Module IN3031 / INM378 Digital Signal Processing and Audio Programming

Tillman Weyde t.e.weyde@city.ac.uk

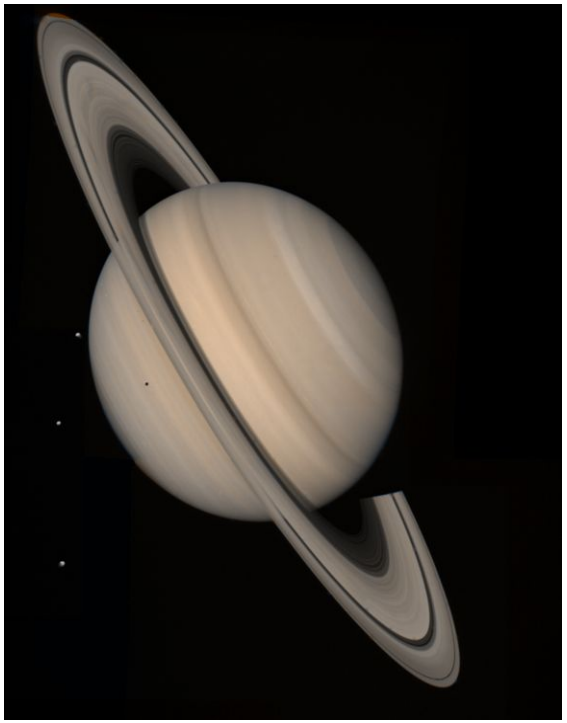
Image Processing Texts

John C. Russ: The image processing handbook. IEEE Press, 3rd edition, 1999.

Rafael C. Gonzalez and Richard E. Woods: Digital image processing. Addison Wesley, 1993.
Chapters 1-2 available at: http://www.imageprocessingplace.com

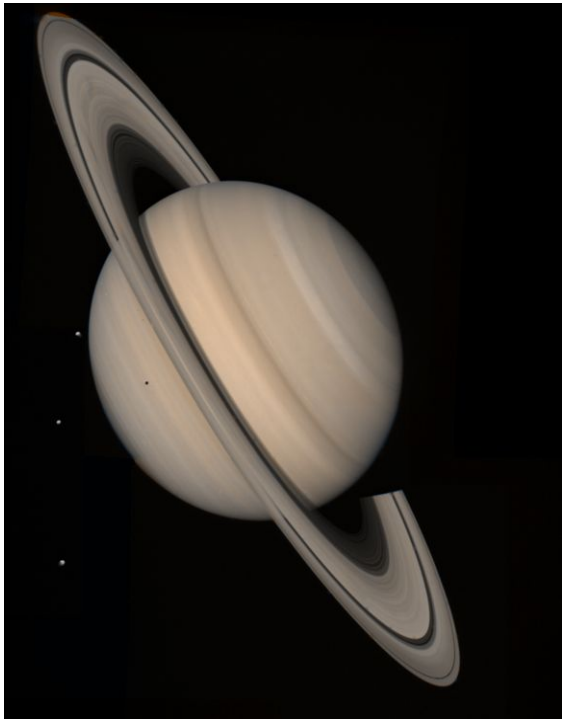
What is an image? (1)

- A visible impression obtained by a camera, telescope, microscope, or other device, or displayed on a computer or video screen - *Oxford Dictionaries*



What is an image? (2)

- In form of function f(x,y)
- Most images are rectangular

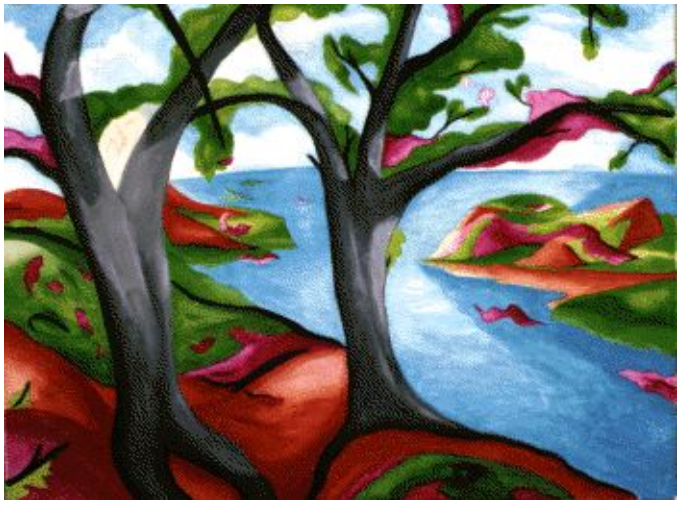


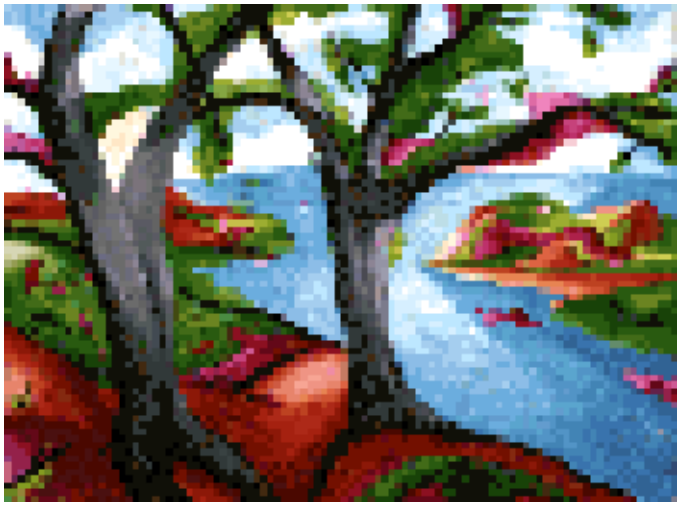
What is Digital Image Processing?

- It concerns the transformation of an image to a digital format and its processing by computers
- Processing: analysis, synthesis, modification
- Why process?
  - Facilitate storage and transmission
  - Restore or enhance images
  - Extract information from images
  - ...

What is a digital image?

Discrete samples f[x,y] representing the continuous image f(x,y)
Image size: max(x) x max(y), e.g. 800x600
Pixels ("picture elements"): elements of the 2-D array





Topics in Digital Image Processing

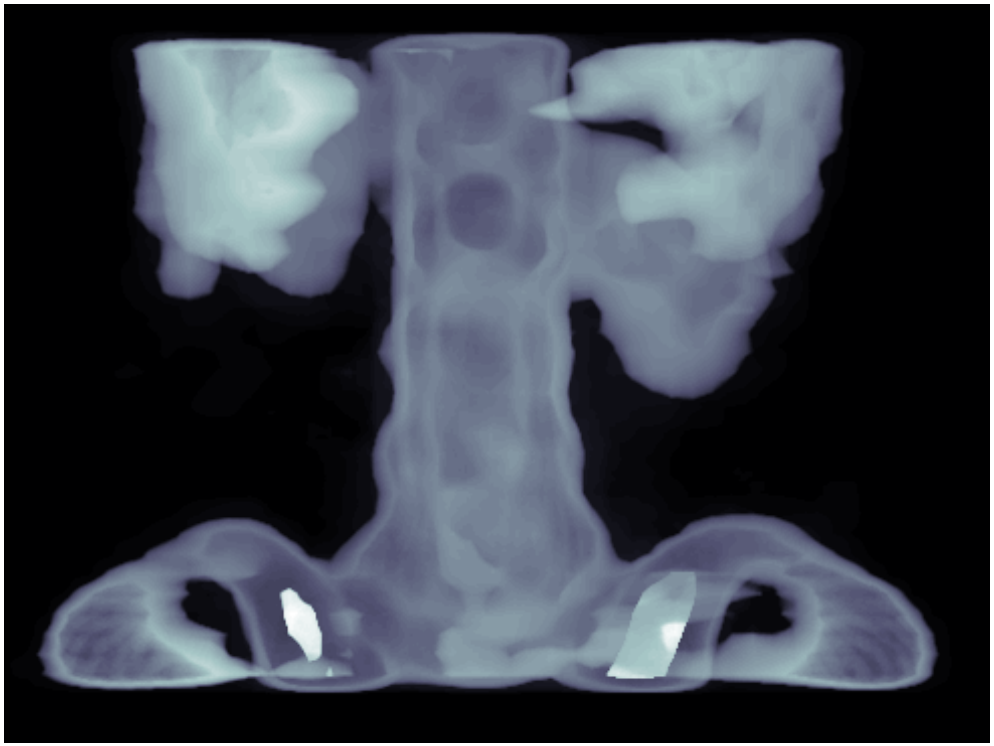
Image compression

PNG (300 kB)

JPG (3 kB)

