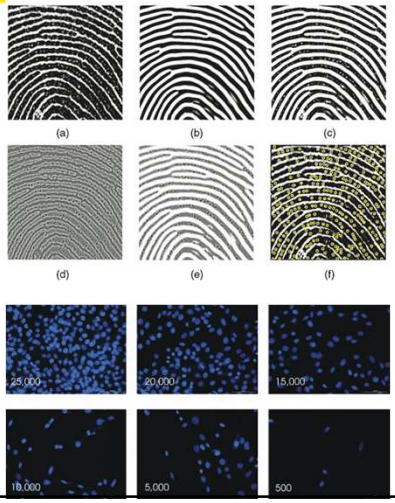
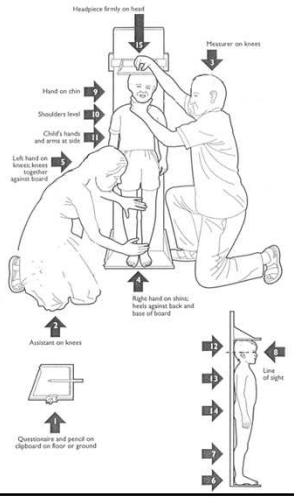
Graduate School of Biomedical Engineering

BIOM1010 - Image processing for physiological Measurement

Heba Khamis

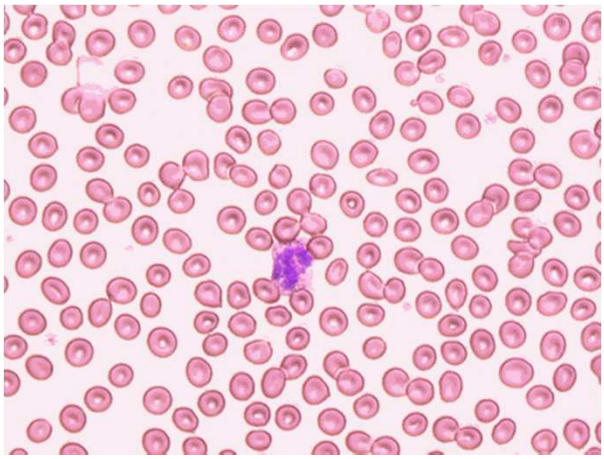




Motivation

- Need to extract and quantify objects and patterns in image data and obtain answers to meaningful biomedical questions

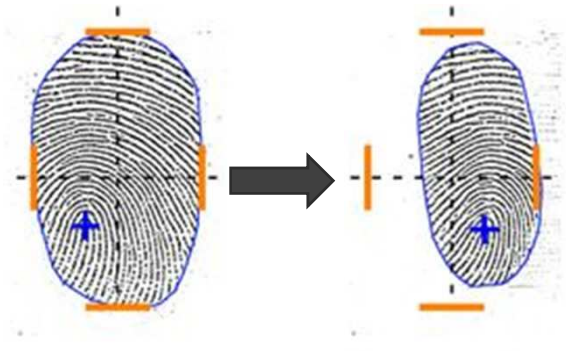
Motivation – Counting Cells



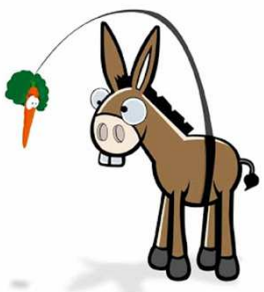
http://singularityhub.com/wp-content/uploads/2008/08/red-blood-cells.bmpood cells

UNSW SYDNEY

Motivation – Tracking Fingerprints



Delhaye, Benoit, Philippe Lefèvre, and Jean-Louis Thonnard. "Dynamics of fingertip contact during the onset of tangential slip." *Journal of The Royal Society Interface* 11.100 (2014): 20140698.



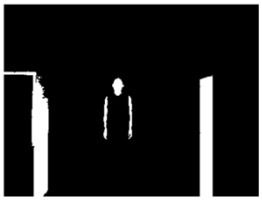
UNSW SYDNEY

Motivation – Malnutrition Assessment

(a) Original photo

(b) Face detection

(c) Human detection



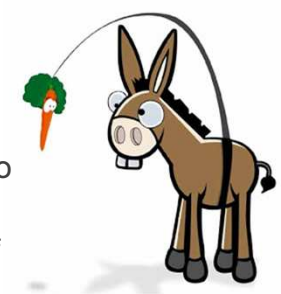
(d) Skin detection

(e) Foreground detection

UNSW SYDNEY

Advantages of Image Processing

- Human vision
  - highly sensitive
  - can be easily biased by pre-conceived notions of objects and concepts
- Automated image analysis provides an unbiased approach to extracting information from image data and testing hypotheses
- Image-analysis routine can be applied to a large number of images
  - facilitating the collection of large amounts of data for statistical analysis



UNSW SYDNEY

Lecture Overview

- What is a *digital* image
- Sources of images
- Sampling and Quantisation
- Resizing and Interpolation
- Complement and Logarithmic Transform
- Arithmetic and Logic
- Filtering and Sharpening
- Histograms, Contrast Stretching and Thresholding
- Morphological Operations
- Segmentation

- Explore some image processing techniques in MATLAB for analysing a wide range of biomedically relevant images
- Far-reaching applications in tissue engineering, haptics, and public health in developing countries, to name just a few

