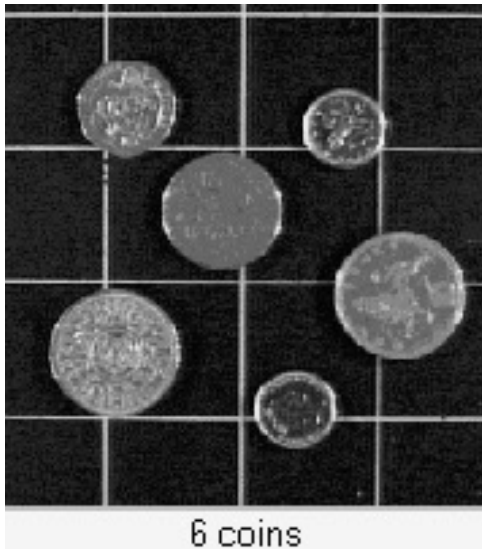
https://uk.mathworks.com/help/images/ref/hough.html

ACK: Prof. Tony Pridmore, UNUK

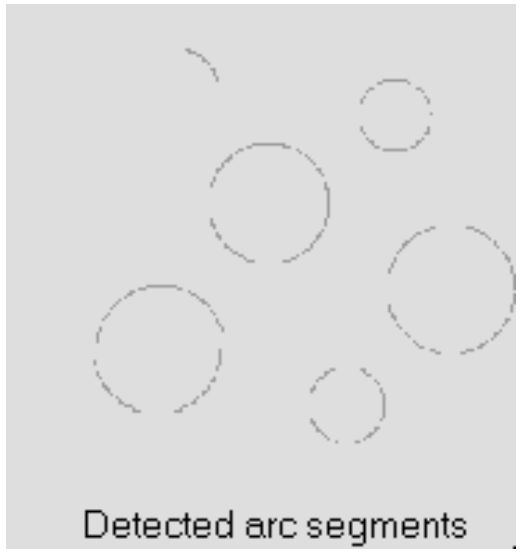
Circles

Parametric equation is $r^2 = (x-a)^2 + (y-b)^2$

- Full problem ⟶ cones in (*a, b, r*) space
- Known r ⟶ circles in (*a, b*) space



6 coins



Detected arc segments

How about ellipses?

The Hough can in principle be applied to any parameterizable curve, though its not always practical

ACK: Prof. Tony Pridmore, UNUK

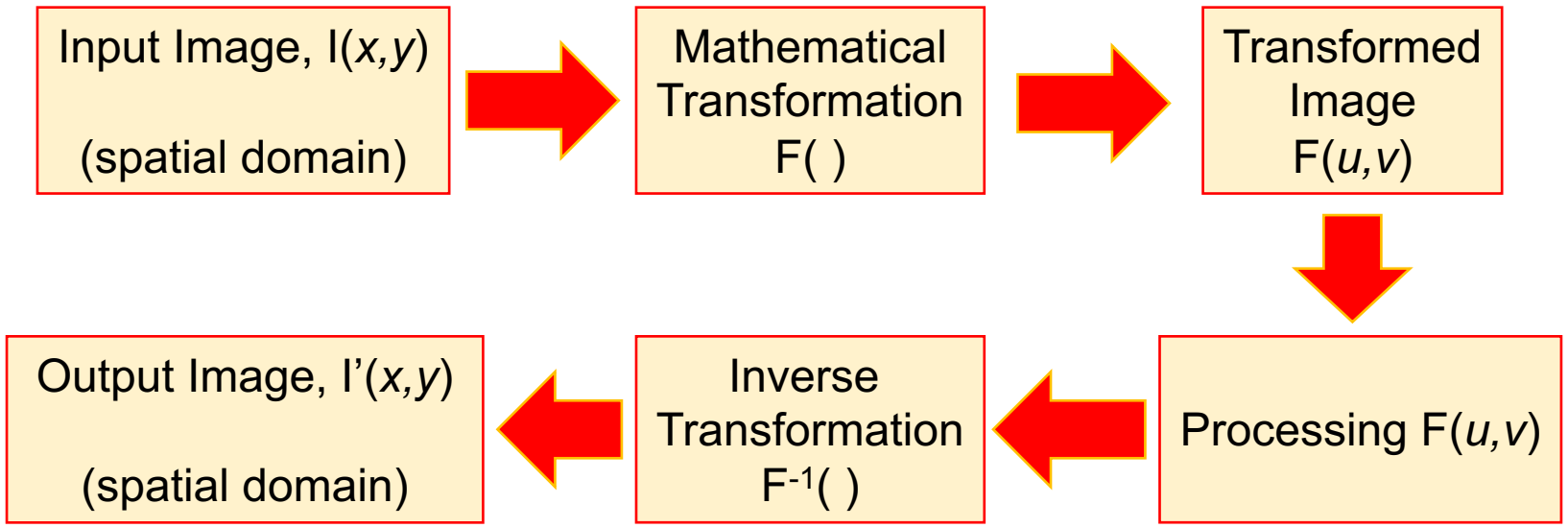
Break

University of Nottingham
UK | CHINA | MALAYSIA

Frequency Domain

An Alternative Representation

Image Transform: The Idea



ACK: Prof. Tony Pridmore, UNUK

The Fourier Transform

- There are many different transformations, the Fourier Transform (FT) or its fast implementation (FFT) is the best known
- COMP 2032 will treat FFT as a black box and will **NOT** consider the underlying mathematics in detail
- Focus will be on the key idea, the frequency space representation, and how it can be used

Any given function (*an image is a 2D function*) can be approximated by a weighted sum of ***basis functions***, e.g., sines and cosines

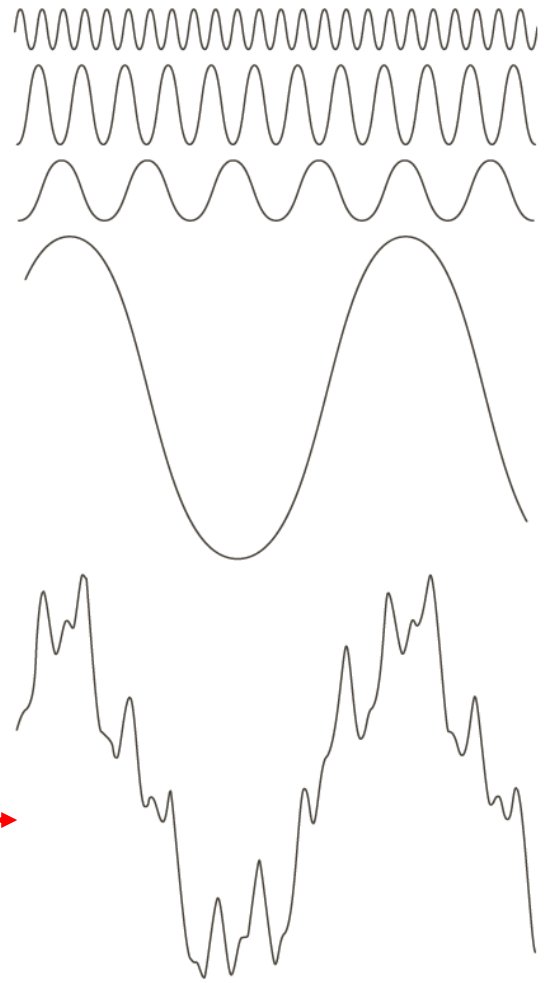
ACK: Prof. Tony Pridmore, UNUK

The Fourier Transform

The basic idea (Fourier 1807)

These 4 functions

… can be combined to form this one



ACK: Prof. Tony Pridmore, UNUK

Image Transforms

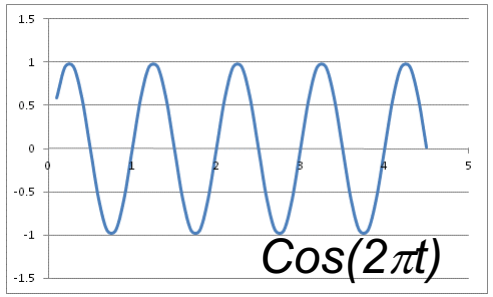
Suppose we just use cosines:

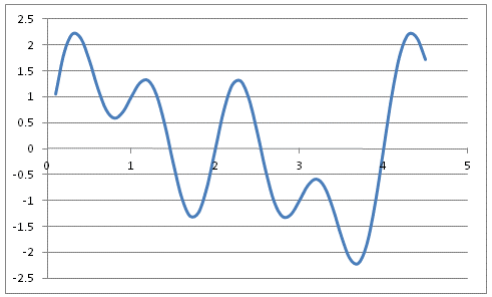
$$f(t) = \sum_i F_i \cos(u_i t)$$

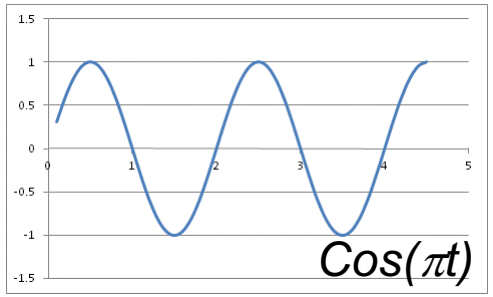
- The weight, F_i, indicates the importance of $cos(u_i t)$
- u_i represents the frequency of the cosine signal
- A larger u_i means $cos(u_i t)$ changes faster → higher frequency component of $f(t)$
- A smaller u_i means $cos(u_i t)$ changes slower → lower frequency component of $f(t)$

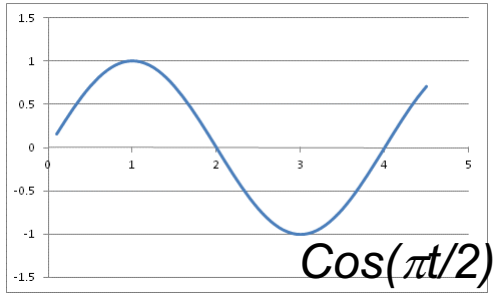
ACK: Prof. Tony Pridmore, UNUK

Image Transforms







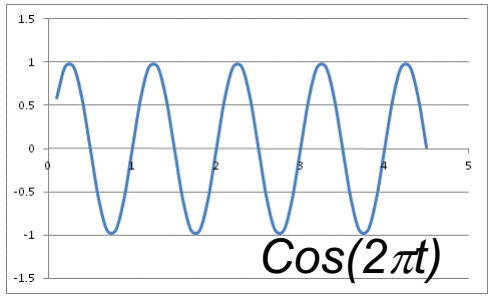


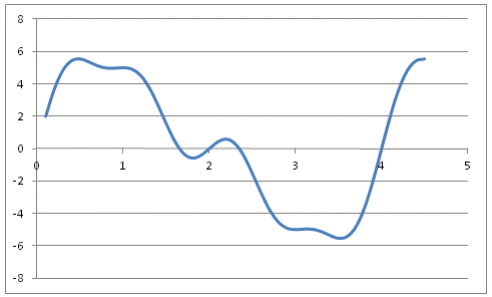
$f(t) = Cos(\pi t/2)+Cos(\pi t)+Cos(2\pi t)$

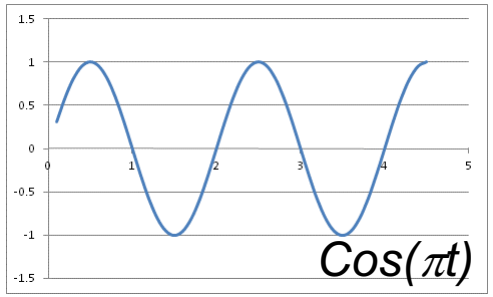
$F_1 = 1 \quad u_1 = \pi/2$
$F_2 = 1 \quad u_2 = \pi$
$F_3 = 1 \quad u_3 = 2\pi$

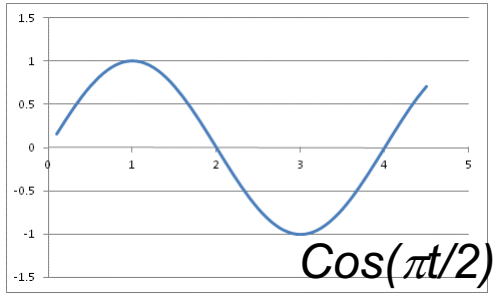
ACK: Prof. Tony Pridmore, UNUK

Image Transforms









$f(t) = 5*Cos(\pi t/2)+2*Cos(\pi t)+Cos(2\pi t)$

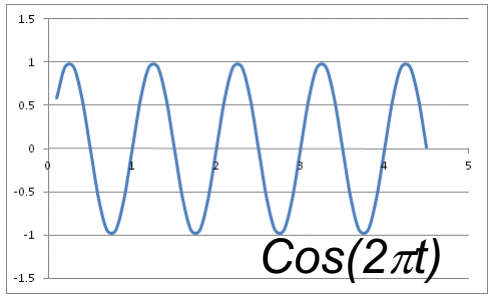
$F_1 = 5 \quad u_1 = \pi/2$

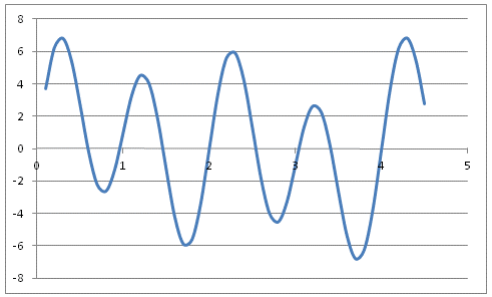
$F_2 = 2 \quad u_2 = \pi$

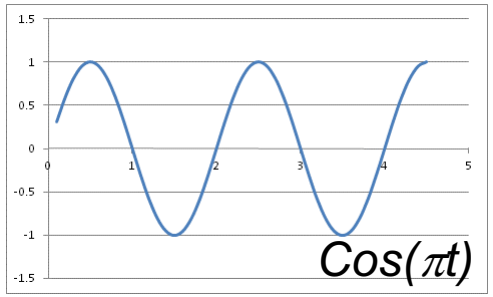
$F_3 = 1 \quad u_3 = 2\pi$

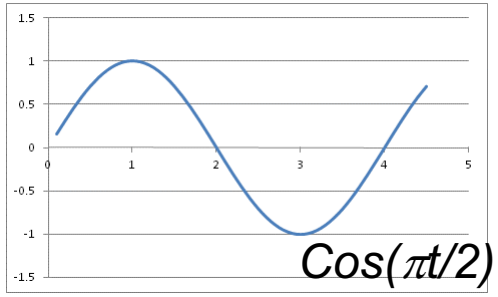
ACK: Prof. Tony Pridmore, UNUK

Image Transforms







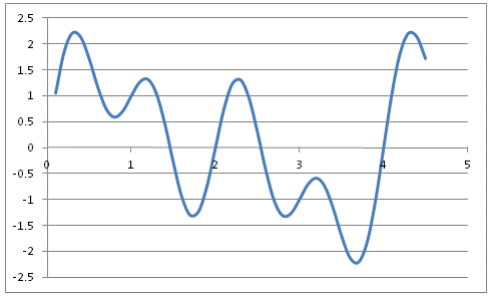


$f(t) = Cos(\pi t/2)+2*Cos(\pi t)+5*Cos(2\pi t)$

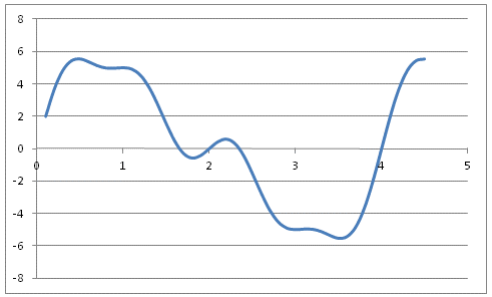
$F_1 = 1 \quad u_1 = \pi/2$
$F_2 = 2 \quad u_2 = \pi$
$F_3 = 5 \quad u_3 = 2\pi$

ACK: Prof. Tony Pridmore, UNUK

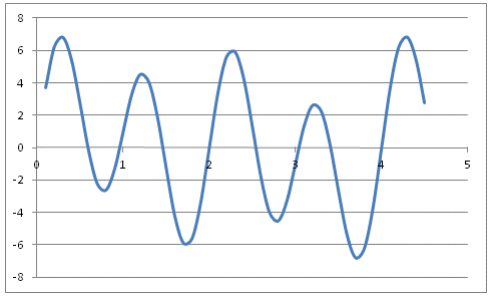
Image Transforms



$F_1 = 1 \quad u_1 = \pi/2$
$F_2 = 1 \quad u_2 = \pi$
$F_3 = 1 \quad u_3 = 2\pi$

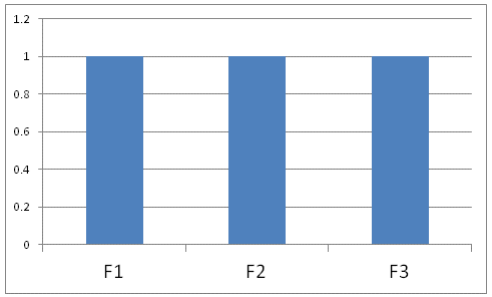


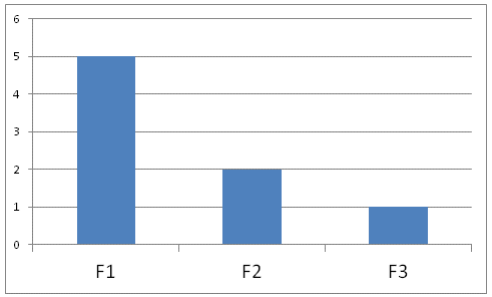
$F_1 = 5 \quad u_1 = \pi/2$
$F_2 = 2 \quad u_2 = \pi$
$F_3 = 1 \quad u_3 = 2\pi$