Topics in Digital Image Processing

Medical Imaging

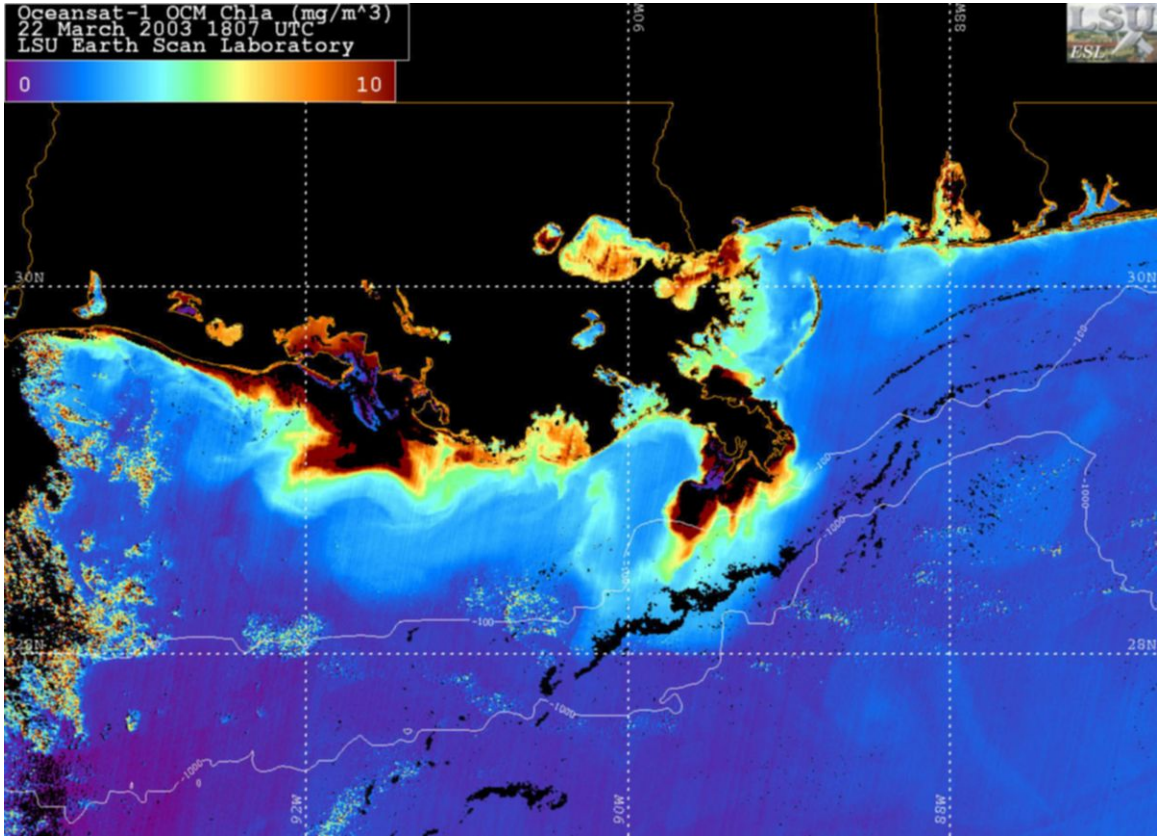


Topics in Digital Image Processing

Digital Image Restoration/Enhancement

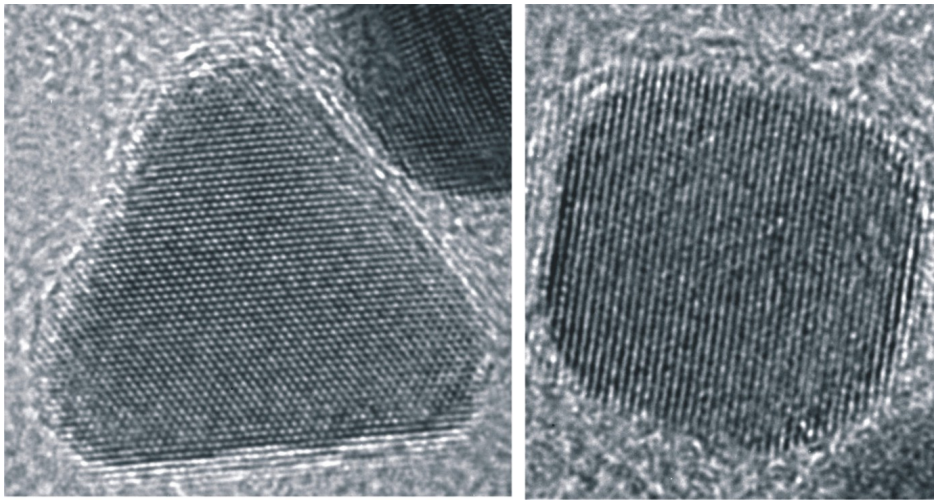
Topics in Digital Image Processing

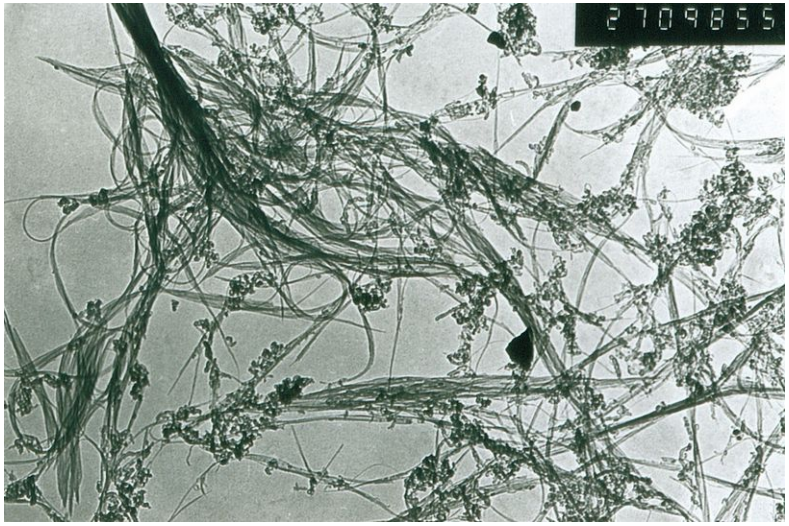
Remote Sensing



Topics in Digital Image Processing

Microscopy





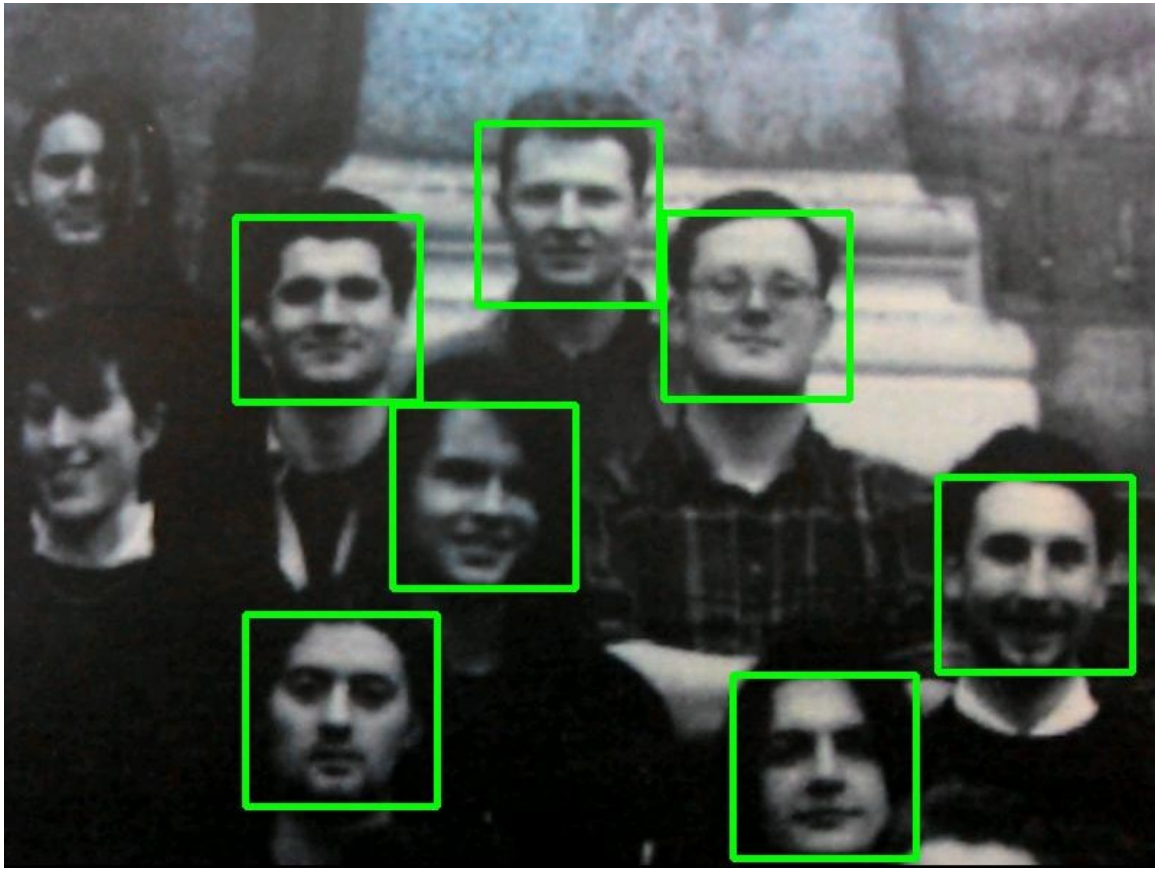
Transmission electron microscopy (TEM) images

Topics in Digital Image Processing

Image Registration

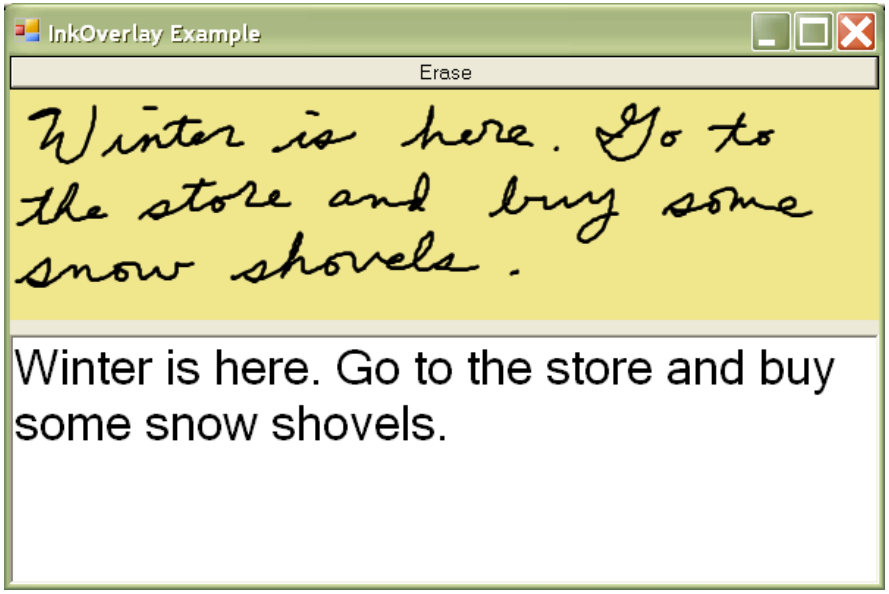
Topics in Digital Image Processing

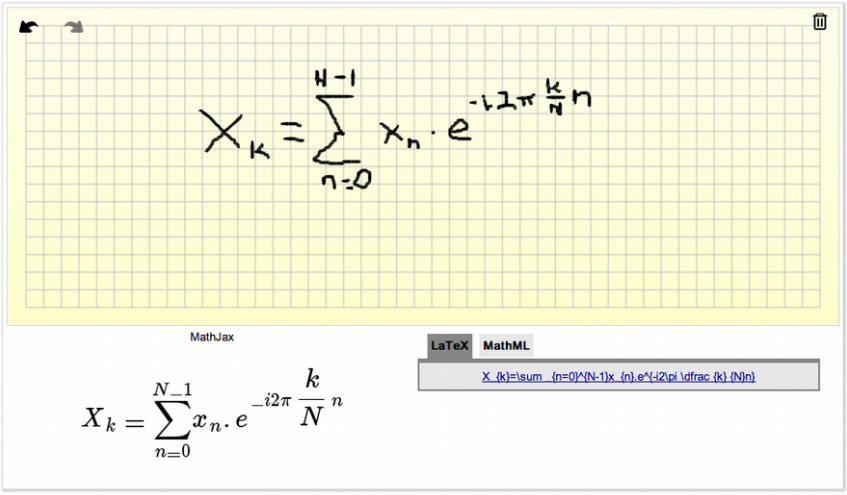
Face Detection



Topics in Digital Image Processing

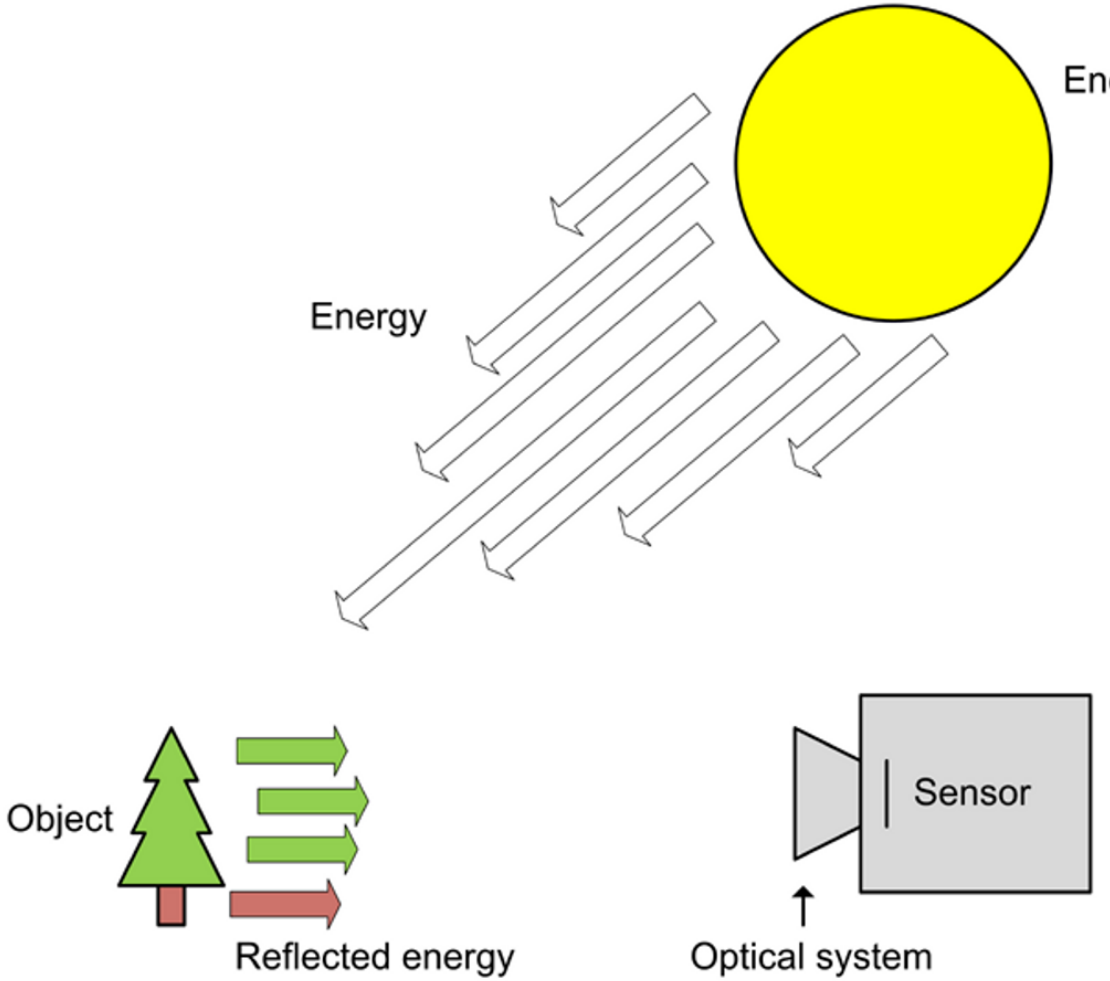
Handwriting recognition





Digital Image Acquisition

Digital Image Acquisition (1)

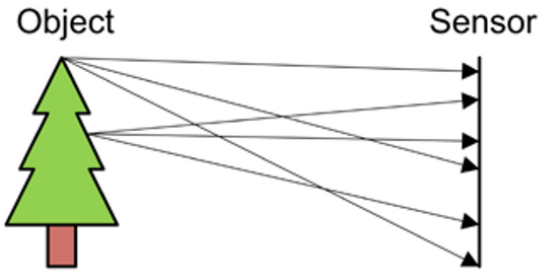


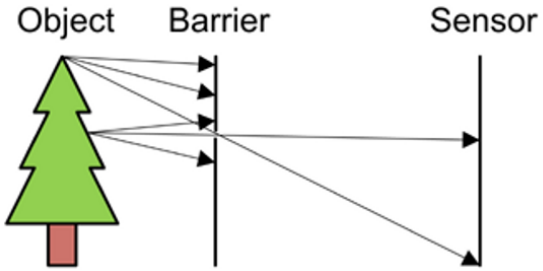
Overview of the typical image acquisition process, with the sun as light source, a tree as object and a camera to capture the image.

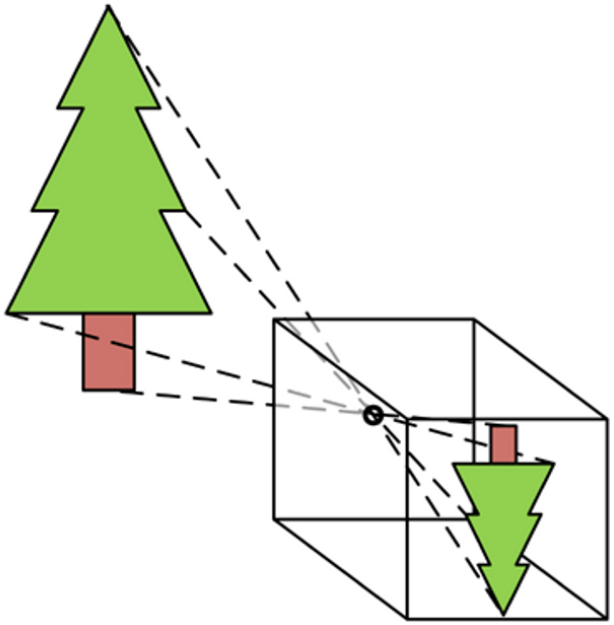
An analog camera would use a film where the digital camera uses a sensor.

Digital Image Acquisition (2)

- Light reflected from the object has to be captured
- A barrier (also called pinhole model) is placed between the object of interest and the sensing material

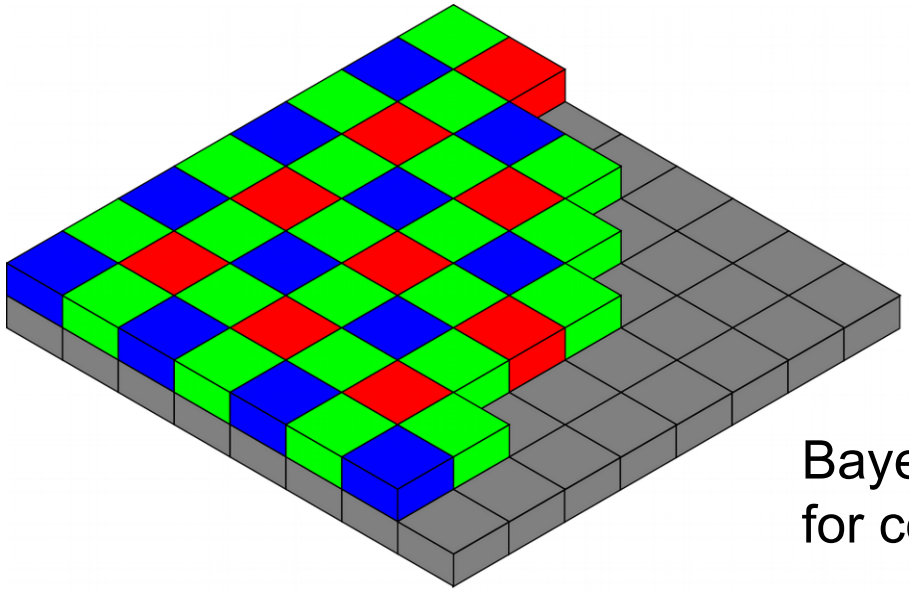






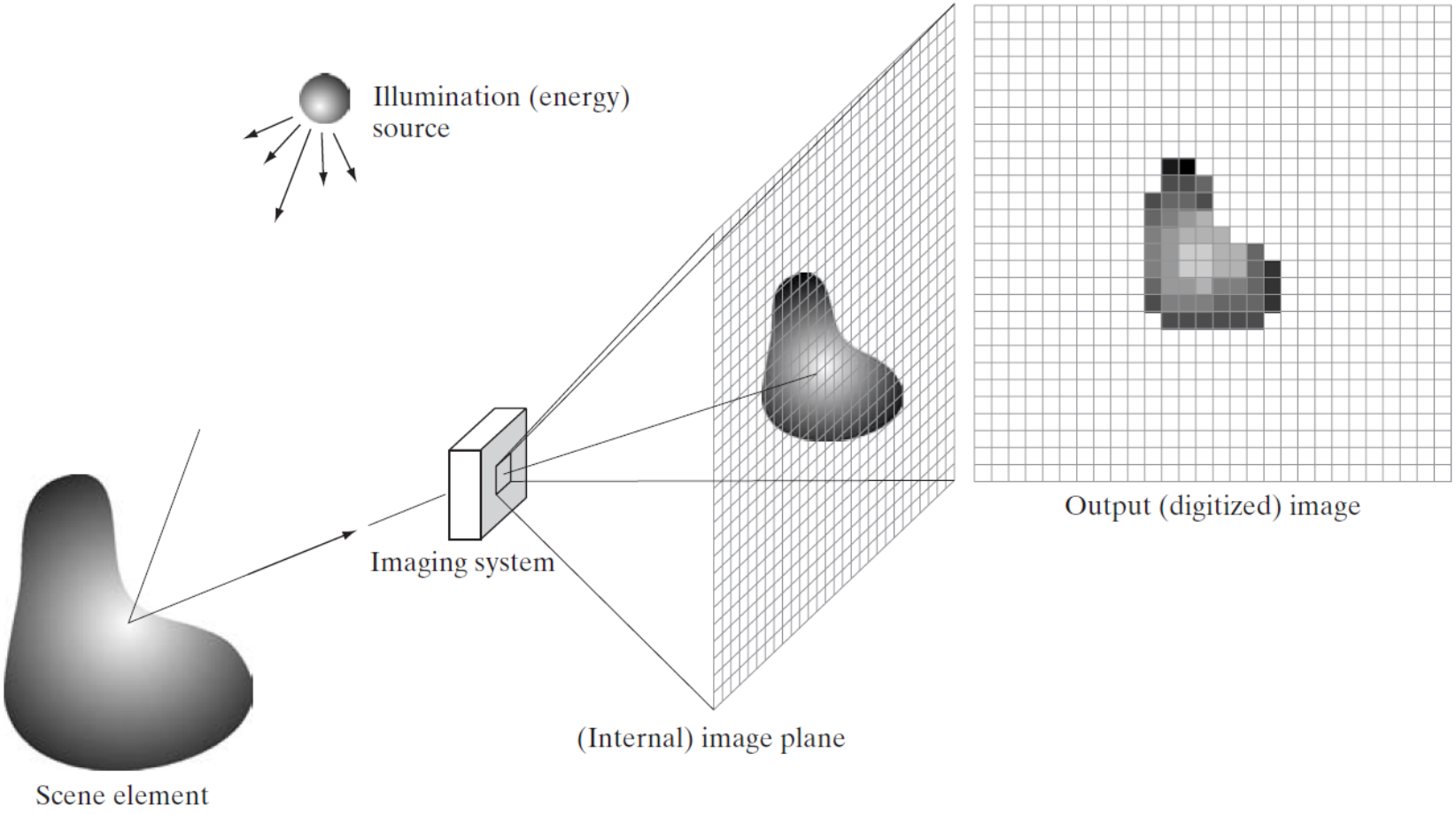
Digital Image Acquisition (3)