UNSW SYDNEY

Image vs. Digital Image

- Image
  - A 2D function f(x,y) where x and y are the spatial (plane) coordinates, and the amplitude value of f at any pair of coordinates (x,y) is called the intensity of the image at that position
- Digital Image
  - If x,y and the amplitude values of f are finite and discrete quantities, we call the image a digital image.
  - Composed of a finite number of elements called pixels
  - Each pixel has a particular location and value

UNSW SYDNEY

Colour vs. Grayscale

- Each pixel contains information about the image intensity at that point
  - Colour Digital Image
    - 3 channels
    - Red, Green, Blue (RGB) colour intensities
  - Grayscale Digital Image
    - 1 channel
    - Grey intensity (ranging from black to white)
  - Usually each intensity is stored as an 8-bit binary number (values 0 – 255)

Example of a Digital Image

- 12 x 12 pixel grayscale image

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 0 \\
0 & 255 & 255 & 255 & 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 \\
0 & 255 & 255 & 0 & 191 & 191 & 191 & 191 & 0 & 255 & 255 & 0 \\
0 & 255 & 0 & 191 & 0 & 191 & 191 & 0 & 191 & 0 & 255 & 0 \\
0 & 255 & 0 & 191 & 191 & 191 & 191 & 191 & 191 & 0 & 255 & 0 \\
0 & 255 & 0 & 191 & 0 & 191 & 191 & 0 & 191 & 0 & 255 & 0 \\
0 & 255 & 0 & 191 & 191 & 0 & 0 & 191 & 191 & 0 & 255 & 0 \\
0 & 255 & 255 & 0 & 191 & 191 & 191 & 191 & 0 & 255 & 255 & 0 \\
0 & 255 & 255 & 255 & 0 & 0 & 0 & 0 & 255 & 255 & 255 & 0 \\
0 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Example of a Digital Image

UNSW SYDNEY

Example of a Digital Image

Pixel intensity value

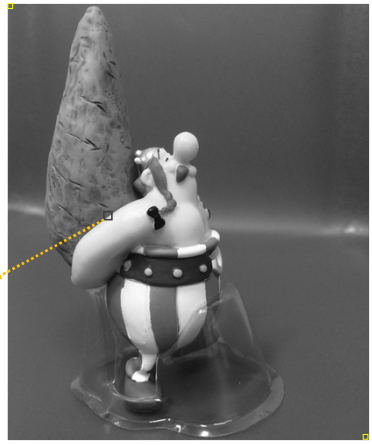
f(1,1) = 103

Pixel location

rows columns

f(645:650,1323:1328) =

```
83 82 82 82 82 82
82 82 82 81 81 81
82 82 81 81 80 80
82 82 81 80 80 79
80 79 78 77 77 77
80 79 78 78 77 77
```



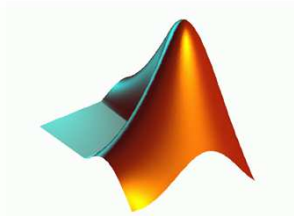
Consider the following image (2724x2336 pixels) to be 2D function or a matrix with rows and columns

In 8-bit representation Pixel intensity values change between 0 (Black) and 255 (White)

f(2724,2336) = 88

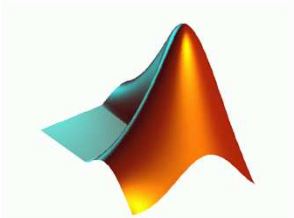
UNSW SYDNEY

MATLAB – Image Representation



- Colour image in MATLAB is:
  - N×M×3 matrix
    - N rows (height)
    - M columns (width)
    - 3 colour intensities - Red, Green, Blue values
- Grayscale image in MATLAB is:
  - N×M×3 matrix
    - where all 3 colour intensity values are equal for a given pixel
  - Or, N×M×1

MATLAB – Reading and Displaying an Image

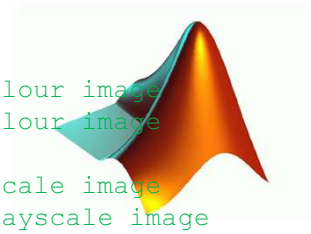


`imread()` – reading an image with different postfixes

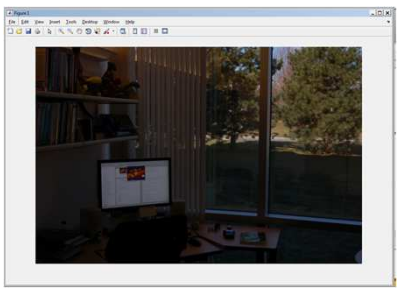
`figure` – opening a new graphical window

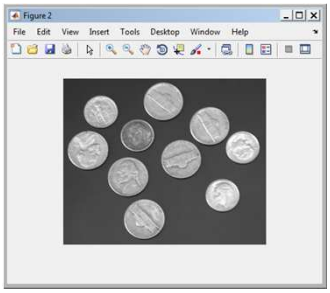
`imshow()` – displaying an image

MATLAB - Example



```
I = imread('office_2.jpg');% read colour image
figure; imshow(I);         % display colour image

J = imread('coins.png');% read grayscale image
figure; imshow(J);         % display grayscale image
```

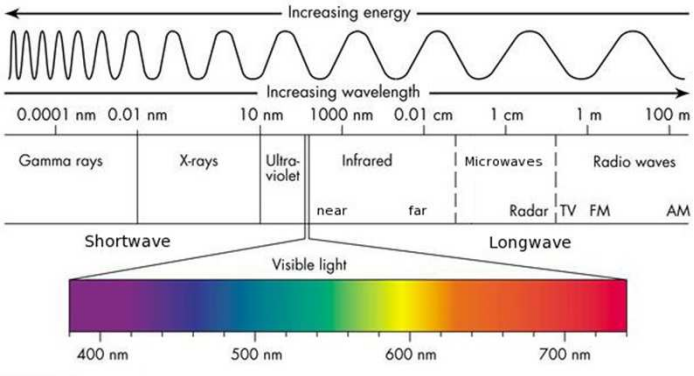




UNSW SYDNEY

Sources of Digital Images

- The principal source for the images is the **electromagnetic** (EM) **energy** spectrum.