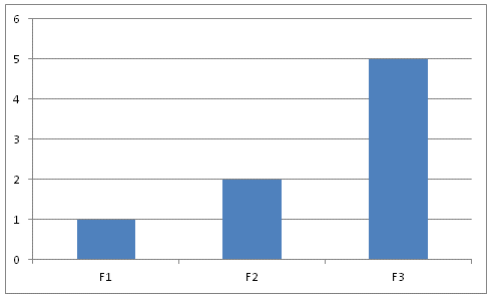


$F_1 = 1 \quad u_1 = \pi/2$
$F_2 = 2 \quad u_2 = \pi$
$F_3 = 5 \quad u_3 = 2\pi$

$$f(t) = F_1 * Cos(\pi t/2) + F_3 * Cos(\pi t) + F_3 * Cos(2\pi t)$$



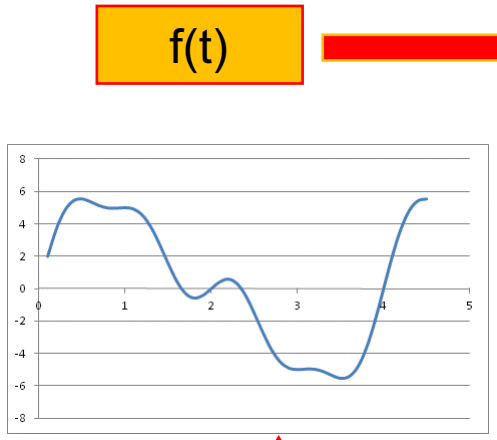


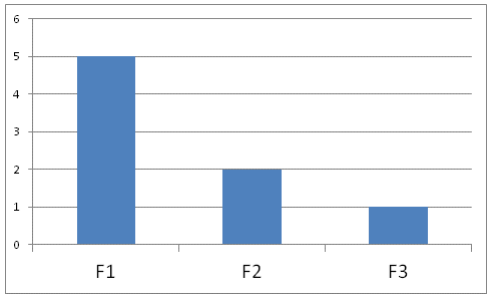


ACK: Prof. Tony Pridmore, UNUK

The Fourier Transform

Simplified, informal version:





f(t) → FFT → F(u)

$$f(t) = \sum_i F_i \cos(u_i t)$$

Input signal (image)

basis functions: actually sines and cosines

Power spectrum or frequency distribution

ACK: Prof. Tony Pridmore, UNUK

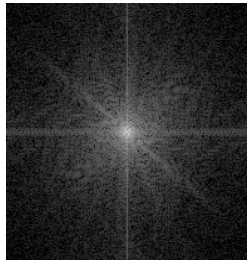
Amplitude vs Phase

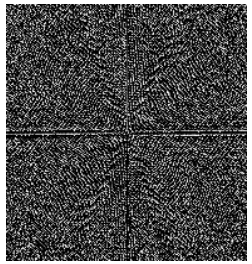
A = “Aron”

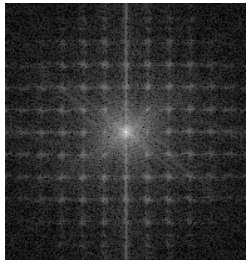
FA = fft2(A)

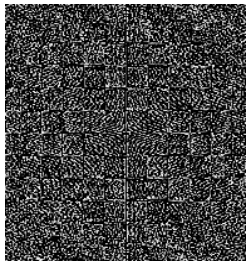
log(abs(FA))

angle(FA)









P = “Phyllis”

FA = fft2(P)

log(abs(FP))

angle(FP)

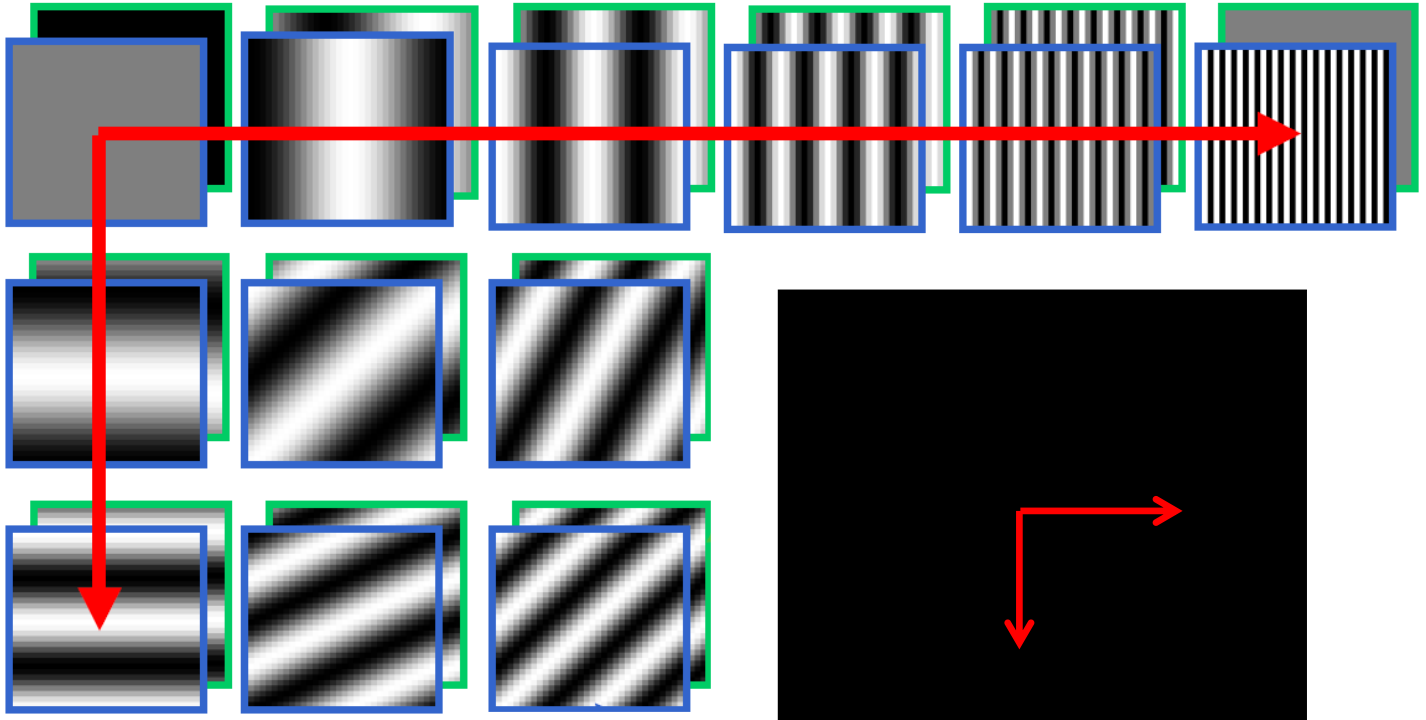
Image processing focuses on the power spectrum, the amplitude

To accurately represent a signal (image), basis functions need to be shifted (and rotated)

- **Amplitude**: relative prominence of basis function
- **Phase**: relative displacement of basis function

ACK: Prof. Tony Pridmore, UNUK

The Power Spectrum in 2D

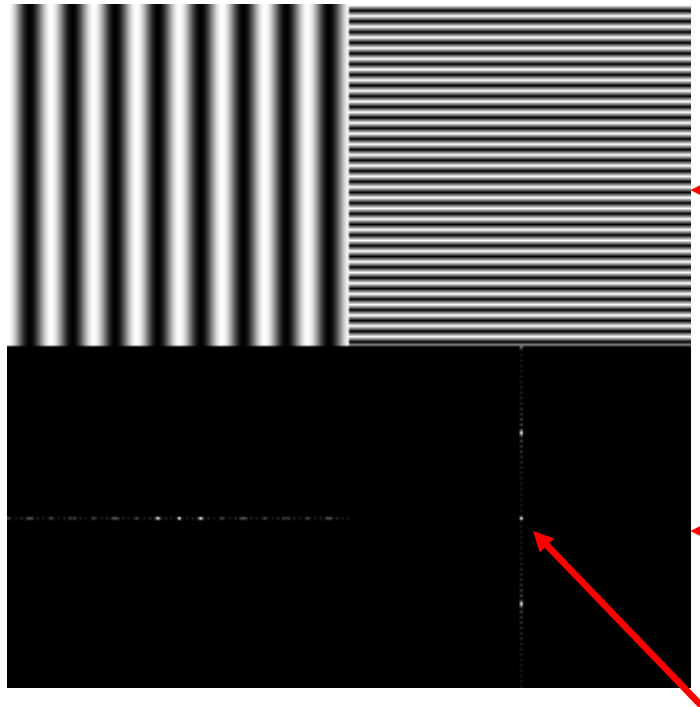


The principle is the same, but the *basis functions* are oriented

- Distance from centre reflects frequency of corresponding basis function
- Direction from centre reflects orientation of corresponding basis function
- Origin records lowest possible frequency – flat, proportional to mean intensity