- Image sensors are used for recording the input light
- An image sensor consists of a 2D array of cells
- Each cell measures light and converts it into voltage
- Digital image sensors: CCD (charge-coupled devices), CMOS (complementary metal–oxide–semiconductors)



Bayer arrangement on sensor for colour image acquisition

Sampling and Quantization (1)



Sampling and Quantization (2)

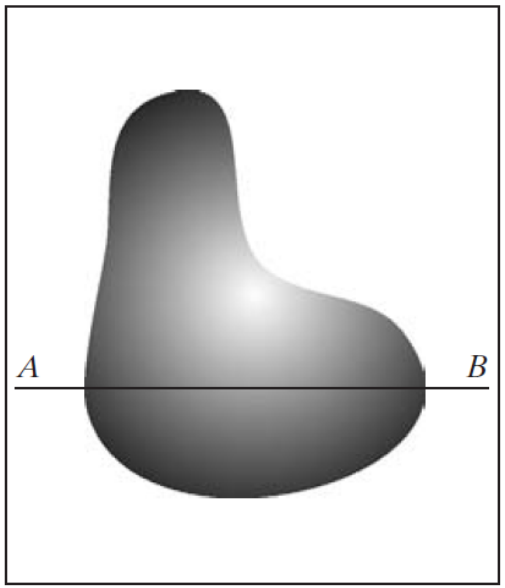
- To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: sampling and quantization.

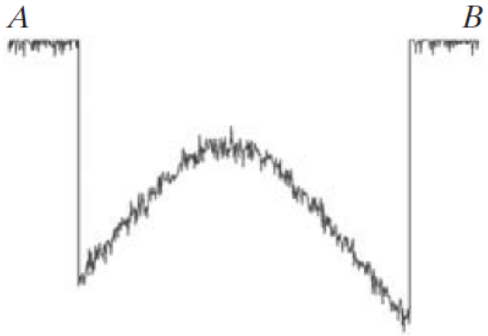
- To convert it to digital form, we have to sample the function f(x,y) in both coordinates and in amplitude.

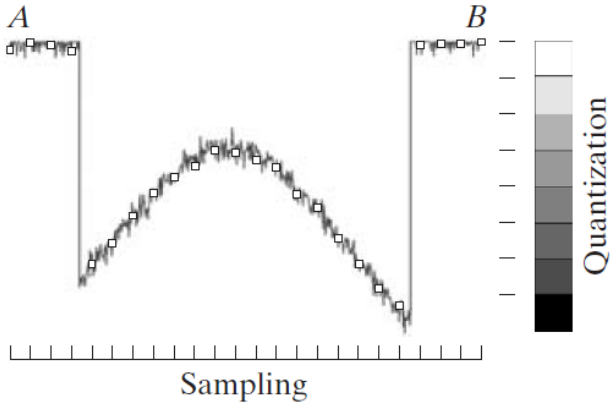
- Digitizing the coordinate values is called sampling.

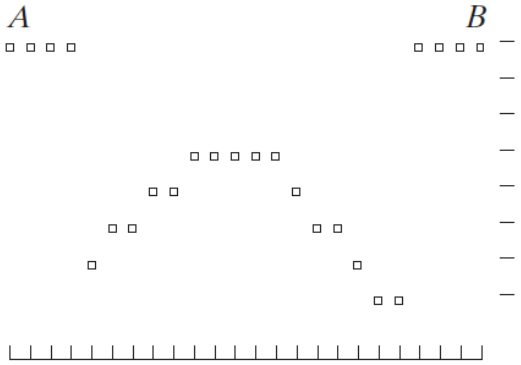
- Digitizing the amplitude values is called quantization.

Sampling and Quantization (3)



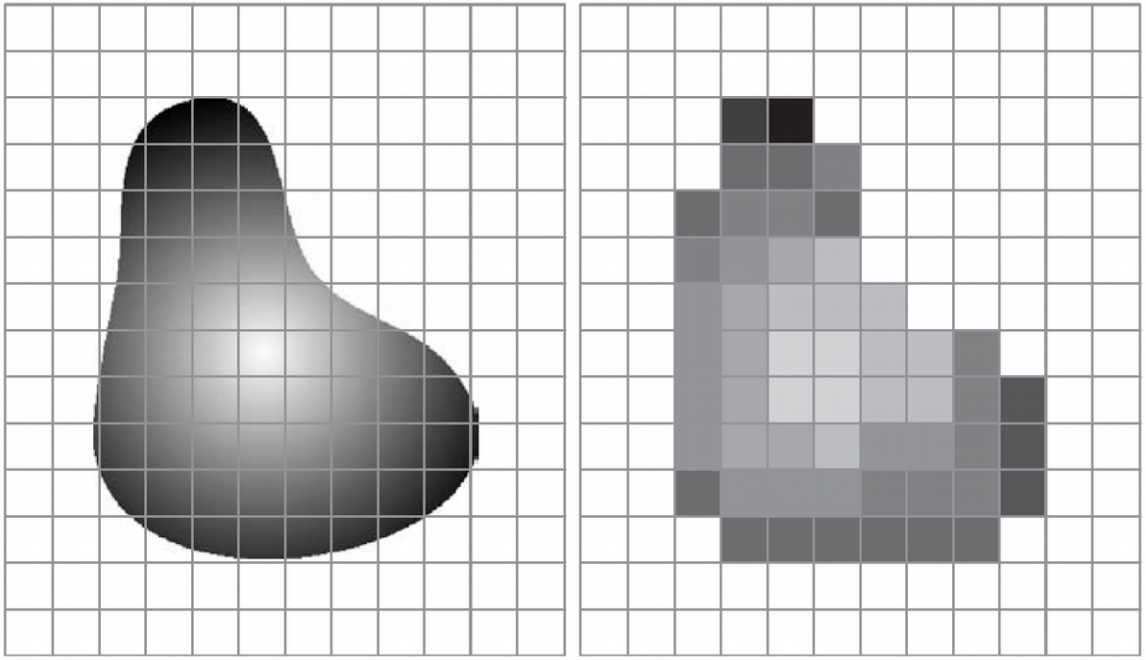






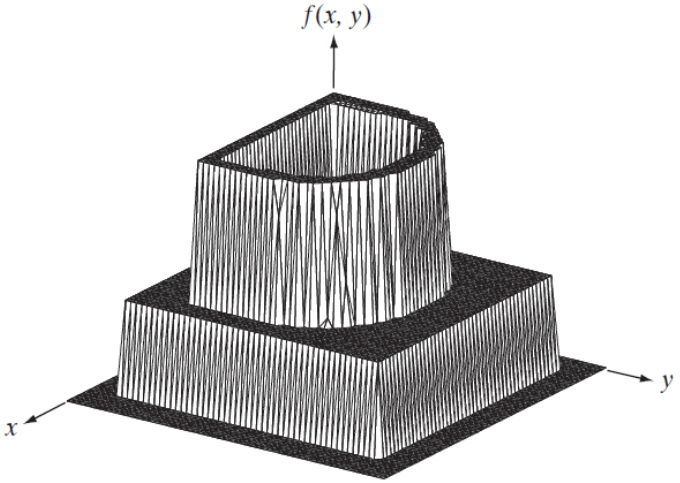
Sampling and Quantization (4)

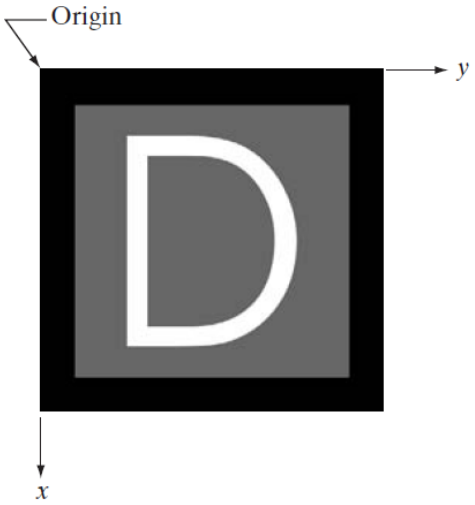
- The quality of a digital image is determined to a large degree by the number of samples and discrete intensity levels used in sampling and quantization.



Representing Digital Images

Image Representation (1)





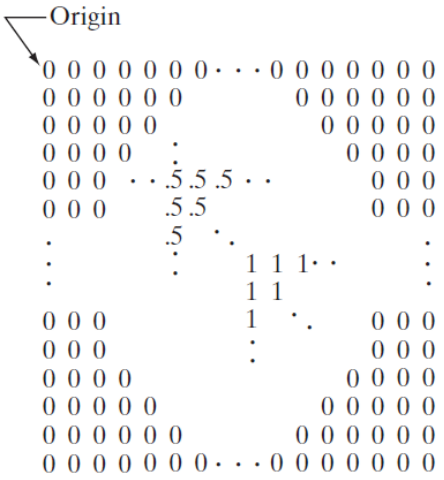


Image Representation (2)

- A digital image can be conveniently represented by a N x M matrix:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & \cdots & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

Each element of the array is called a pixel.

Spatial Resolution

- Sampling is the principal factor in determining the spatial resolution of an image:

256x256

128x128

64x64

Gray Levels

- Gray-level resolution refers to the smallest representable change in gray level.
- Usually an integer power of 2 (e.g. 8 bits = 256) as number of levels
- Figure: image in 16, 8, 4, and 2 gray levels

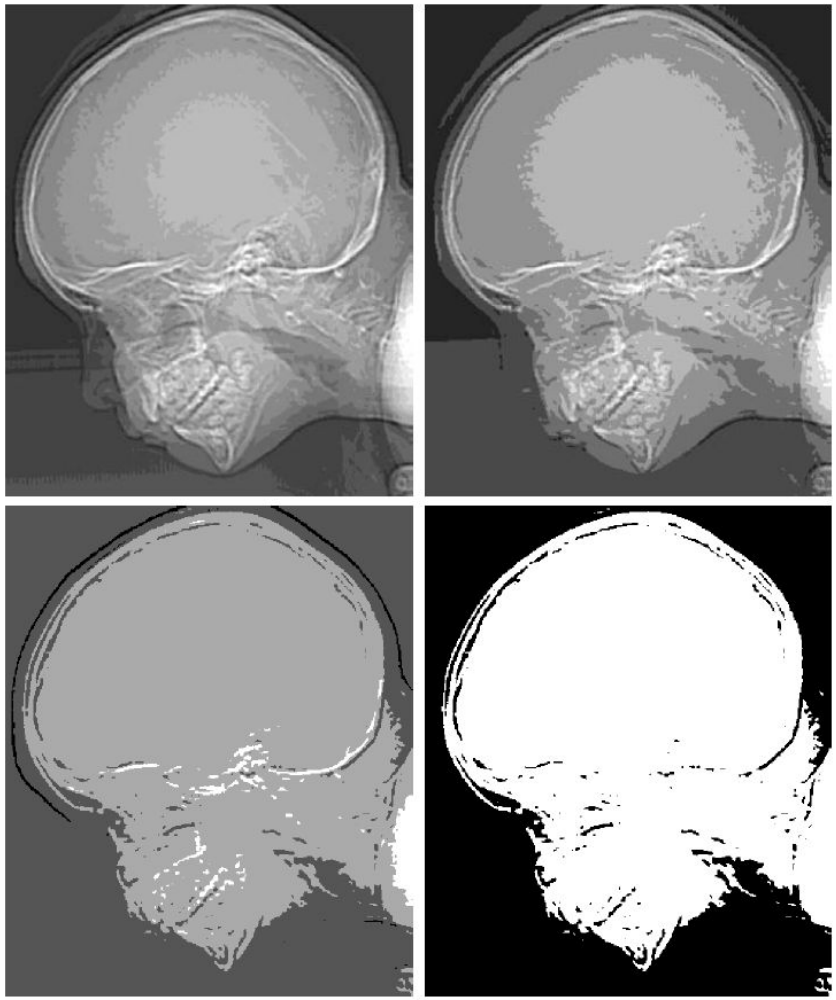
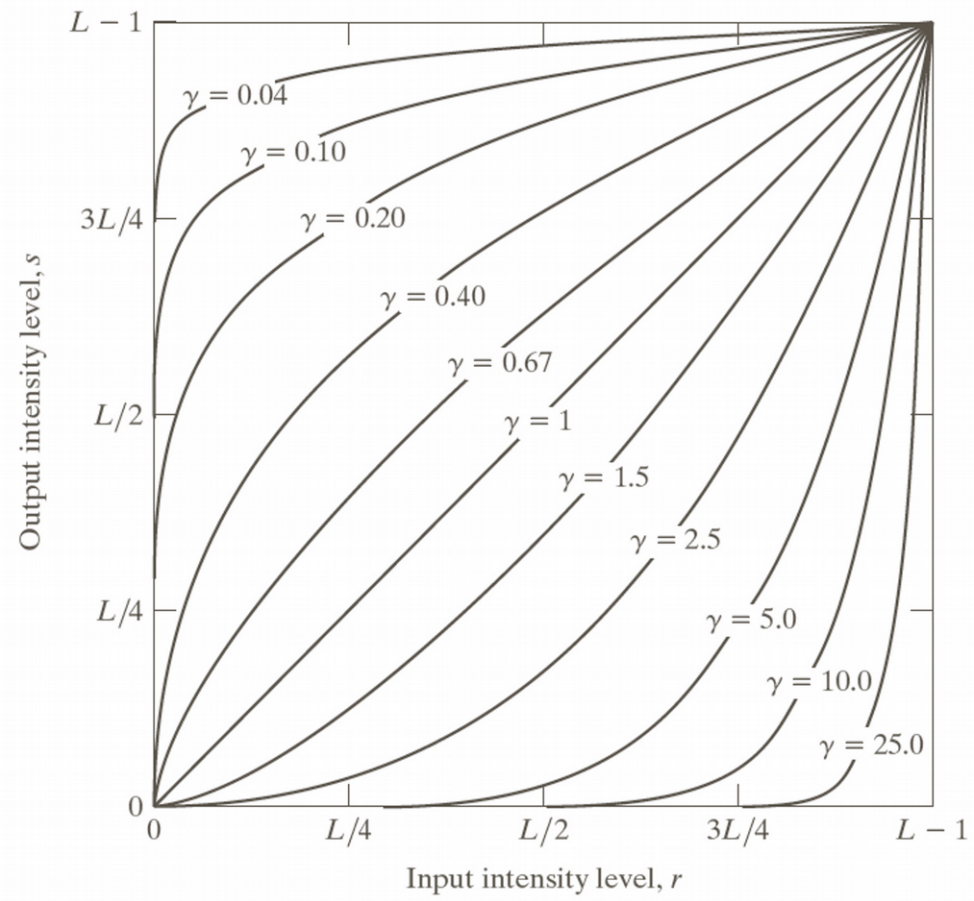


Image Negatives

- Given an image with L gray levels, its negative transformation for pixel with level r is given by: $s = L\text{-}1\text{-}r$

Gamma Correction (1)

$s = cr^{\gamma}$ (c: constant)



- In some devices (e.g. CRT TVs) the gamma curve is nonlinear, and needs to be corrected.