UNSW SYDNEY

Gamma Rays

Gamma-Ray Imaging
İn nuclear medicine

Gamma-Ray imaging of
A starburst galaxy
about 12 million light-
years away

X-Rays

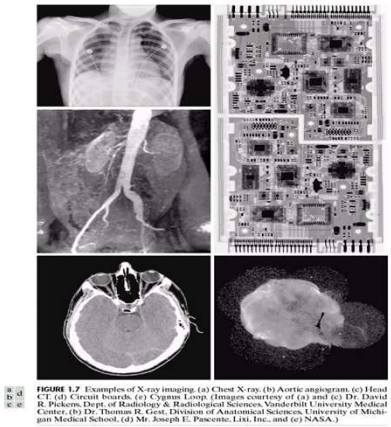
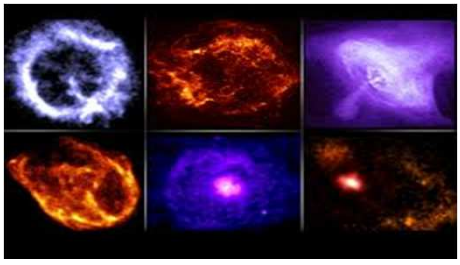


FIGURE 1.7 Examples of X-ray imaging. (a) Chest X-ray. (b) Aortic angiogram. (c) Head CT. (d) Circuit boards. (e) Cygnus Loop (Images courtesy of (a) and (c) Dr. David R. Pickens, Dept. of Radiology & Radiological Sciences, Vanderbilt University Medical Center, (b) Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, (d) Mr. Joseph E. Pascente, Lixi, Inc., and (e) NASA.)

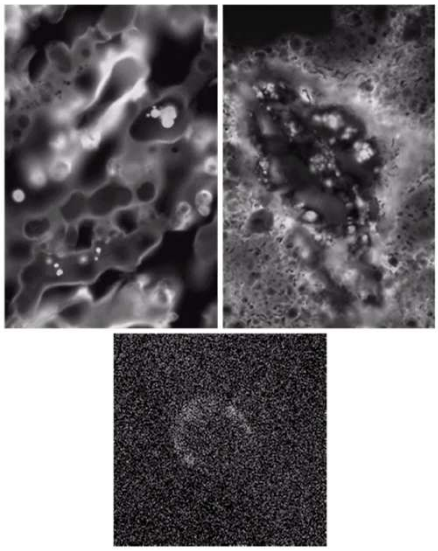


X-ray images from the space
The Chandra X-Ray Observatory

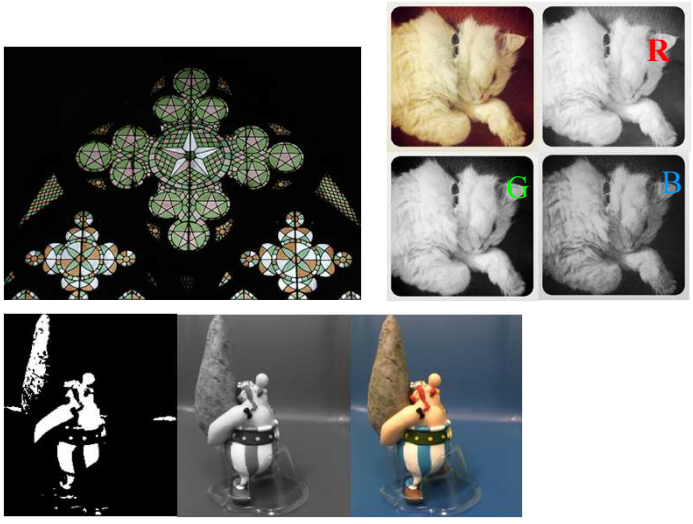
Ultra Violet

a b
c

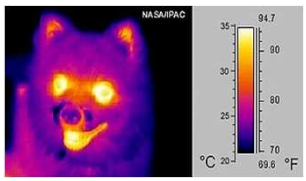
FIGURE 1.8
Examples of ultraviolet imaging.
(a) Normal corn.
(b) Smut corn.
(c) Cygnus Loop.
(Images courtesy of (a) and (b) Dr. Michael W. Davidson, Florida State University, (c) NASA.)