ACK: Prof. Tony Pridmore, UNUK

The Power Spectrum in 2D



Two images

Their power spectra

Origin at centre

Two responses because basis functions are even: f(-x) = f(x)

ACK: Prof. Tony Pridmore, UNUK

Frequency Domain

Translation does not change the spectrum

Rotation does

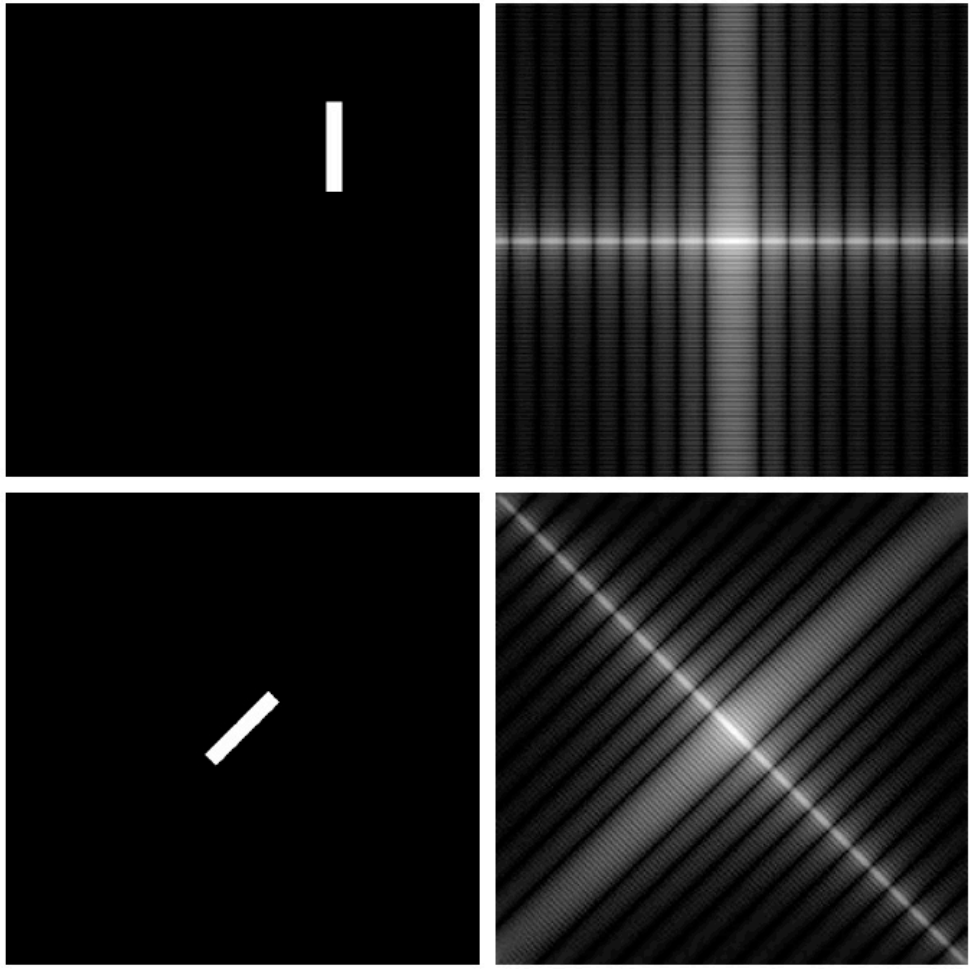


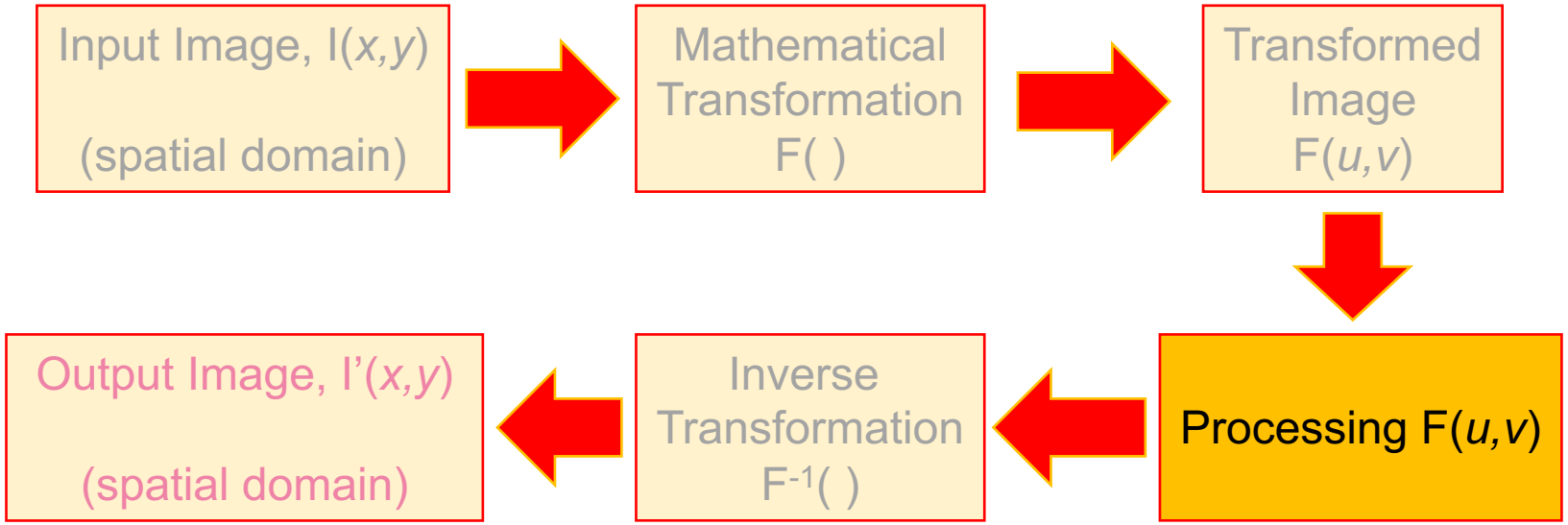
Image | Spectrum

ACK: Prof. Tony Pridmore, UNUK

University of Nottingham
UK | CHINA | MALAYSIA

Frequency Domain Processing

Frequency Domain Processing



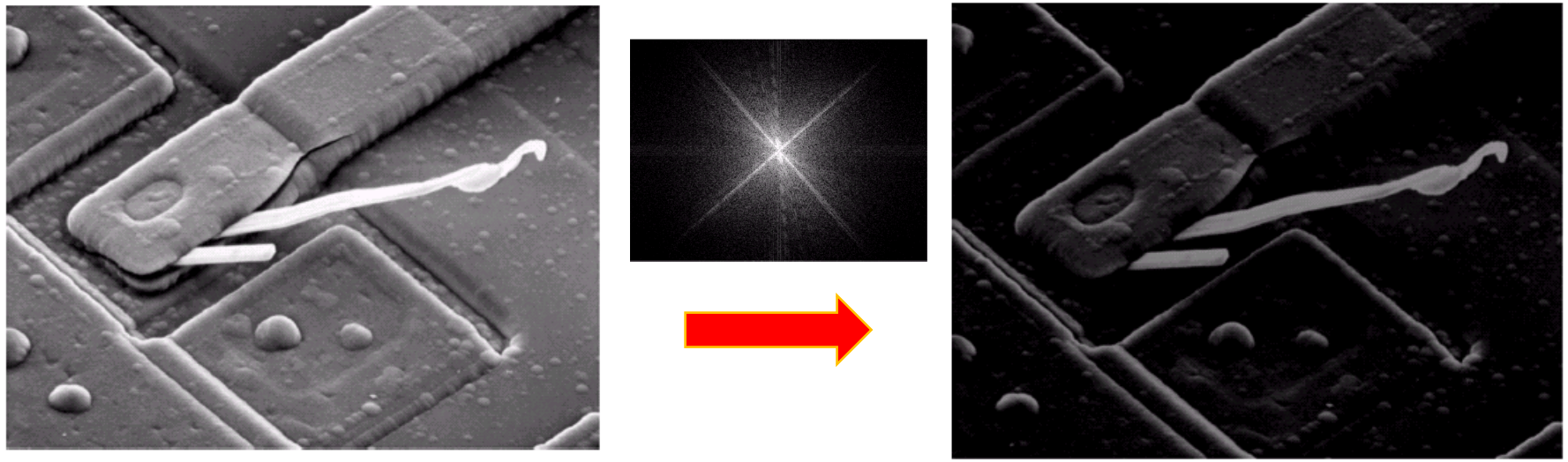
- Smoothing = low pass filtering = attenuate high frequency components
- Sharpening = high pass filtering = attenuate low frequency components
- Band limiting = set all components to 0 outside a given frequency range