- Gamma correction is important in displaying images correctly (otherwise they might be either bleached or dark).

Gamma Correction (2)

γ = 1.0

γ = 0.5

γ = 1.5

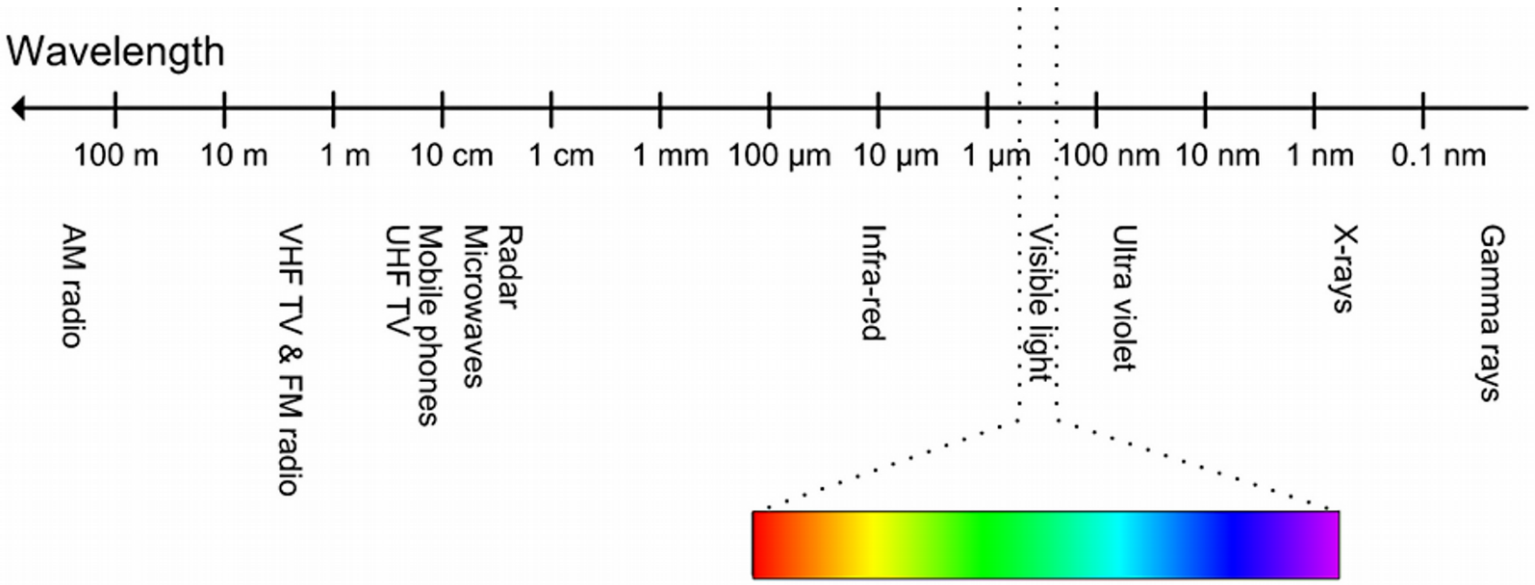
Elementary Image Processing Operations

- Addition: $C[i][j] = A[i][j]+B[i][j]$
- Subtraction: $C[i][j] = A[i][j]-B[i][j]$
- Multiplication by a constant c: $C[i][j] = c\,A[i][j]$

Colour Image Processing

Colour Image Processing (1)

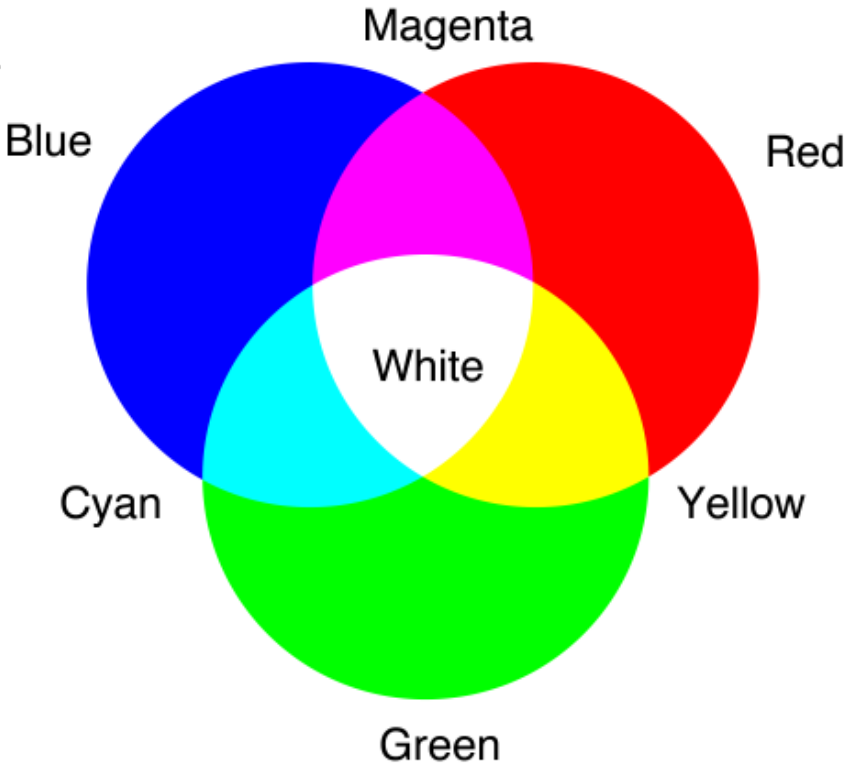
- Visible light is composed of a relatively narrow brand of frequencies in the electromagnetic spectrum.
- Chromatic light spans the electromagnetic spectrum from approximately 400 to 700 nm.



Colour Image Processing (2)

- 6 to 7 million cones in the human eye can be divided into 3 principal sensing categories, corresponding roughly to red (R), green (G), and blue (B) – also called primary colours.

- The primary colors can be added to produce the secondary colors of light – cyan (C), magenta (M), and yellow (Y).

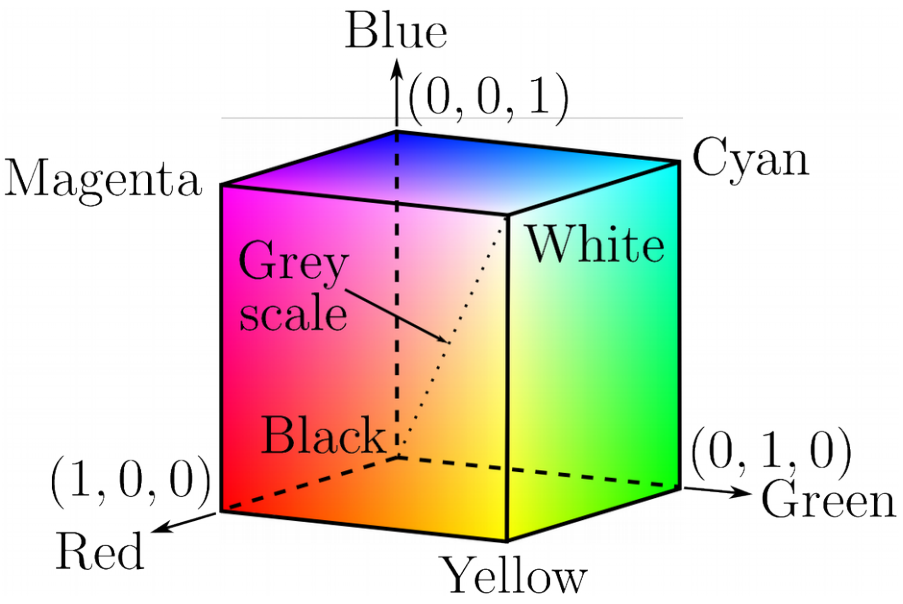


Colour Image Processing (3)

- Characteristics used to distinguish one color from another are brightness, hue, and saturation.
  - Hue represents the dominant color as perceived by an observer.
  - Saturation refers to the relatives purity or the amount of white light mixed with a hue.
  - Brightness embodies the notion of intensity.

Colour Models (1)

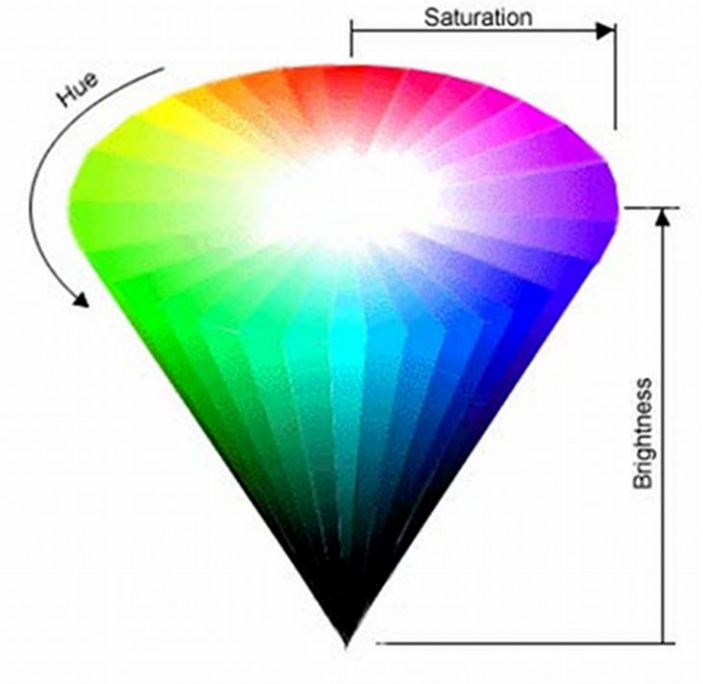
- The RGB (red, green, blue) Colour Model: used in colour monitors, most video cameras and many file formats.
- Red, green, and blue are each 8-bit images.



Colour Models (2)

Colour Models (3)

- CMY (cyan, magenta, yellow) and CMYK (cyan, magenta, yellow, black) models for color printing.
- HSI (hue, saturation, intensity) model, which corresponds closely with the way humans describe and interpret color.



Colour Models (4)

- Converting RGB to CMY:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Colour Models (5)

- Converting RGB to HSI:

$$H = \begin{cases} \theta & if \quad B \le G \\ 360 - \theta & if \quad B > G \end{cases}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$$

$$I = \frac{1}{3}(R+G+B)$$

Histograms

Image Histograms (1)

- Distribution of the pixel intensity values

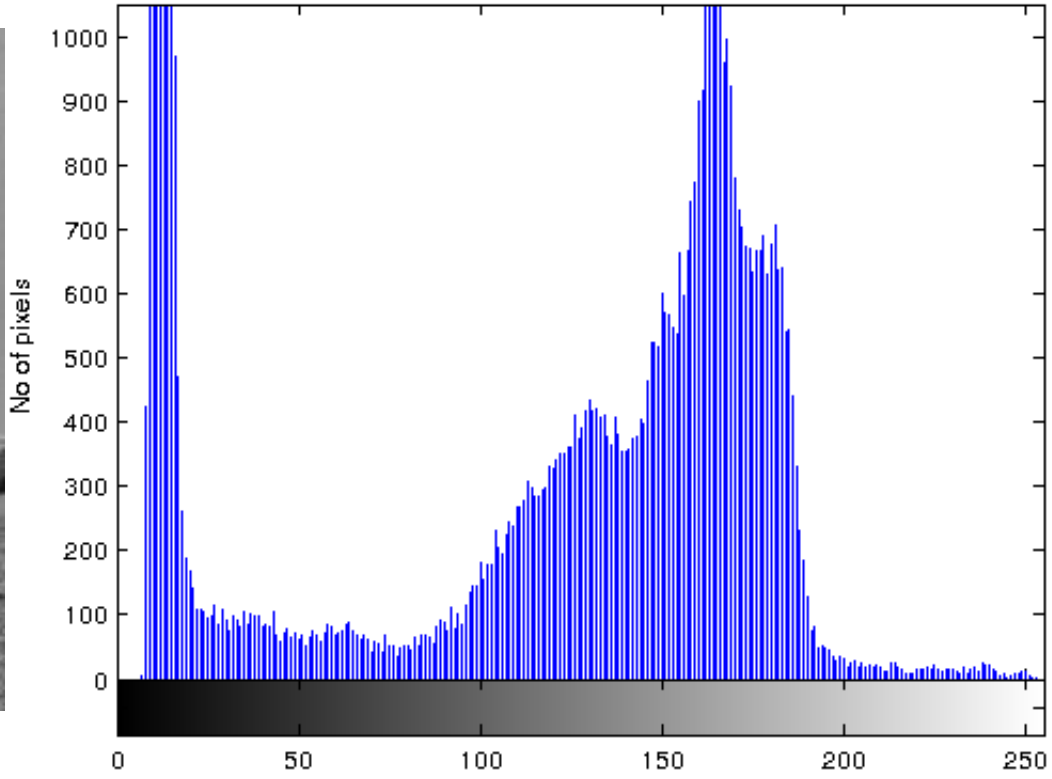
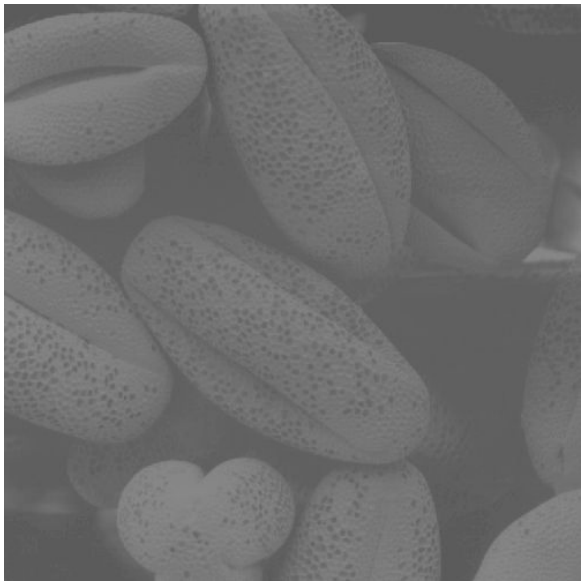
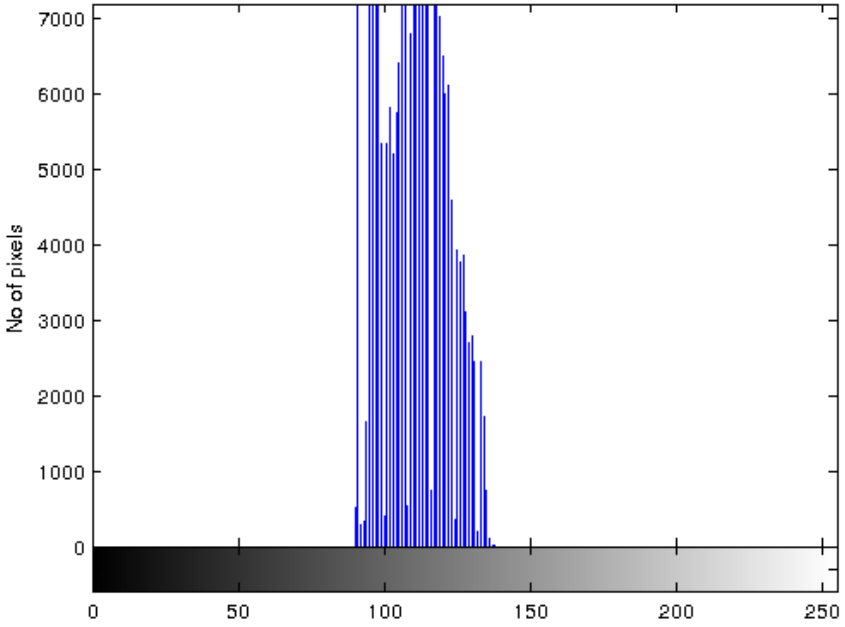


Image Histograms (2)

- Histogram manipulation can be used for image enhancement.
- Figure: image with low contrast





Histogram Equalisation (1)

- Histogram equalisation: method for contrast adjustment
- Increases the contrast of a low-contrast image
- Goal: find a transformation

$$s = T(r)$$

that is applied to each pixel of the input image $x[m,n]$, such that a uniform distribution of gray levels results for the output image $y[m,n]$.