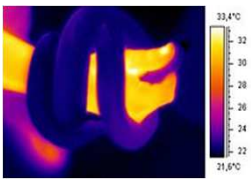
Visible Light



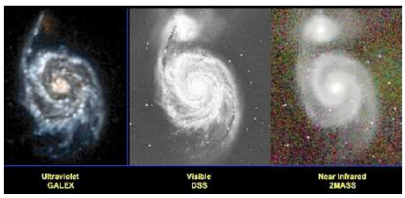
Infrared



infrared ("thermal") image



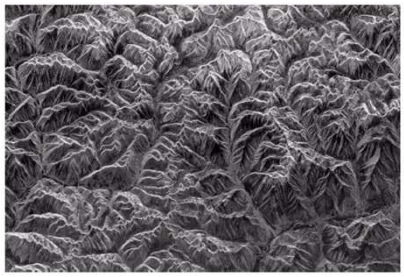
Snake around the arm

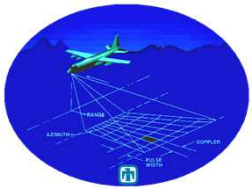


Messier 51 in ultraviolet (GALEX), visible (DSS), and near infrared (2MASS). *Courtesy of James Fanson.*

Microwave

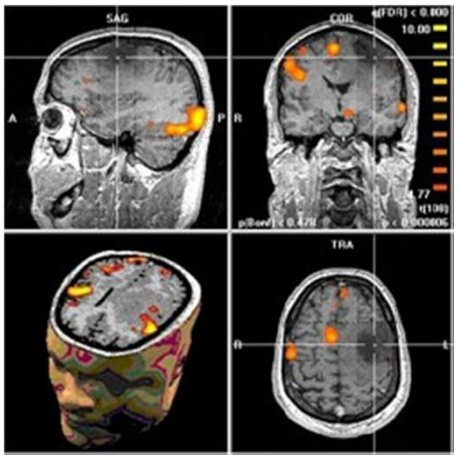
FIGURE 1.16
Spaceborne radar image of mountains in southeast Tibet. (Courtesy of NASA.)





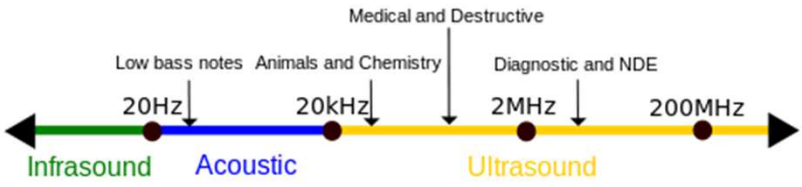
Synthetic Aperture Radar System

Radiowave



MRI image slices from the brain

Ultrasound imaging (non-electromagnetic image source)



Ultrasonic spectrum

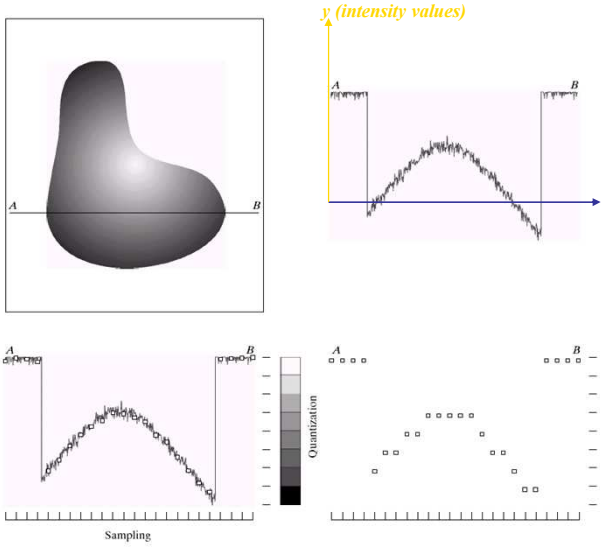
Ultrasonic Baby image during pregnancy

Ultrasound image acquisition device

Sampling and Quantisation

- In order to become suitable for digital processing, an image function f(x,y) must be digitised both spatially (sampling) and in amplitude (quantisation).
- The sampling rate determines the spatial resolution of the digitised image, while the quantisation level determines the number of grey levels in the digitised image.
- The transition between continuous values of the image function and its digital equivalent is called quantisation.

Sampling and Quantisation



a b
c d

Generating a digital image.

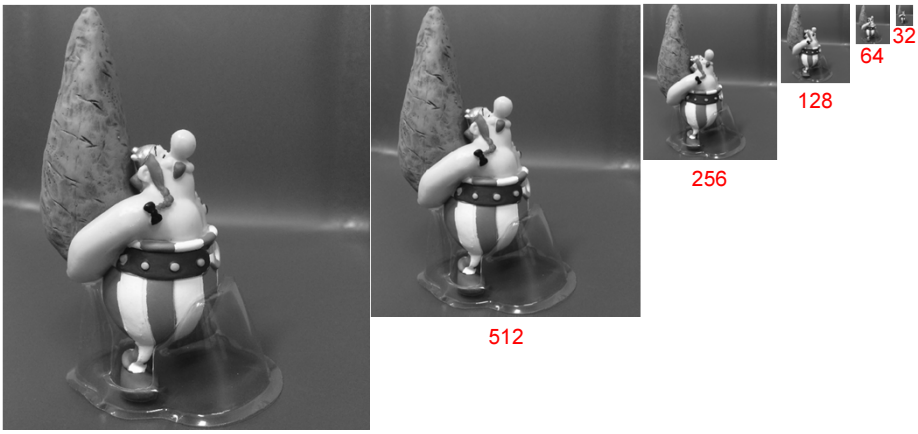
(a) Continuous image.

(b) A scaling line from A to B in the continuous image, used to illustrate the concepts of sampling and quantisation.

(c) sampling and quantisation.