ACK: Prof. Tony Pridmore, UNUK

The Simplest Example

Set the origin of the power spectrum to zero, leave the rest unchanged

- Reduces average intensity of image to 0
- Negative intensities have been set to 0 in the display



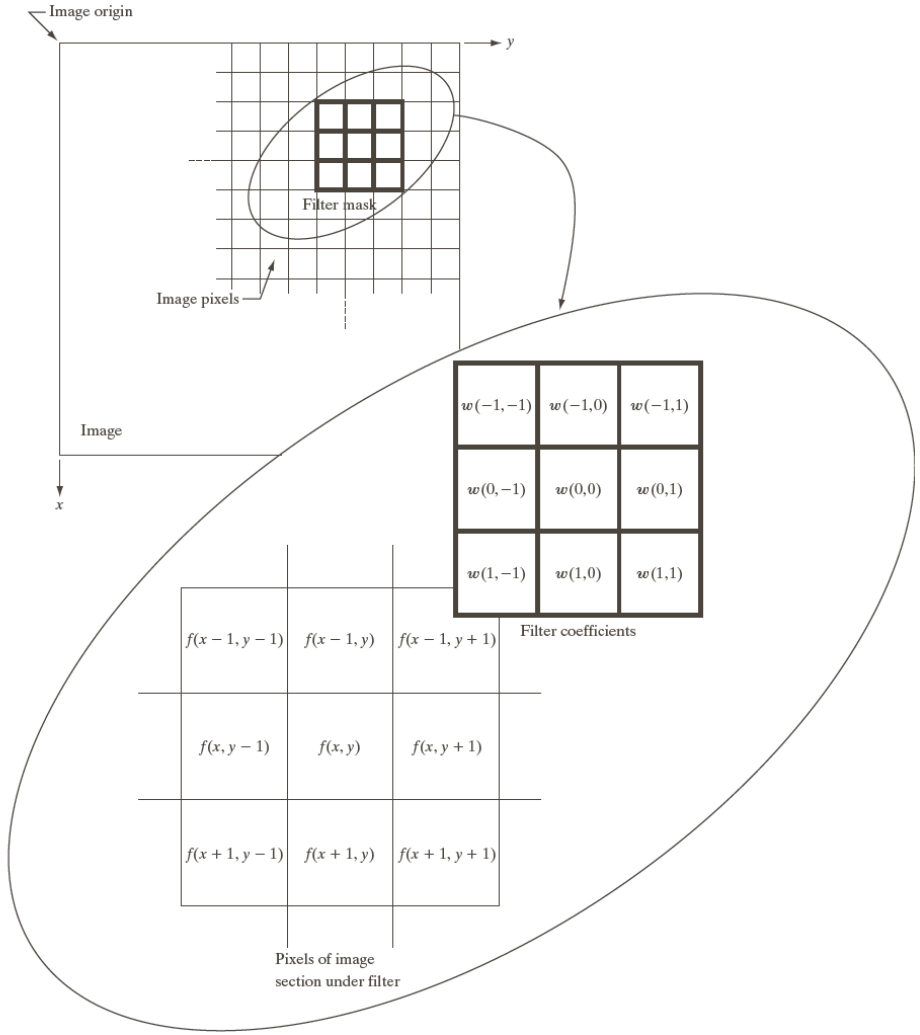
ACK: Prof. Tony Pridmore, UNUK

Convolution

- In spatial domain, processing is achieved via convolution (*)
- The *convolution theorem* underpins frequency domain processing

$$f(x,y) * g(x,y) \Leftrightarrow F(u,v)G(u,v)$$

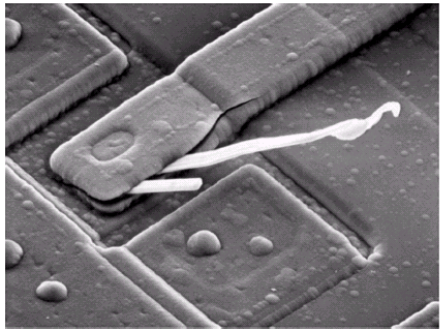
Convolution in spatial domain becomes multiplication in frequency domain



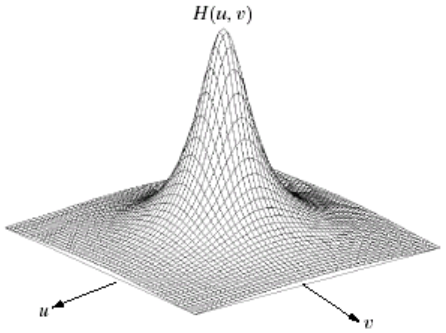
ACK: Prof. Tony Pridmore, UNUK

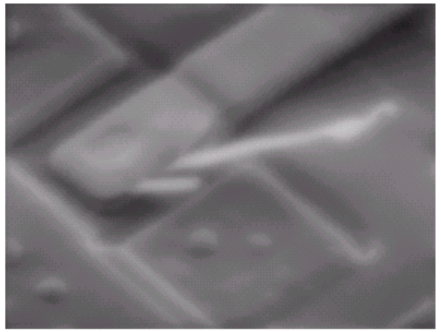
Filtering in Frequency Domain

Filtering = Multiply power spectrum by a mask designed to change the weights of selected basis functions