Histogram Equalisation (2)

- Implementation:
  - Compute the image histogram (n: total number of pixels, n_k: number of pixels with gray level r_k):

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0,1,2,\ldots,L-1$$

  Compute the cumulative histogram:

  - $$s_k = \sum_{j=0}^{k} p_r(r_j)$$

  New gray level values:

  - $$h(r_k) = \frac{s_k - s_{min}}{(MN) - s_{min}}(L-1)$$

Histogram Equalisation (3)

- Example:

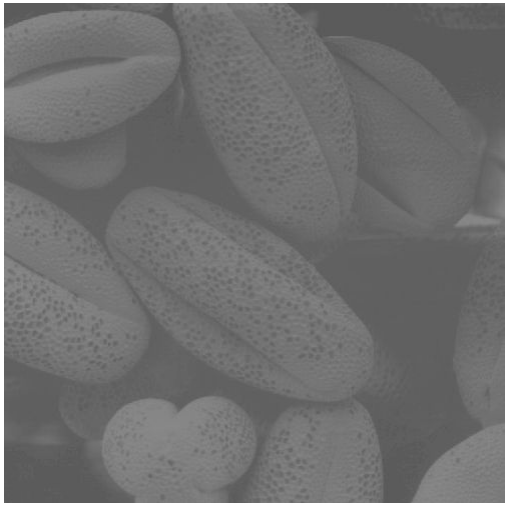
52	55	63	67	63
63	59	55	90	90
63	59	68	90	68
63	58	68	55	63
67	52	68	59	55

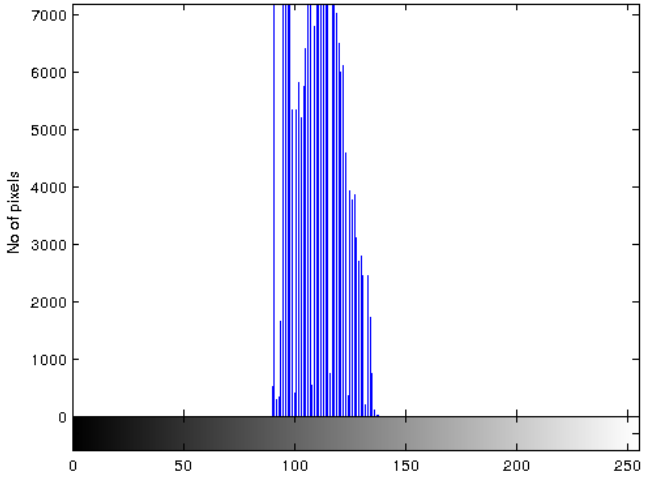
Image

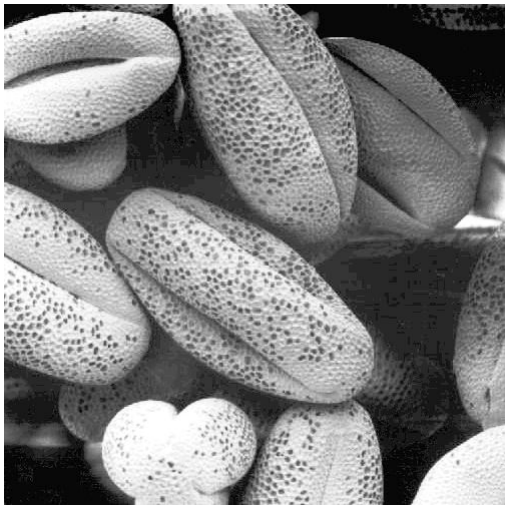
52	2	2	0
55	4	6	44
58	1	7	55
59	3	10	89
63	6	16	155
67	2	18	177
68	4	22	222
90	3	25	255

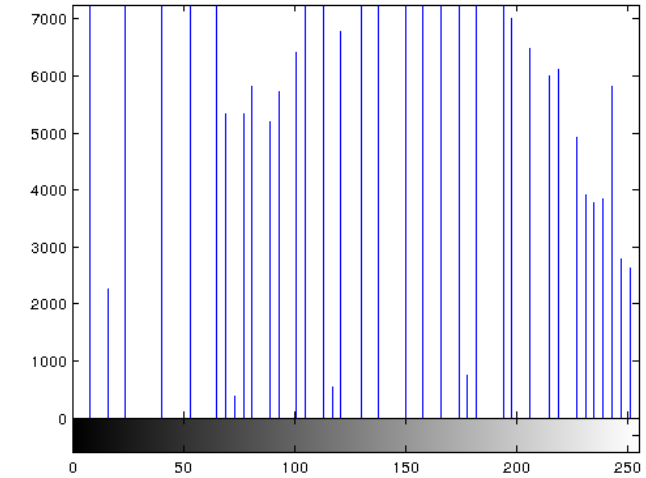
Pixels, histogram, cumulative histogram, new pixel values

Histogram Equalisation (4)





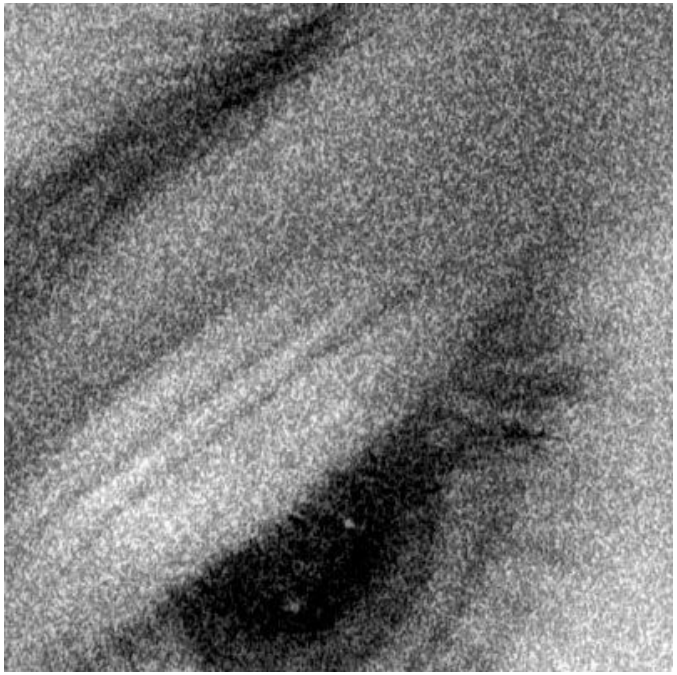


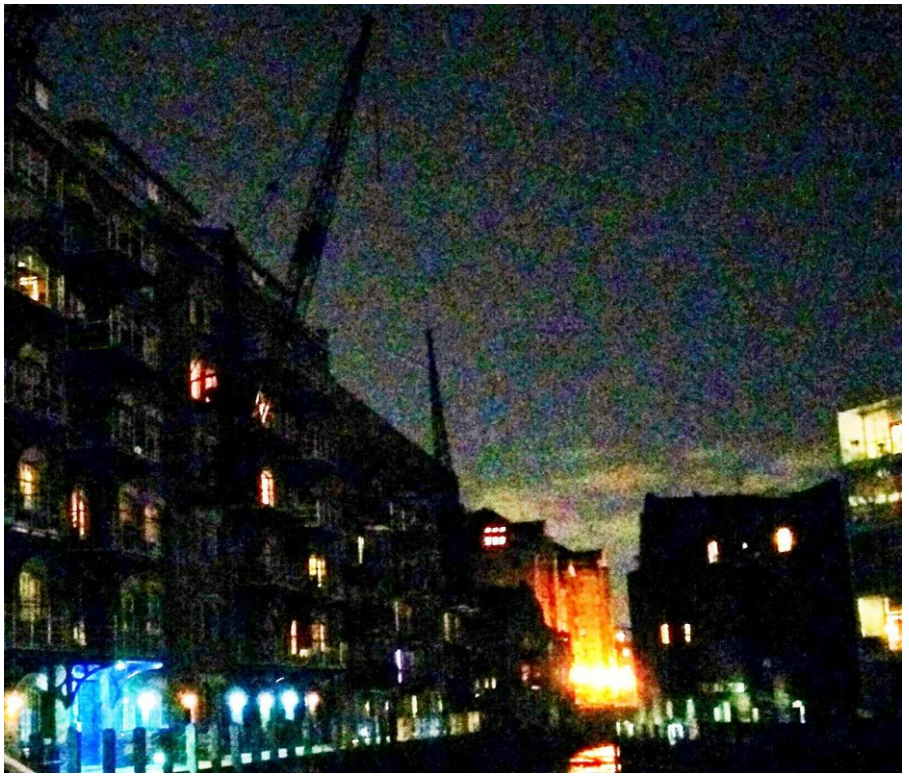


Noise in Image Processing

Noise in Image Processing (1)

- Digital images are corrupted by noise either during image acquisition (film grain noise, CCD noise...) or during image transmission.

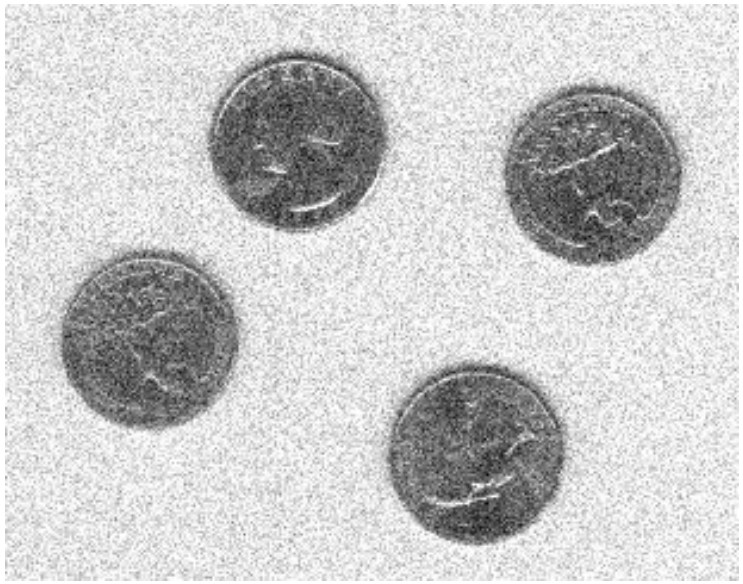




Noise in Image Processing (2)

- Types of noise: salt & pepper, Gaussian, uniform...

Salt & Pepper Noise
(impulsive noise)



Gaussian white noise

Basic Image Processing in Matlab

Image Processing in Matlab (1)

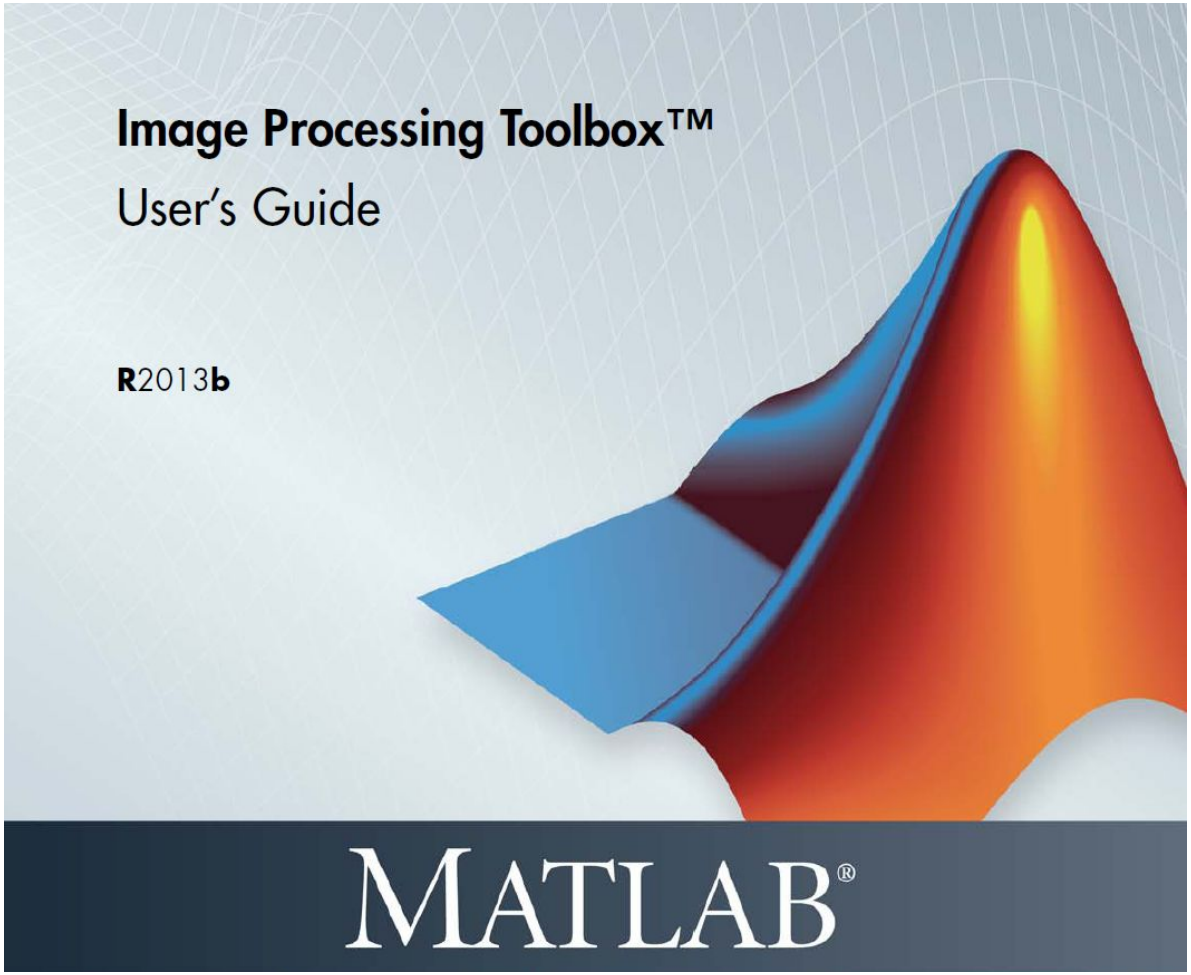


Image Processing in Matlab (2)

- The Matlab Image Processing Toolbox provides algorithms, functions, and apps for image processing and visualisation.