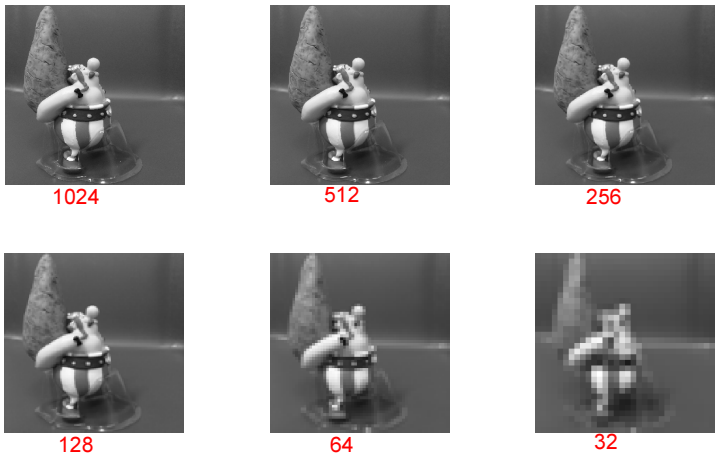
(d) Digital scan line.

Sampling



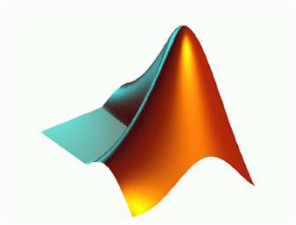
UNSW SYDNEY

Sampling



UNSW SYDNEY

MATLAB - Sampling



```
% read colour image
imC=imread('peppers.png');

% convert image to grayscale
imG=rgb2gray(imC);

% resize image
im1=imresize(imG, [1024 1024]); % http://au.mathworks.com/help/images/ref/imresize.html
im2=imresize(im1, [1024 1024]/2);
im3=imresize(im1, [1024 1024]/4);
im4=imresize(im1, [1024 1024]/8);
im5=imresize(im1, [1024 1024]/16);

% display images
figure;imshow(im1);
figure;imshow(im2);
figure;imshow(im3);
figure;imshow(im4);
figure;imshow(im5);
```

UNSW SYDNEY

Quantisation


8-bit


7-bit


6-bit


5-bit


4-bit

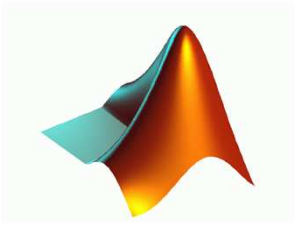

3-bit


2-bit


1-bit

UNSW SYDNEY

MATLAB - Quantisation

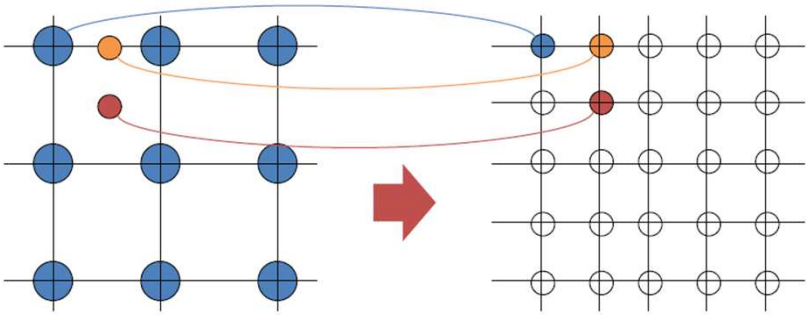


```
% read colour image
imC = imread('peppers.png');

% convert image to grayscale
imG = rgb2gray(imC);

% requantise image
[im2,map2] = gray2ind(imG,2^7); % http://au.mathworks.com/help/images/ref/gray2ind.html
[im3,map3] = gray2ind(imG,2^6);
[im4,map4] = gray2ind(imG,2^5);
[im5,map5] = gray2ind(imG,2^4);

% display images
figure;imshow(imG);
figure;imshow(im2,map2);
figure;imshow(im3,map3);
figure;imshow(im4,map4);
figure;imshow(im5,map5);
```


Resizing an Image

- Zooming / Up-sampling
  - To blow up (increase the size of) a small image (or sub-image)
  - Results in poor resolution
  - Creating new pixel locations
  - Assigning gray-level values to these locations
    - Interpolation!
- Example of bulls$*#t:
  https://www.youtube.com/watch?v=3uoM5kfZIQ0

Resizing an Image



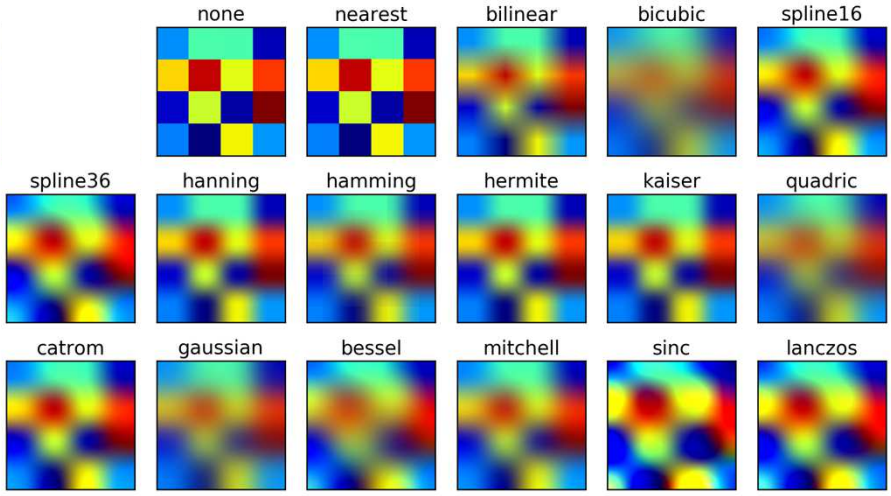
- We need to assign values to each pixel in the output image
- So scan through the output image; at each pixel calculate the value from the input image at the corresponding location
- If not at an integer position in the input image, we need to interpolate

Interpolation Methods

Yang CS, Kao SP, Lee FB, Hung PS. 2004. **Twelve Different Interpolation Methods: a Case Study of Surfer 8.0.** International Society for Photogrammetry and Remote Sensing; 778–783.