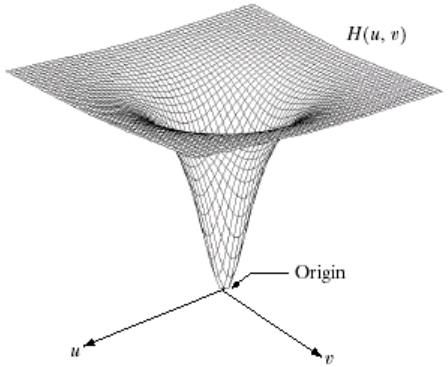


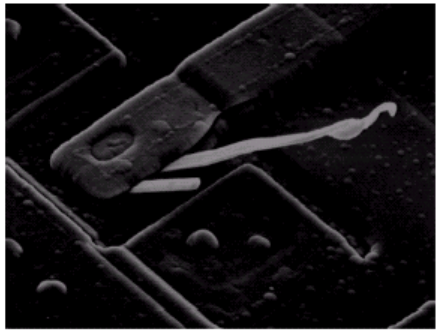
Low-pass filtering





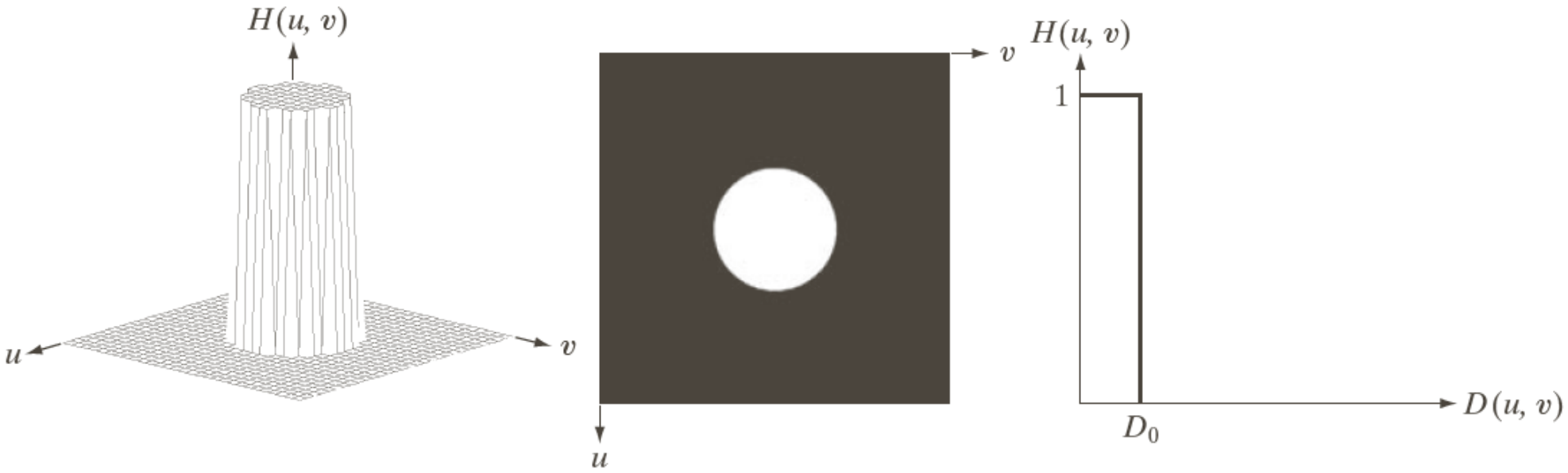
High-pass filtering





ACK: Prof. Tony Pridmore, UNUK

Ideal Low-pass Filtering



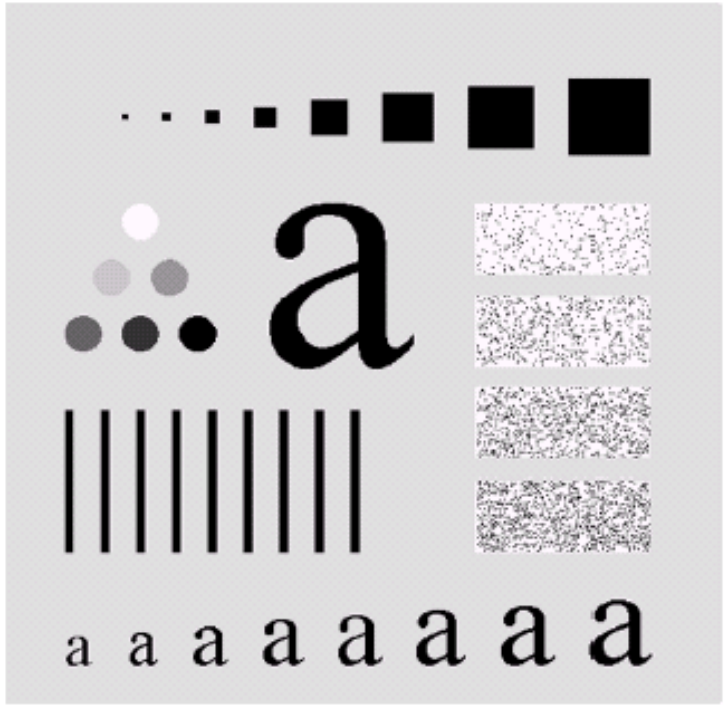
Perspective plot | Image view | Profile

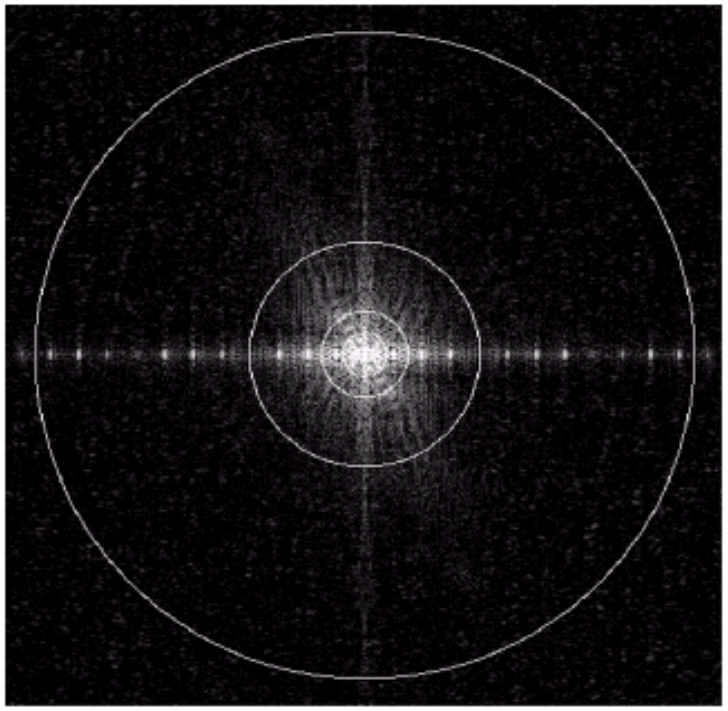
Allows user to specify precisely which frequencies are to be kept and which removed – good if you know which are which

ACK: Prof. Tony Pridmore, UNUK

Ideal Low-pass Filtering

- Sample image and its Fourier spectrum
- Circles mark ideal cut-offs at D_0 = 5, 15, 30, 80, 230



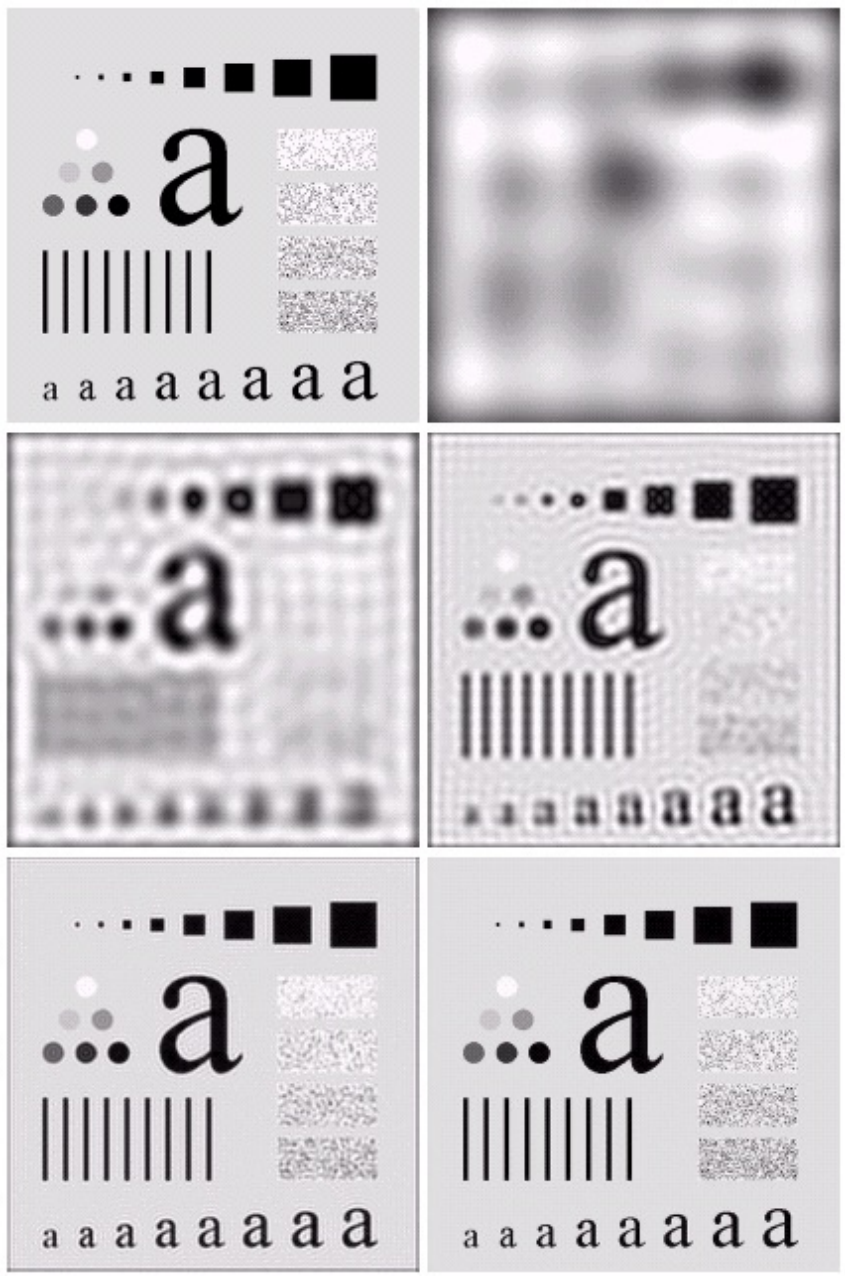


ACK: Prof. Tony Pridmore, UNUK

Unintended Consequences

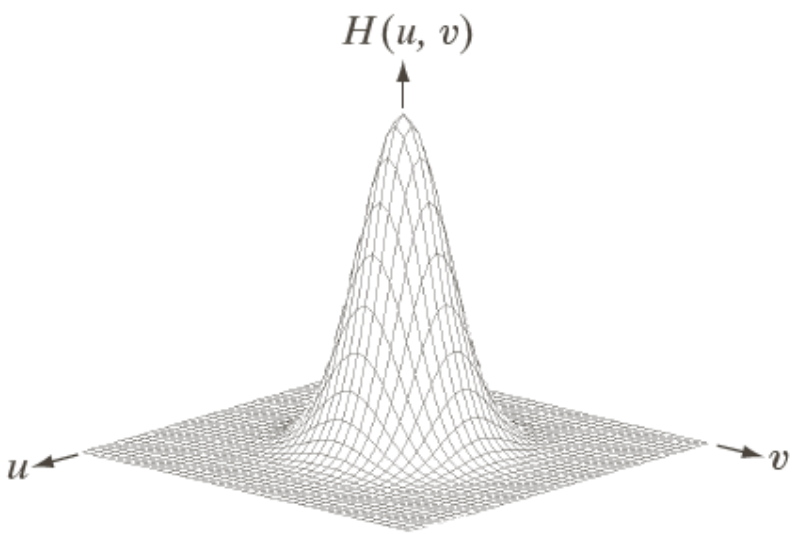
- Ideal low-pass filtering with (*top-left to bottom-right)* D_0 = 5, 15, 30, 80, 230
- Note the *ringing* artefact caused by hard boundary between frequencies kept and removed
- Frequency of ringing effect increases with D_0