- Supports many image formats (JPG, TIF, PNG, BMP...)

- Tools for: image enhancement, filtering, analysis, segmentation, transformations, navigation, display...

- Check Moodle for MATLAB Download/Install info

- Check toolbox documentation!

Image Processing in Matlab (3)

- Read an image:

```
I = imread('pout.tif');
```

- Display an image:

```
imshow(I);
```

- Save image file:

```
imwrite (I2, 'pout2.png');
```

- Convert RGB image to grayscale:

```
J = rgb2gray(I);
```

Image Processing in Matlab (4)

- Image negative:

```
J = imcomplement(I);
```

- Image information:

```
imfinfo('pout2.png');
```

- Information about supported image formats:

```
imformats
```

- Resize image:

```
J = imresize(I,scale); //e.g. scale=1.5
```

Image Processing in Matlab (5)

- Add noise:

```
J = imnoise(I,'salt & pepper',0.02);
```

- RGB to HSV/HSI models:

```
I_HSV = rgb2hsv(I_RGB);
```

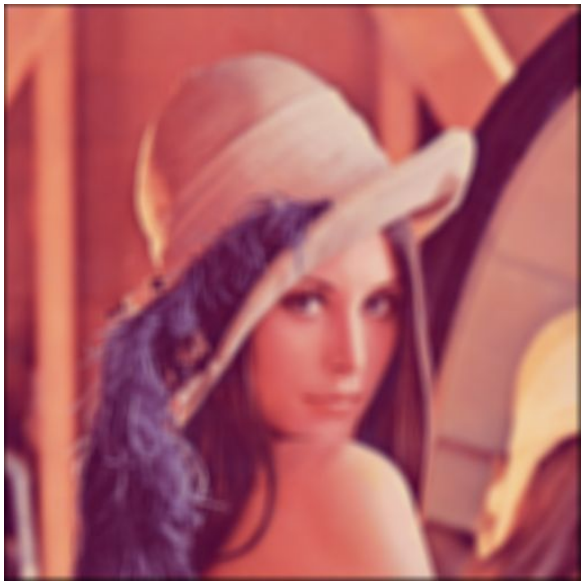
- Rotate image:

```
J = imrotate(I,angle);
```

- Crop image:

```
J = imcrop(I);
```

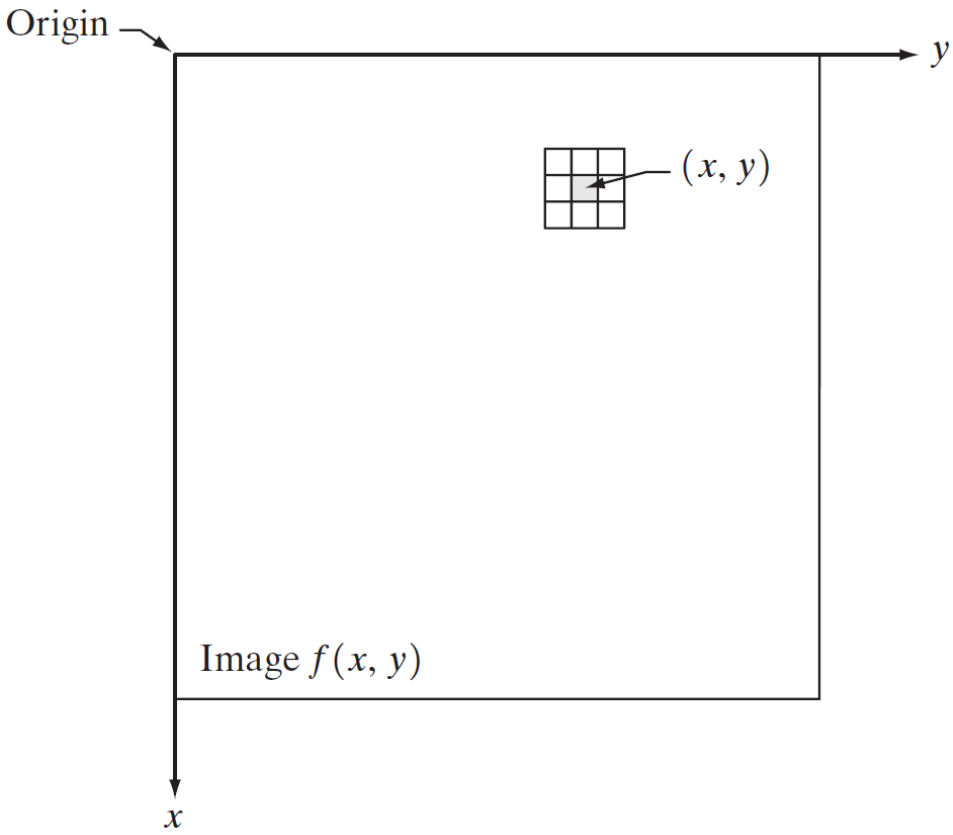
Digital Image Filtering - Transforms



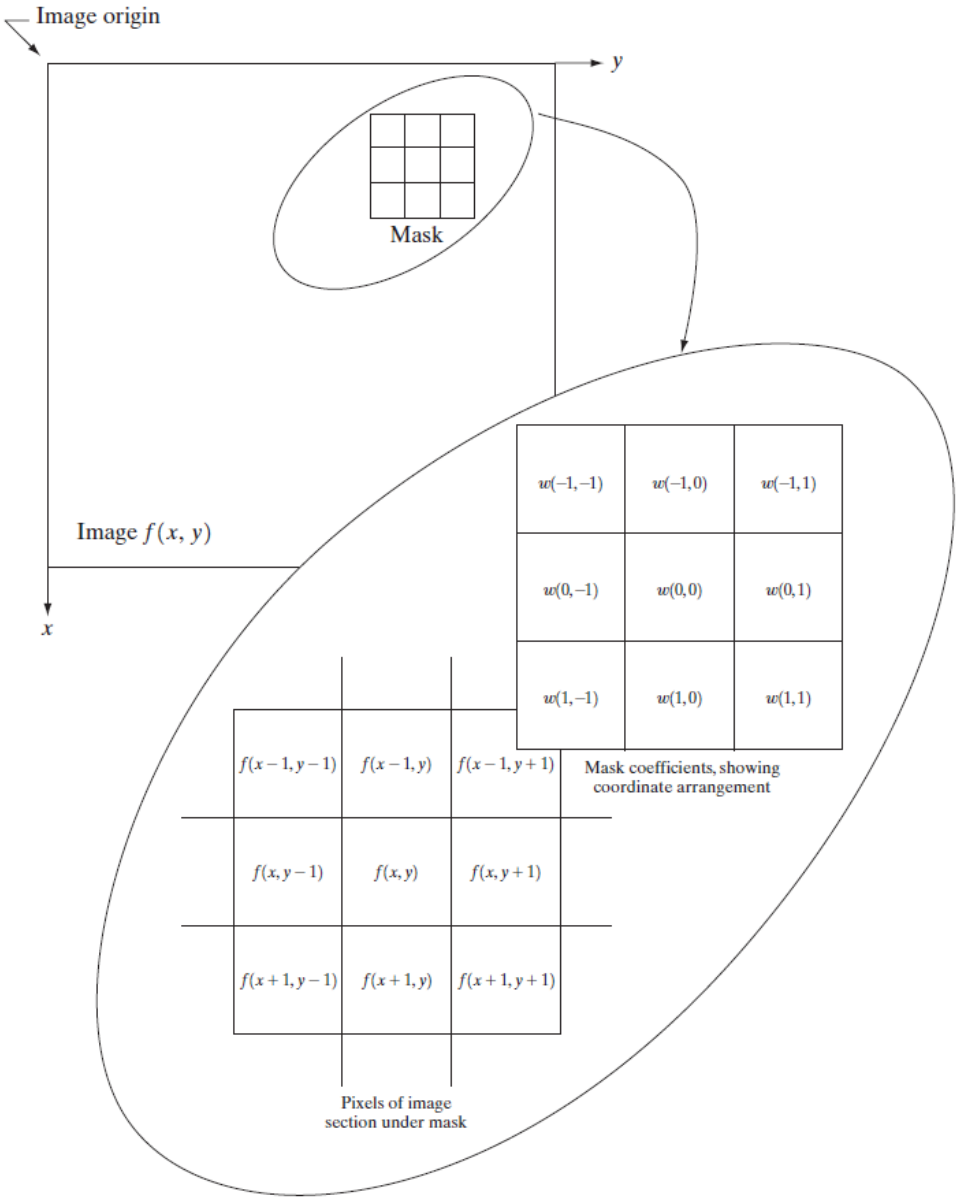
Spatial Filtering

Spatial Filtering (1)

- Spatial domain: pixels composing an image
- Spatial domain methods operate directly on pixels
- Spatial filtering typically uses neighborhood pixels
- Figure: Neighborhood about a point (x,y)



- Mask: small sub-image used for spatial filtering
- Each mask has corresponding mask coefficients
- Coefficients determine the effect of the filtering process (e.g. sharpening, blurring)
- Figure: a 3x3 mask



Spatial Filtering (3)

- Masks are sometimes called convolution masks
- Spatial filtering is directly related to convolution
- Figure: a 3x3 mask
- Mask output (z: pixel values):

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$R = \sum w_i z_i = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$$

Smoothing Filters (1)

- Smoothing filters are used for blurring and noise reduction

Smoothing Filters (2)

- Smoothing linear filter: its output is the average of the pixels contained in the neighborhood of the filter mask.
- Also called averaging filters or lowpass filters.
- The resulting image has reduced “sharp” transitions.
- This also results in blurry edges!

Smoothing Filters (3)

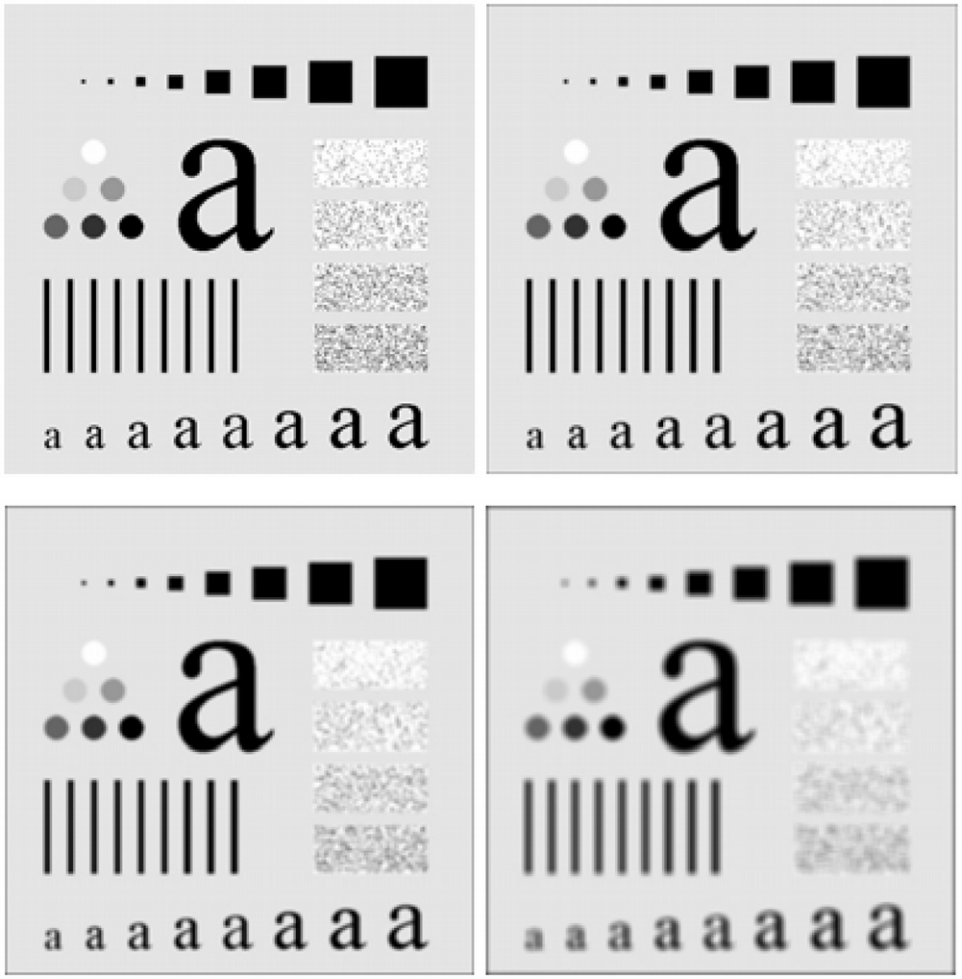
- Figure: two 3x3 smoothing masks.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Left filter: average. Right filter: weighted average.

Smoothing Filters (4)

- Figure: smoothing with averaging masks of sizes 3, 5, 9, and 11 pixels.
- Note the black border with high mask size: result of zero padding.
- Zero padding: expanding the image borders with 0's and then trimming off padded area.

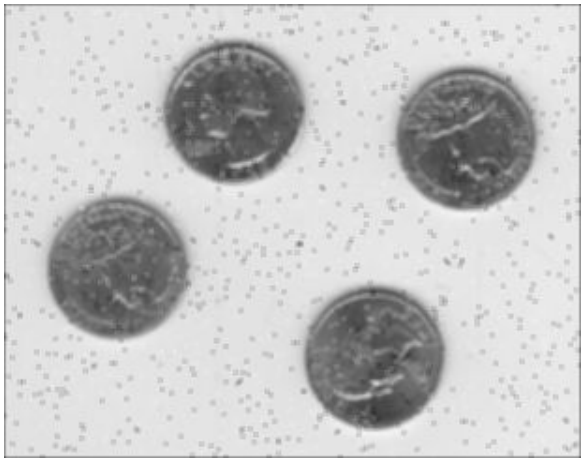


Smoothing Filters (5)

- Order-statistics filters: filters whose response is based on ordering the pixels contained in the filter area.
- Most common order-statistics filter: median filter.
- Median filters have excellent noise reduction capabilities, with less blurring.
- Example: given a 3x3 neighborhood with values (10, 20, 20, 20, 15, 20, 20, 25, 100)
  By sorting these values, the median is the 5th largest value, i.e. 20.