1. Inverse Distance to a Power
2. Kriging
3. Minimum Curvature
4. Modified Shepard's
5. Natural Neighbor
6. Nearest Neighbor
7. Polynomial Regression
8. Radial Basis Function Interpolation
9. Triangulation with Linear Interpolation
10. Moving Average
11. Data Metrics
12. Local Polynomial

Interpolation Methods



UNSW SYDNEY

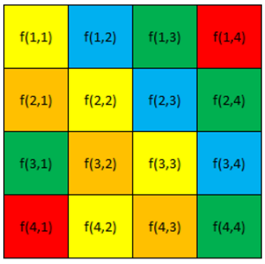
Interpolation Methods

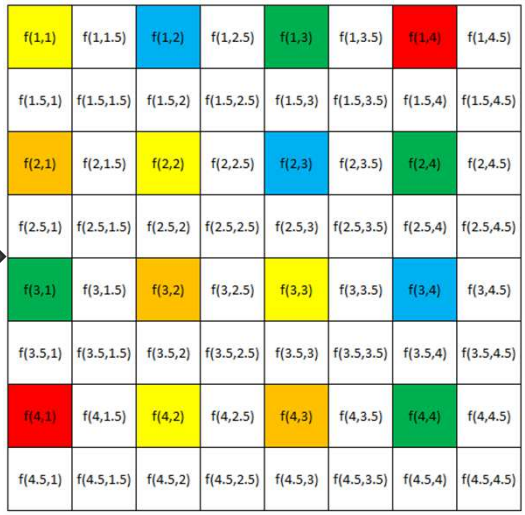
- 3 main types of 2D Interpolations:
  - Nearest neighbor interpolation
  - Bilinear interpolation
  - Bicubic interpolation

UNSW SYDNEY

Nearest Neighbour Interpolation

- Example:
  expand a 4x4 image to 8x8



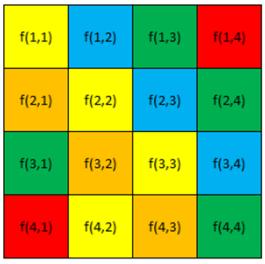


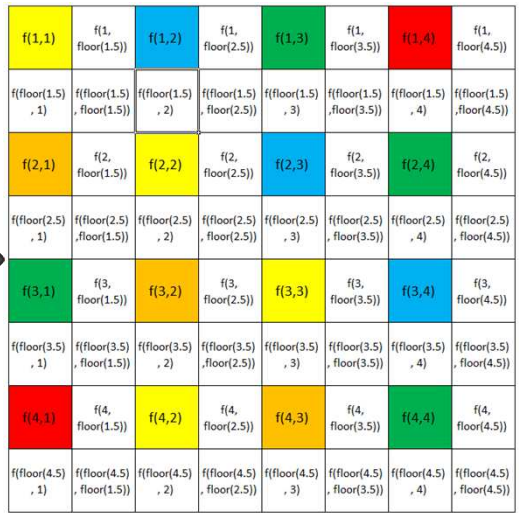
- Unknown pixel values:
  f(x,y) = f(floor(x),floor(y))

UNSW SYDNEY

Nearest Neighbour Interpolation

- Example:
  expand a 4x4 image to 8x8



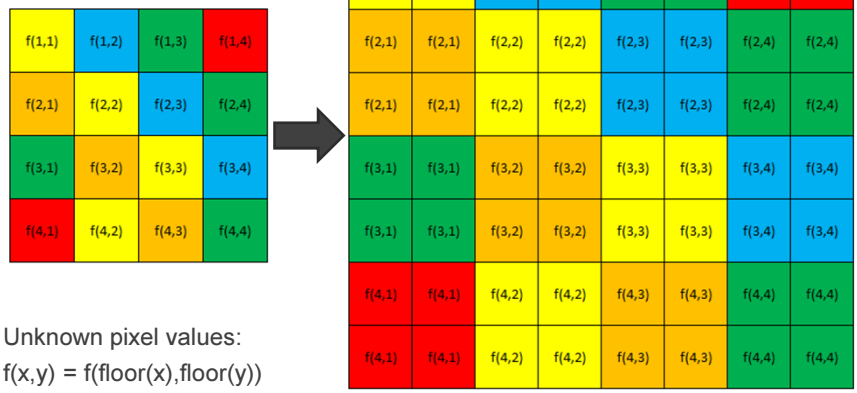


- Unknown pixel values:
  f(x,y) = f(floor(x),floor(y))

UNSW SYDNEY

Nearest Neighbour Interpolation

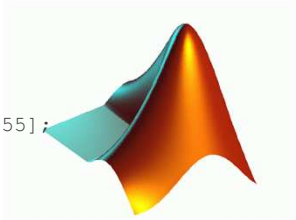
- Example:
  expand a 4x4 image to 8x8



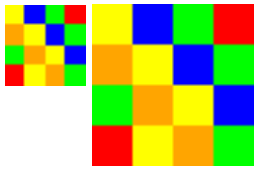
- Unknown pixel values:
  f(x,y) = f(floor(x),floor(y))