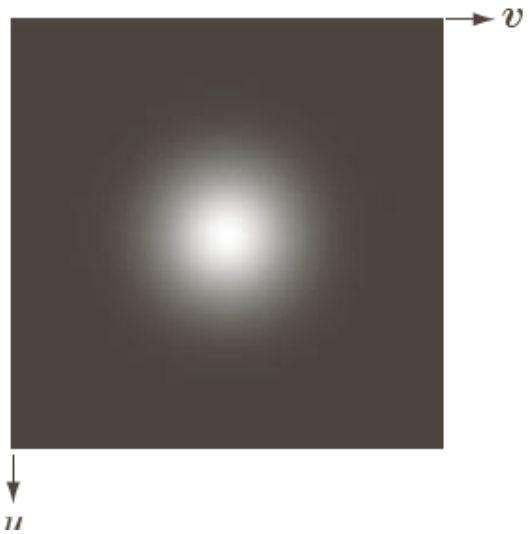
  - The “missing” components needed to make the uniform regions uniform become higher frequency as D_0 increases

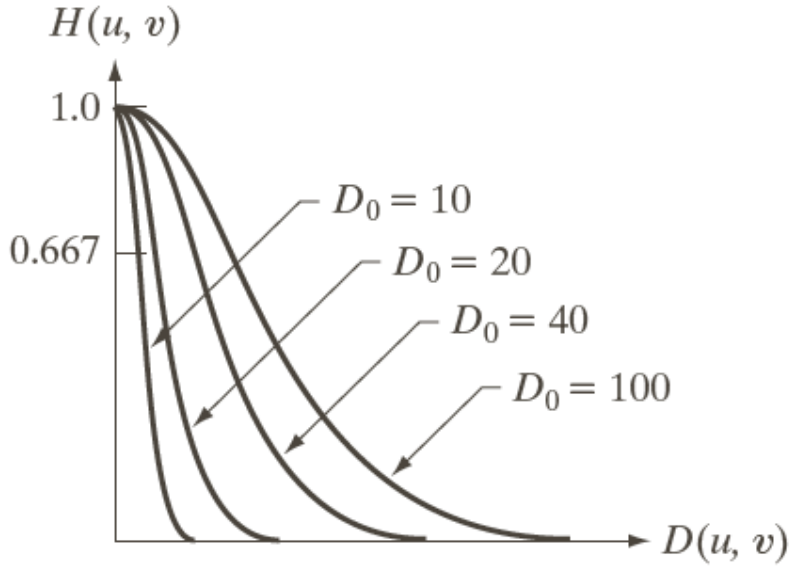


ACK: Prof. Tony Pridmore, UNUK

Low-pass Gaussian Filters







Perspective plot

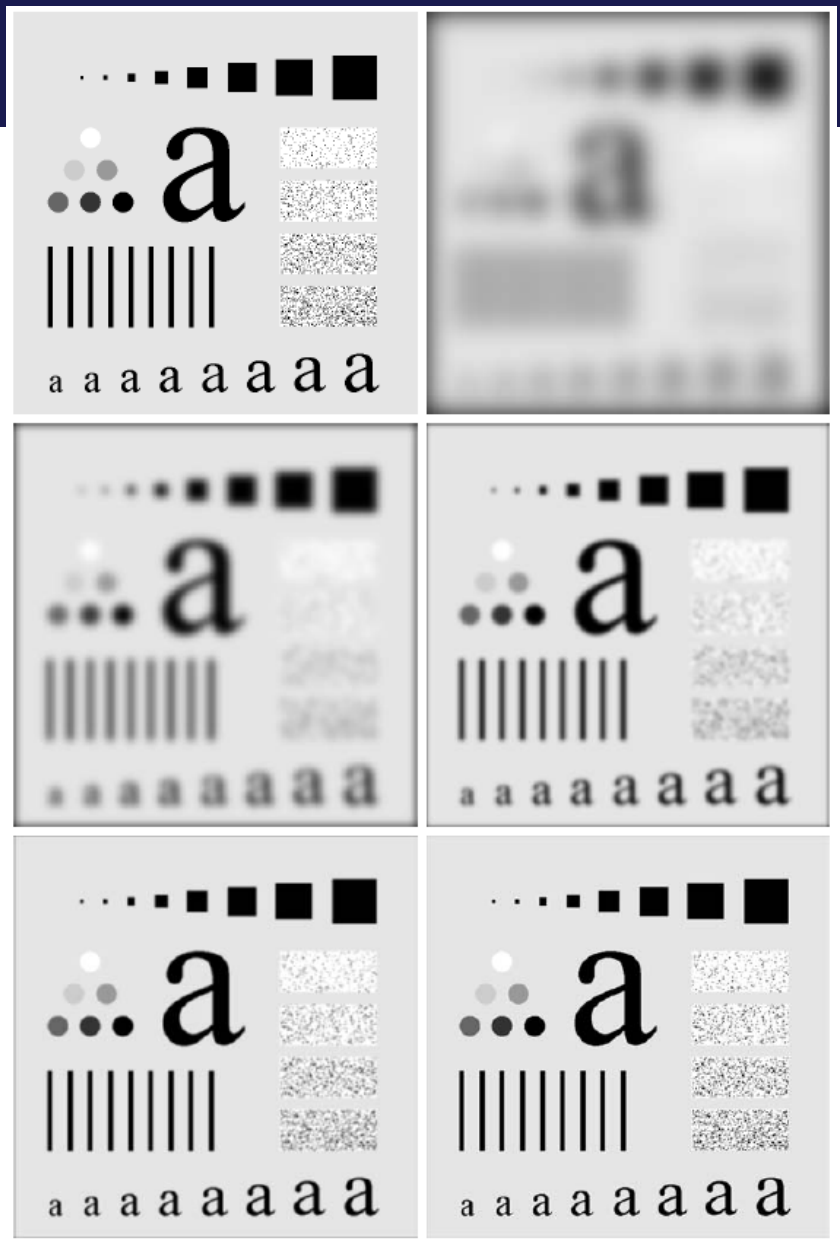
Image view

Profile

ACK: Prof. Tony Pridmore, UNUK

Low-pass Gaussian Filters

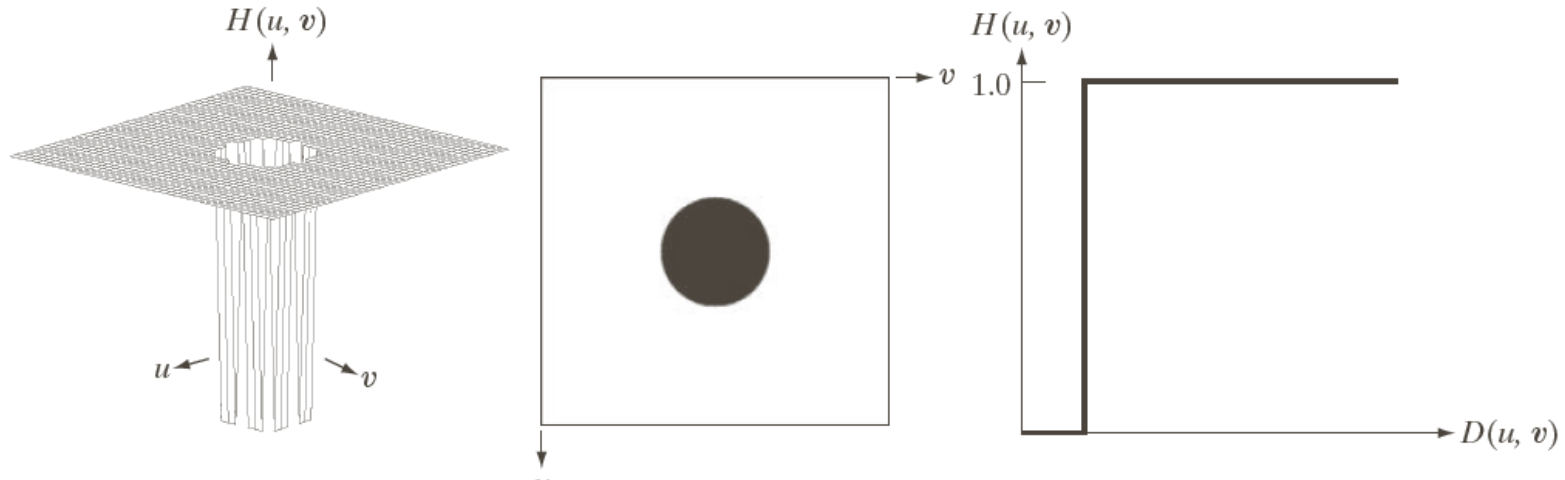
Low-pass Gaussian Filtering with cut-off frequencies at (*top-left to bottom-right)*
D_0 = 5, 15, 30, 80, 230