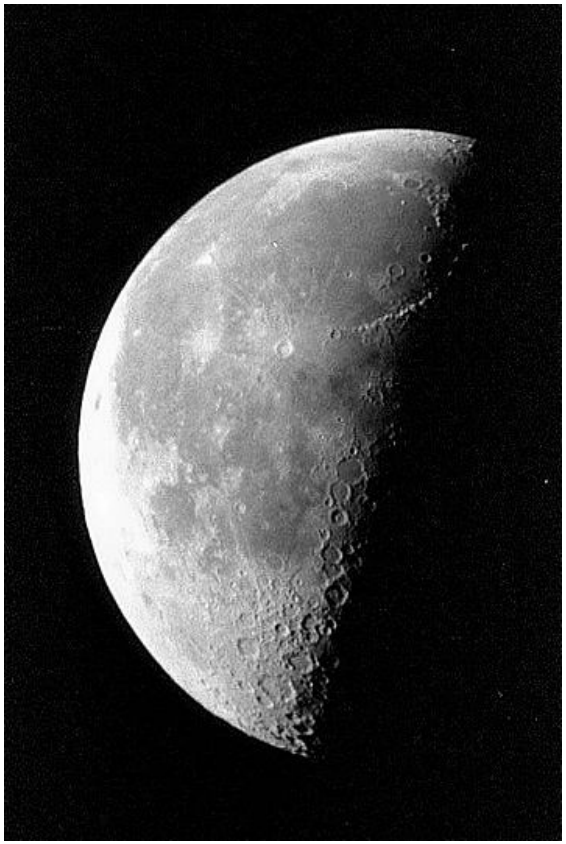
Smoothing Filters (6)

- Figure: Noisy image, average-filtered, median-filtered


Sharpening Filters (1)

- Objective: to highlight fine detail or enhance detail that has been blurred.



Sharpening Filters (2)

- Image sharpening can enhance edges and discontinuities (including noise).
- Since smoothing can be achieved by pixel summing, it is logical that sharpening can be achieved by differentiation.
- The Laplacian is the most common sharpening filter.
- The Laplacian is based on second-order difference:
  - First-order differenc (1-D): $f(x+1) - f(x)$
  - Second-order derivative (1-D): $f(x+1) + f(x-1) - 2f(x)$
- First-order difference are suitable for edge enhancement
- Second-order derivatives have the ability to enhance fine detail.

Sharpening Filters (3)

- The Laplacian highlights discontinuities and edges.
- Figure: different implementations of the Laplacian.

0	1	0
1	−4	1
0	1	0

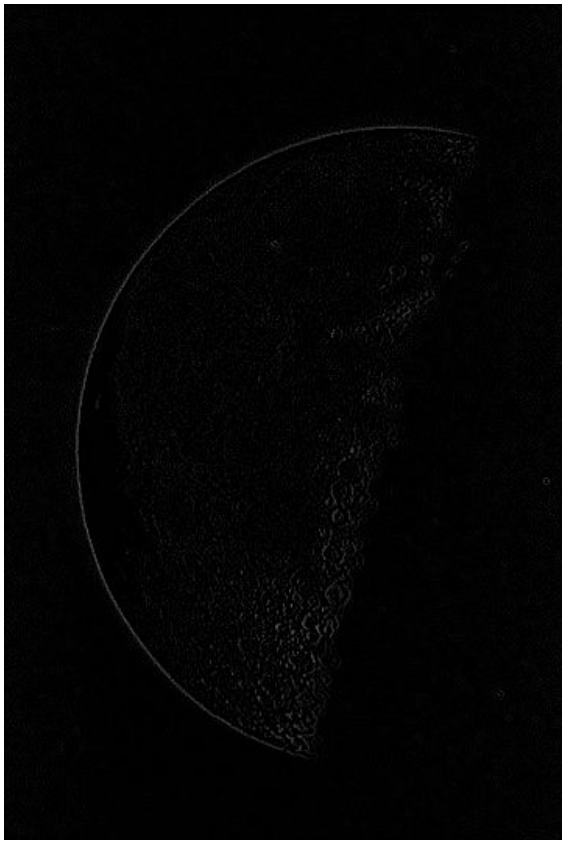
1	1	1
1	−8	1
1	1	1

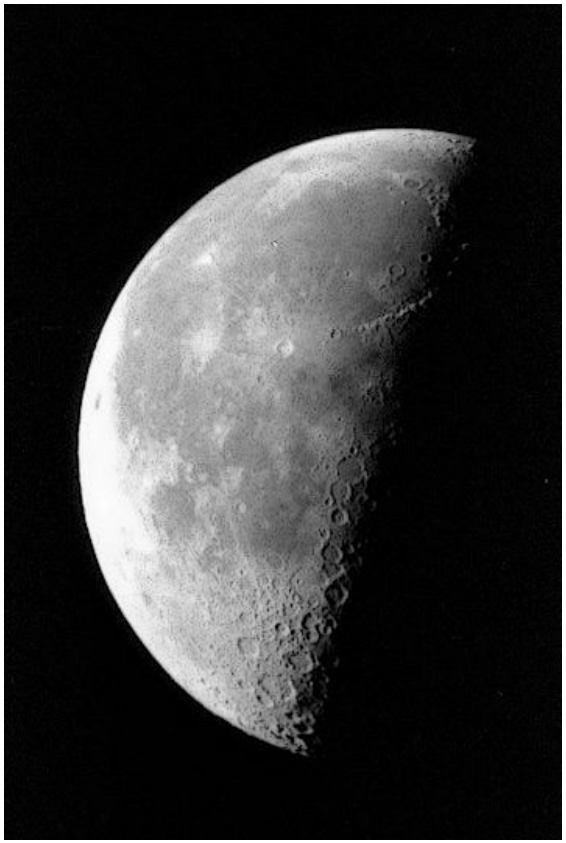
0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Sharpening Filters (4)

- For detail enhancement, add the Laplacian from the original image (if the center coefficient is positive).





Sharpening Filters (5)

- Instead of performing image enhancement in two steps (computing mask and then subtracting from image), this can be done in one step, with the following mask(s):

0	−1	0
−1	5	−1
0	−1	0

−1	−1	−1
−1	9	−1
−1	−1	−1

Sharpening Filters (6)

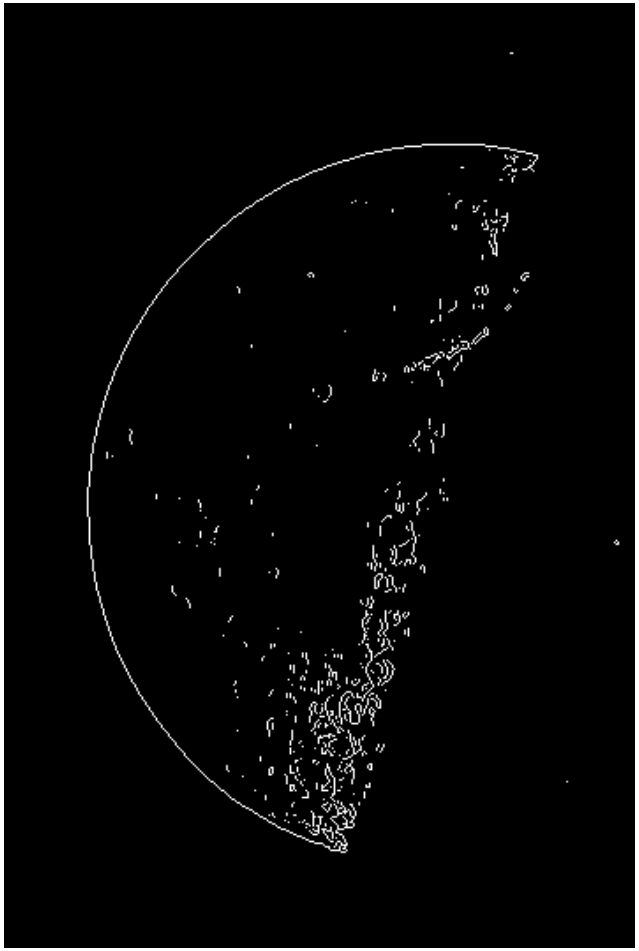
- Other sharpening method: using first derivatives
- Sobel masks (for horizontal and vertical edges):

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sharpening Filters (7)

- By combining output of vertical and horizontal filters, we can perform edge detection.
- Figure: edges computed using Sobel masks.



2-D Convolution (1)

- The process of moving a mask from pixel to pixel, and compute a quantity at each pixel, is the foundation of the convolution process.
- Given image $f[x,y]$ and mask $h[x,y]$, the 2-D convolution is defined as:

$$f[x,y]*h[x,y]=\sum_{m}^{M-1}\sum_{n}^{N-1} f[m,n]h[x-m,y-n]$$

- When given a filter for convolution, we need to rotate the mask 180 degrees

2-D Convolution (2)

Example:

Image

2	2	2
2	2	2
2	2	2

Mask

1	2	1
2	1	2
1	2	1

Darker gray area in mask denotes its centre.

Output

2	6	8	6	2
6	12	18	12	6
8	18	26	18	8
6	12	18	12	6
2	6	8	6	2

Darker gray area represents output if we crop to original image size.

2-D Convolution (3)

- As in 1-D case, the 2-D convolution is symmetric:

$$f[x,y] * h[x,y] = h[x,y] * f[x,y]$$

- In image processing, we usually compute convolutions in the frequency domain instead of the spatial domain (next topic)

Filtering in the Frequency Domain

Filtering in the Frequency Domain

- **Most image filtering** is done in the **frequency domain**
- Main reason: **computational efficiency**.
- Also: frequency transforms (and especially the Fourier transform) have strong mathematical foundations and properties that make image processing faster & easier.

1-D Discrete Fourier Transform

- In previous weeks, we saw the one-dimensional discrete Fourier transform (DFT):

$$X[k] = \sum x[n] e^{-i\frac{2\pi}{N}nk}$$

- X is the **spectrum** of **signal** x
- **Coefficients** of finite **combination** of **complex sinusoids**, ordered by frequencies
- Inverse Fourier transform returns original signal

2-D Discrete Fourier Transform (1)

- Extending the 1-D DFT into two dimensions is straightforward:

$$X[k,l] = \sum \sum x[m,n] e^{-i2\pi(km/M + ln/N)}$$

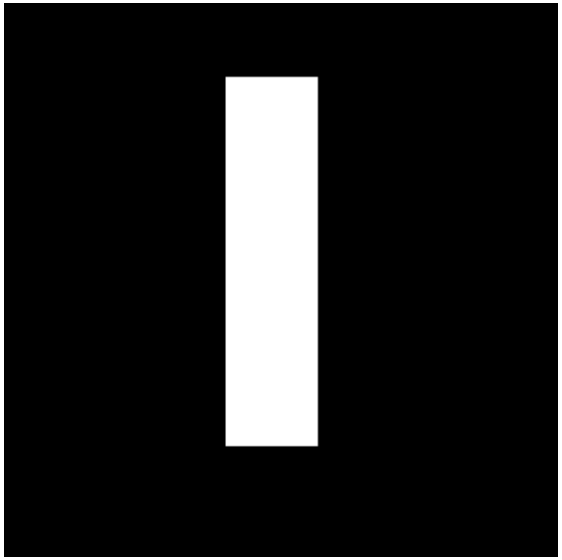
- Similarly, we can obtain the inverse 2-D DFT:

$$x[m,n] = \frac{1}{MN} \sum \sum X[k,l] e^{i2\pi(km/M + ln/N)}$$

2-D Discrete Fourier Transform (2)

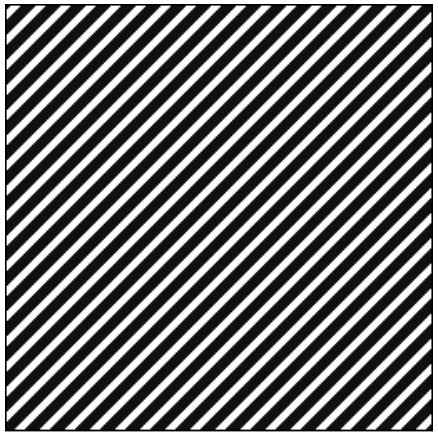
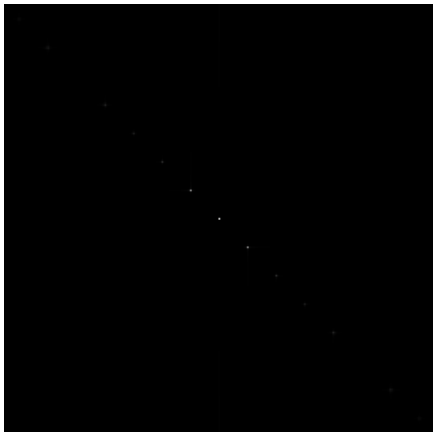
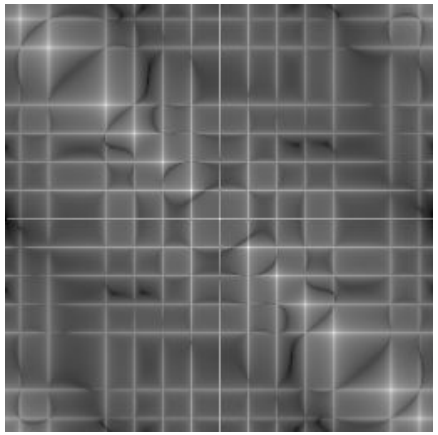
- As in the 1-D case, $X(k,l)$ is complex.
- $X(k,l)$ gives the magnitude (absolute) and phase (angle, arg) for each frequency in the continuum and is called the spectrum of $x[m,n]$.

Figure: rectangle and its DFT



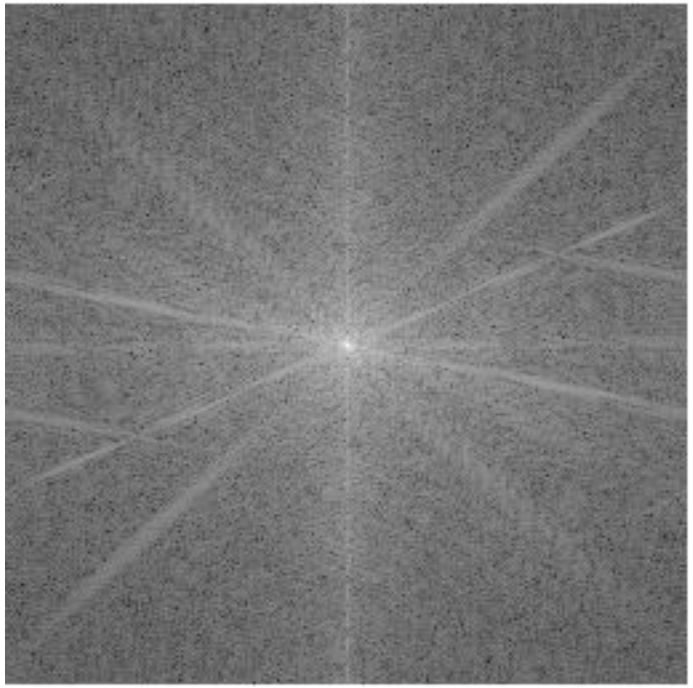
2-D Discrete Fourier Transform (3)

- It is not very easy to make direct associations between an image and its transform...
- But: we can make some general statements about the frequency components of an image. Figure: image, DFT, and logarithm of the DFT.