UNSW SYDNEY

MATLAB – Resizing – Nearest Neighbour



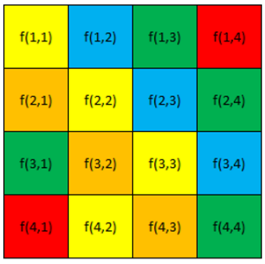
```
% define colours
r = [255 0 0];     g = [0 255 0];     b = [0 0 255];
y = [255 255 0];   o = [255 165 0];

% setup image
I = zeros(4,4,3);
I(1,:,:) = [y; b; g; r];
I(2,:,:) = [o; y; b; g];
I(3,:,:) = [g; o; y; b];
I(4,:,:) = [r; y; o; g];
I = uint8(I);

% display image
figure; imshow(I);

% resize image using Nearest Neighbour Interpolation
J = imresize(I,2,'nearest');

% http://au.mathworks.com/help/images/ref/imresize.html
```

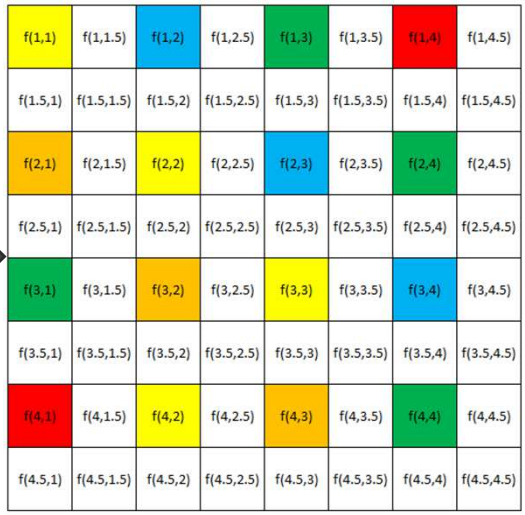


UNSW SYDNEY

Bilinear Interpolation

- Example:
  expand a 4x4 image to 8x8





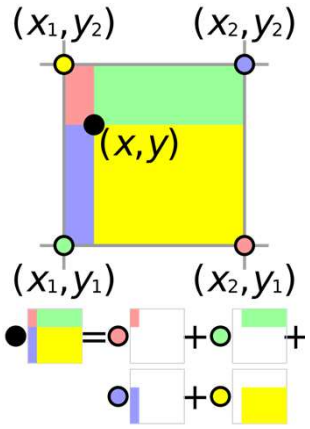
- Unknown pixel values:
  f(x,y) = the distance weighted average of the four nearest pixel values

UNSW SYDNEY

Bilinear Interpolation

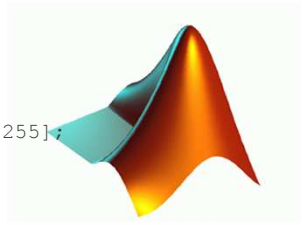
- $\frac{f(x,y)}{(x_2-x_1)(y_2-y_1)} =$
  $f(x_1,y_2)(x-x_1)(y-y_2) +$
  $f(x_2,y_2)(x-x_2)(y-y_2) +$
  $f(x_1,y_1)(x-x_1)(y-y_1) +$
  $f(x_2,y_1)(x-x_2)(y-y_1)$

- The value at the black spot is the sum of the value at each coloured spot multiplied by the area of the rectangle of the same colour, divided by the total area of all four rectangles

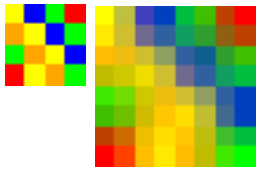


UNSW SYDNEY

MATLAB – Resizing – Bilinear

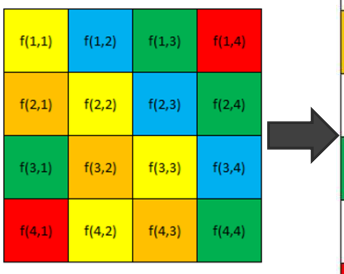


```
% define colours
r = [255 0 0];     g = [0 255 0];     b = [0 0 255];
y = [255 255 0];   o = [255 165 0];

% setup image
I = zeros(4,4,3);
I(1,:,:) = [y; b; g; r];
I(2,:,:) = [o; y; b; g];
I(3,:,:) = [g; o; y; b];
I(4,:,:) = [r; y; o; g];
I = uint8(I);

% display image
figure; imshow(I);

% resize image using Bilinear Interpolation
J = imresize(I,2,'bilinear');

% http://au.mathworks.com/help/images/ref/imresize.html
```