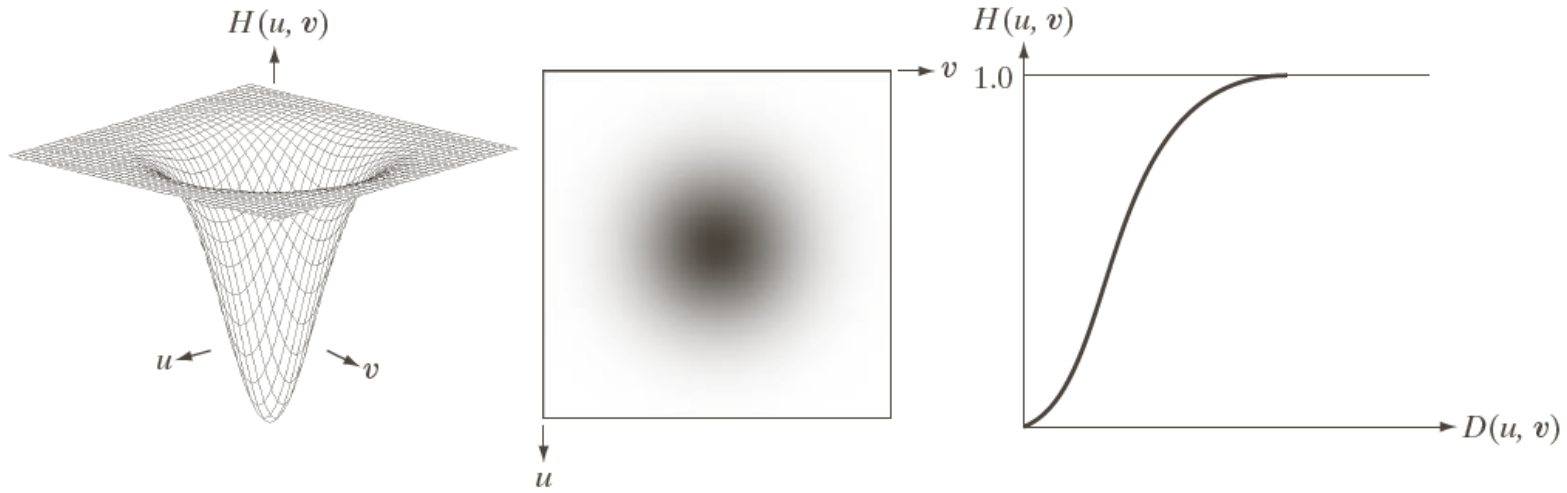
ACK: Prof. Tony Pridmore, UNUK

High-pass Filters

Ideal



Gaussian



ACK: Prof. Tony Pridmore, UNUK

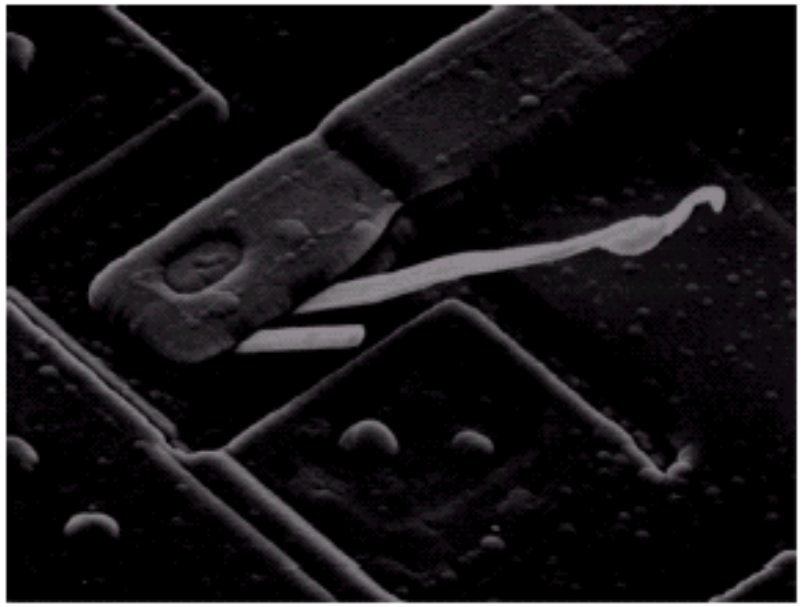
High-pass Gaussian Filters

Gaussian High-pass filtering with D_0 = 30, 60 and 160

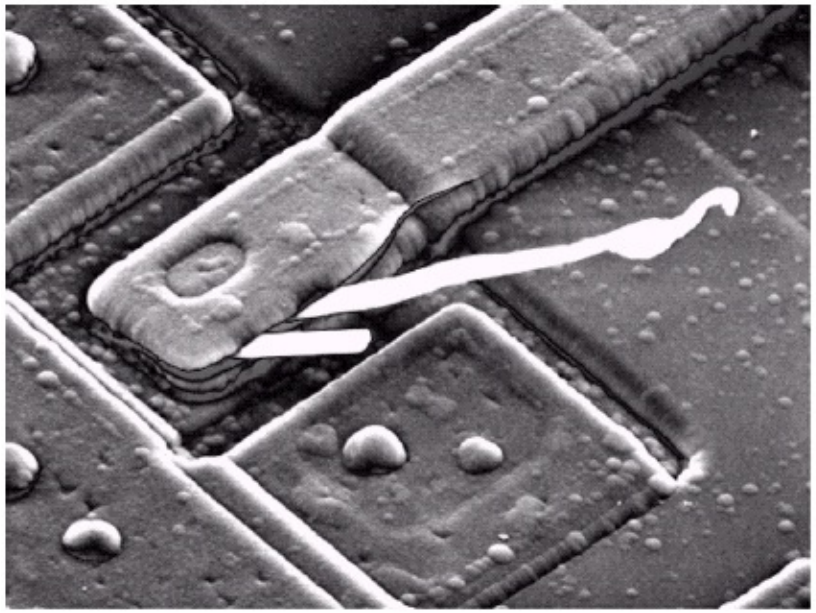
ACK: Prof. Tony Pridmore, UNUK

High-pass Filtering

Raw high-pass filters set origin, and so mean intensity to 0, making images dark. Add a small constant to the filter so the mean remains positive



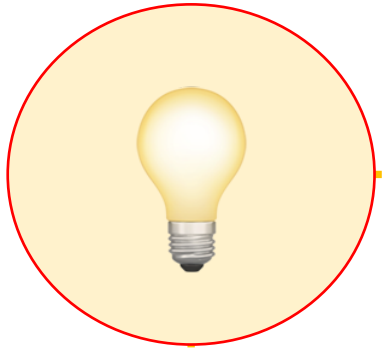
Gaussian high pass filtered SEM image



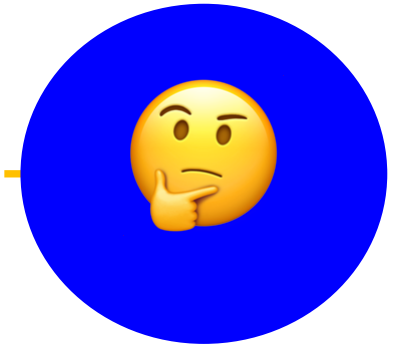
Gaussian high pass filter plus a constant

ACK: Prof. Tony Pridmore, UNUK

Summary



1. How to find lines?
2. The Hough Transform
3. Other parameter spaces
4. What is frequency domain?
5. The frequency domain processing




University of
Nottingham
UK | CHINA | MALAYSIA
Questions


University of
Nottingham
UK | CHINA | MALAYSIA
NEXT:
Image Compression