2-D Discrete Fourier Transform (4)

- A more realistic example:



2-D Discrete Fourier Transform (5)

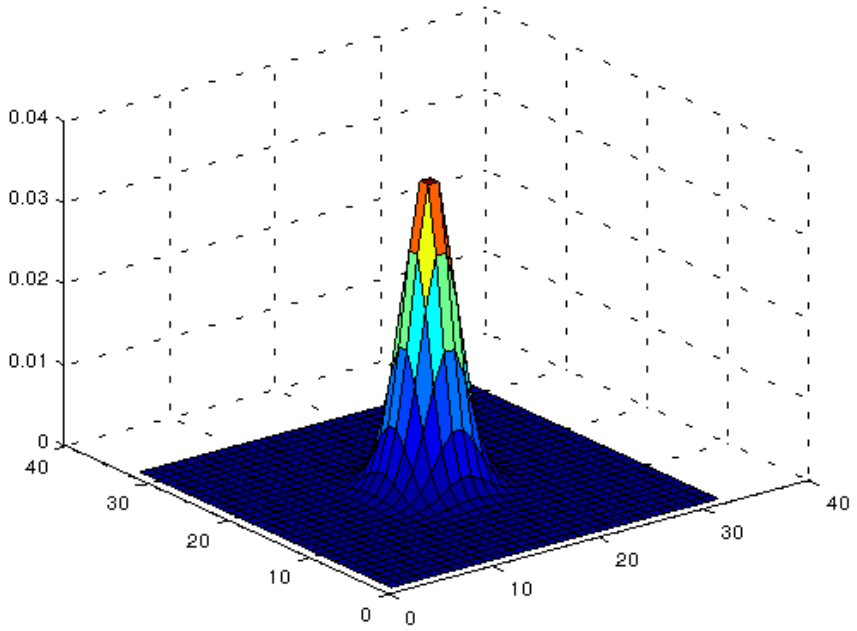
- Convolution Theorem: convolution in the spatial domain can be expressed by multiplication in the frequency domain:

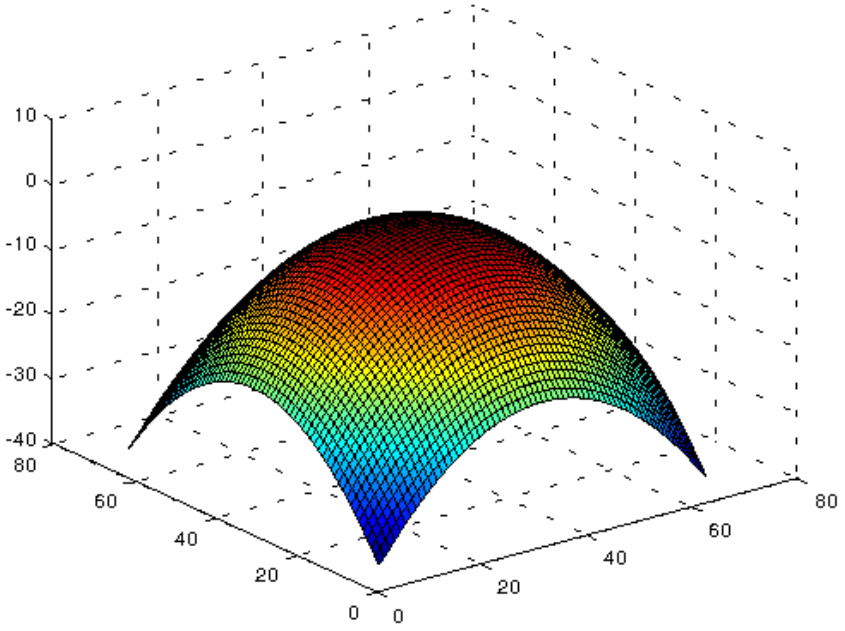
$$f[x,y] * h[x,y] \Leftrightarrow F[k,l]\,H[k,l]$$

- In practice when filtering, we compute the DFT of an image and its filter. We multiply them, and compute the inverse DFT.
- Fast Fourier Transform (FFT): fast implementation. O(M log M) instead of $O(M^2)$.

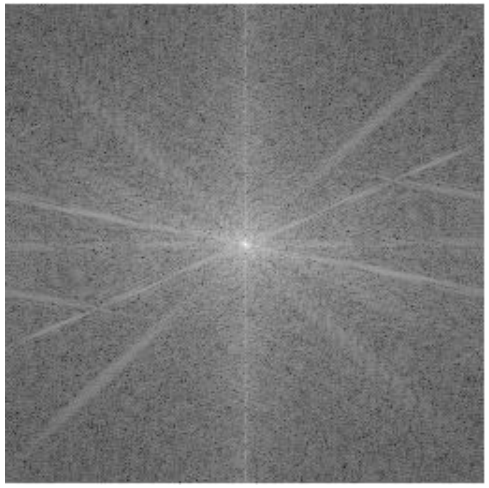
Frequency Domain Filters (1)

- Figure: Gaussian mask and its DFT (lowpass filter)





Frequency Domain Filters (2)



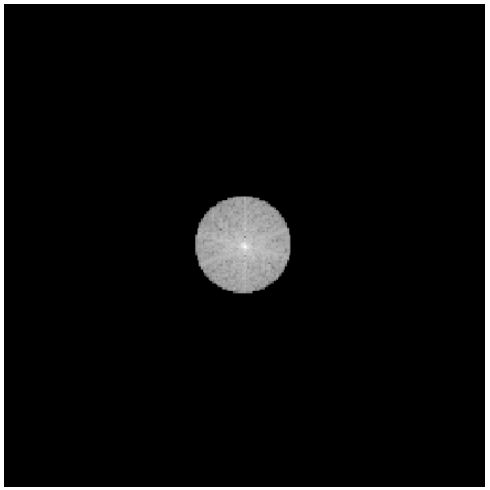


Figure: lowpass filtering example.

Frequency Domain Filters (3)

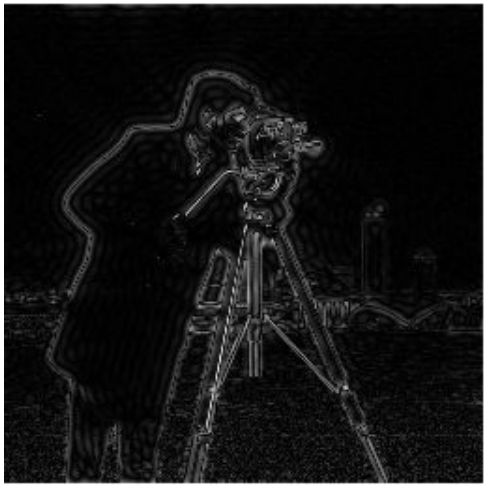
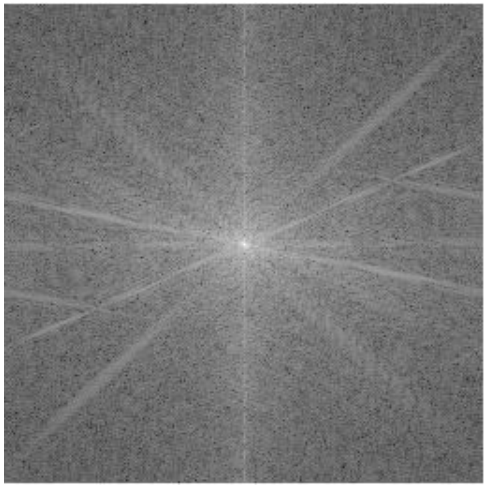
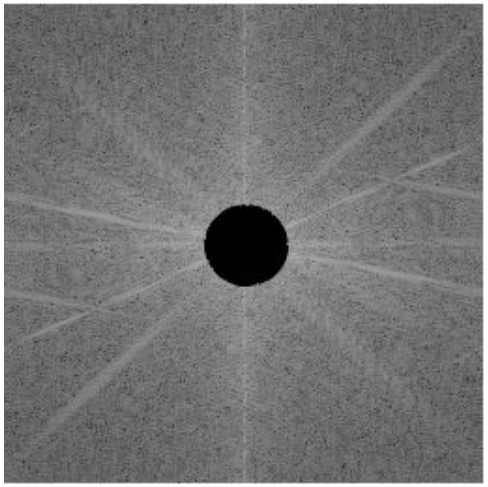


Figure: highpass filtering example.





Frequency Domain Filters (4)

- Smoothing frequency-domain filters (ordered from sharp to smooth cut-offs): ideal, Gaussian, Butterworth.

- Sharpening frequency-domain filters: same types as smoothing filters, typically their complement.

Image Filtering and Transforms in Matlab

Filtering and Transforms in Matlab (1)

- Define averaging filter:

```
h = ones(5,5) / 25;
```

- Filter an image:

```
J = imfilter(I,h);
```

- Use pre-existing filter (Gaussian, Laplacian, Sobel...):

```
h = fspecial(type);
```

- Compute frequency response for filter:

```
H = freqz2(h,n1,n2);  //e.g. n1,n2=256
```

Filtering and Transforms in Matlab (2)

- Perform median filtering:

```
J = medfilt2(I,[3 3]);
```

- 2-D FFT:

```
F = fft2(I);
```

- We typically visualise the logarithm of the magnitude:

```
F2 = log(abs(F)); //try imagesc for plotting
```

- Shift zero-frequency component to center of spectrum:

```
F2 = fftshift(F);
```

Filtering and Transforms in Matlab (3)

- Edge detection:

```
BW = edge(I,'sobel');
```

- Inverse 2-D FFT:

```
B = ifft2(F);
```

- 2-D convolution:

```
C = conv2(A,B);
```

Properties of the 2-D Fourier Transform

2-D DFT Properties (1)

- Similar properties with the 1-D Fourier Transform
- Linearity:

$$a\,x_1 + b\,x_2 \Leftrightarrow a\,X_1 + b\,X_2$$

- Scaling:

$$x[ak, bl] \Leftrightarrow \frac{1}{|ab|} X[k, l]$$

2-D DFT Properties (2)

- Separability:

$$X[k,l]=\frac{1}{M}\sum_{m=0}^{M-1} X[m,l]e^{-j2\pi lm/M}$$

- Where:

$$X[m,l]=\frac{1}{N}\sum_{n=0}^{N-1} x[m,n]e^{-j2\pi ln/N}$$

- In other words: we can compute the 2-D transform by first computing a 1-D transform over each row, and then computing a 1-D transform over each column.

2-D Correlation - Template Matching

2-D Correlation (1)

- As in the 1-D case, the correlation measures the similarity of two signals by adding their products at each sampled time.

- Definition of 2-D correlation:

$$x[m,n] \circ y[m,n] = \frac{1}{RS} \sum_{r=0}^{R-1} \sum_{S=0}^{S-1} x^*[r,s]\, y[m+r, n+s]$$

where * denotes complex conjugate (since we deal with real signals tough, $x^* = x$)

2-D Correlation (2)

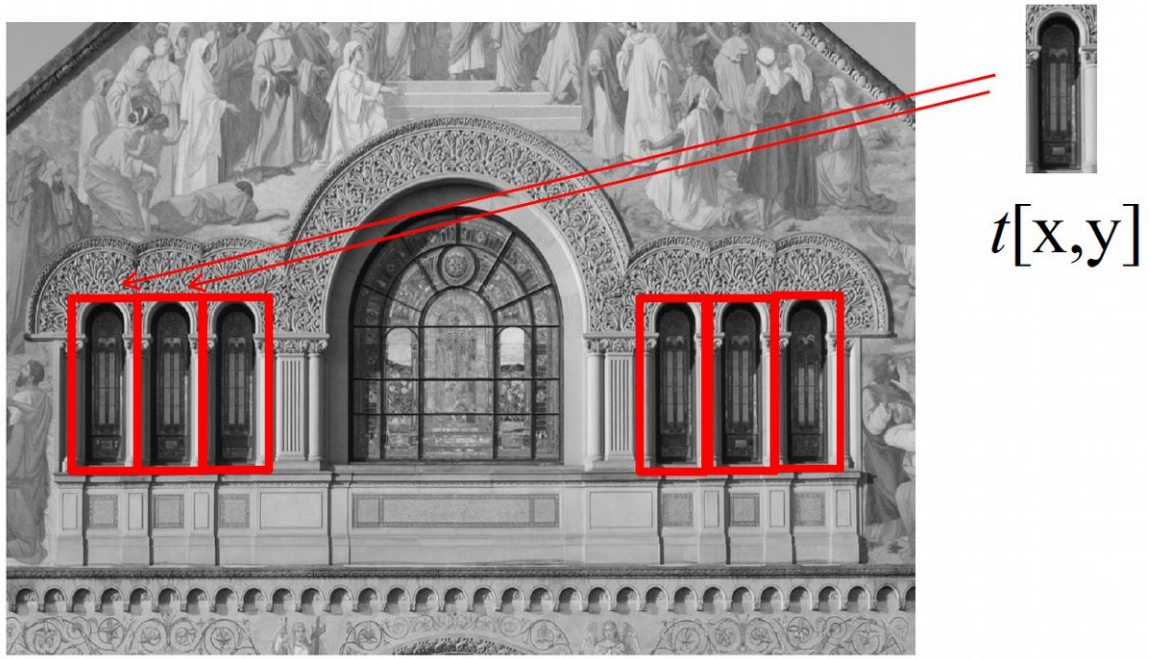
- The correlation is very similar to the convolution (with the exception of the conjugate and the positive sign)
- It can be implemented using convolution in Matlab:
  `conv2(a, rot90(conj(b),2))`

- The positive sign means that we don't mirror the mask about the origin.

- Correlation theorem:
  $x[m,n] \circ y[m,n] \Leftrightarrow X^*[k,l]Y[k,l]$

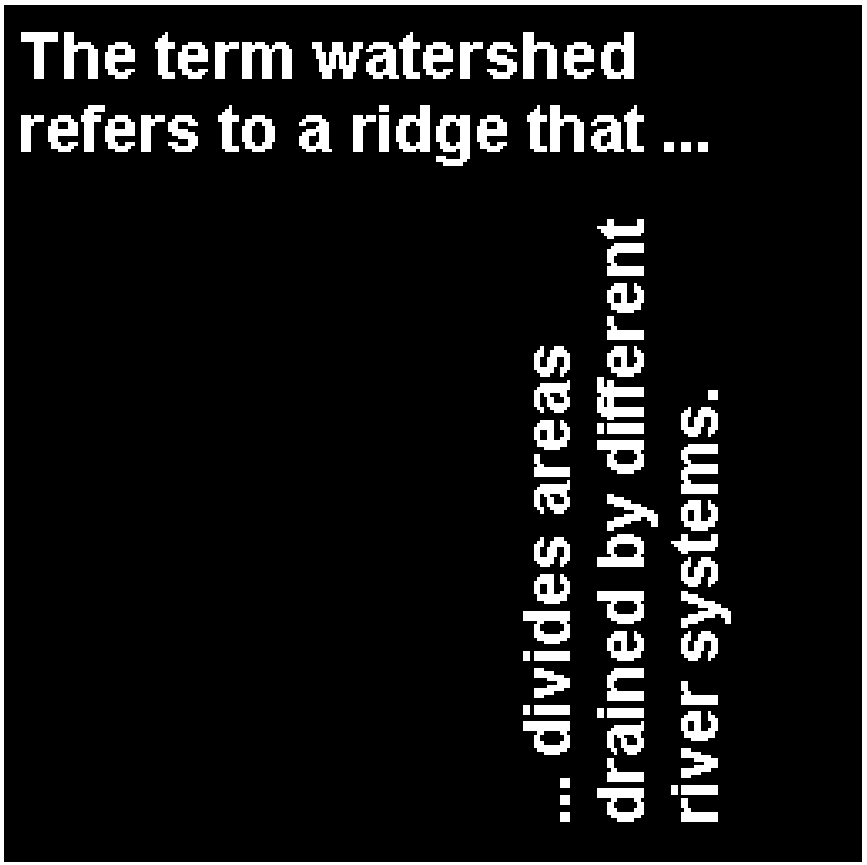
Template Matching (1)

- The principal use of correlation is for matching (i.e. locate an object)



Template Matching (2)

- Use of correlation for template matching:



a

- 2-D Correlation – Template Matching
- Directly done in Matlab with 2d cross-correlation:

```
C = xcorr2(A,B);
```

Template Matching (3)

- Left: 2-D correlation. Right: thresholded correlation.