Bicubic Interpolation

- Example:
  expand a 4x4 image to 8x8



f(1,1)	f(1,1.5)	f(1,2)	f(1,2.5)	f(1,3)	f(1,3.5)	f(1,4)	f(1,4.5)
f(1.5,1)	f(1.5,1.5)	f(1.5,2)	f(1.5,2.5)	f(1.5,3)	f(1.5,3.5)	f(1.5,4)	f(1.5,4.5)
f(2,1)	f(2,1.5)	f(2,2)	f(2,2.5)	f(2,3)	f(2,3.5)	f(2,4)	f(2,4.5)
f(2.5,1)	f(2.5,1.5)	f(2.5,2)	f(2.5,2.5)	f(2.5,3)	f(2.5,3.5)	f(2.5,4)	f(2.5,4.5)
f(3,1)	f(3,1.5)	f(3,2)	f(3,2.5)	f(3,3)	f(3,3.5)	f(3,4)	f(3,4.5)
f(3.5,1)	f(3.5,1.5)	f(3.5,2)	f(3.5,2.5)	f(3.5,3)	f(3.5,3.5)	f(3.5,4)	f(3.5,4.5)
f(4,1)	f(4,1.5)	f(4,2)	f(4,2.5)	f(4,3)	f(4,3.5)	f(4,4)	f(4,4.5)
f(4.5,1)	f(4.5,1.5)	f(4.5,2)	f(4.5,2.5)	f(4.5,3)	f(4.5,3.5)	f(4.5,4)	f(4.5,4.5)

- Unknown pixel values:
  f(x,y) = the distance weighted average of the sixteen nearest pixel values

Nearest Neighbour, Bilinear, Bicubic

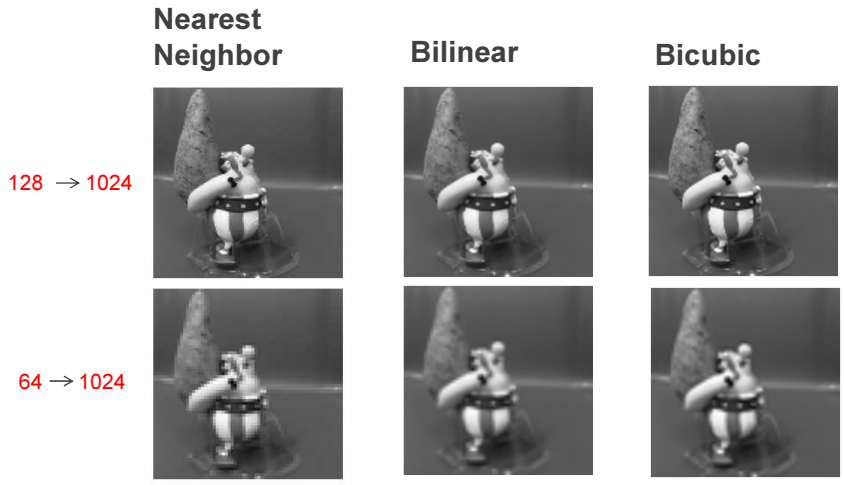
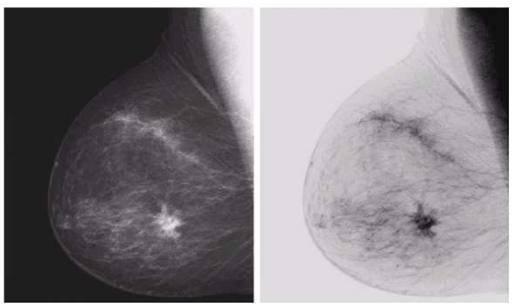


Image Complement/Negative

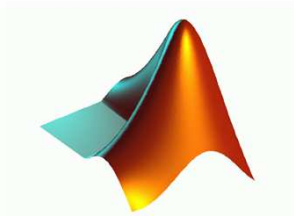
- $Q(i, j) = L - P(i, j)$

  Where L is the maximum pixel value that the image type allows



(left) Original digital mammogram. (right) Negative image obtained using the negative transformation

MATLAB – Image Complement

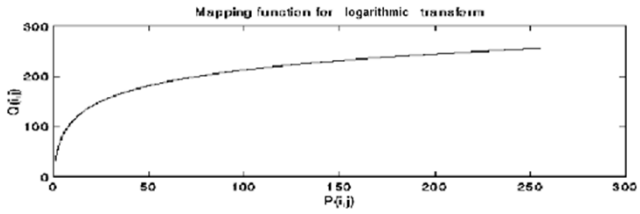


```
% read grayscale image
I = imread('coins.png');

% display image
figure; imshow(I);

% compute image complement
J = imcomplement(I); % http://au.mathworks.com/help/images/ref/imcomplement.html

% display image
figure; imshow(J);
```

UNSW SYDNEY

Logarithmic Transform

- $Q(i,j) = c\log(1 + |P(i,j)|)$
- 1+ is because log(0) is undefined
- c is a constant scaling factor such that $Q(i,j) = 255$ for $|P(i,j)| =$ maximum input value
- log can be any base



UNSW SYDNEY

Logarithmic Transform

- The logarithmic operator enhances the low intensity pixel values, while compressing high intensity values into a relatively small pixel range.
- Hence, if an image contains some important high intensity information, applying the logarithmic operator might lead to loss of information.

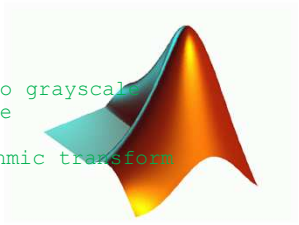
Logarithmic Transform

- Appropriate for enhancing low pixel values at the expense of loss of information in the high pixel values
- E.g.

- Photographed in front of a bright background
- Film material has low dynamic range
  - ➢ graylevels on the subject's face are clustered in a small pixel value range.
- A logarithmic transform spreads face pixels over a wider range, while higher values are compressed

- Less appropriate in images that contain a lot of details in the high pixel values.

MATLAB – Logarithmic Transform



```
I = imread('office_2.jpg');% read colour image
P = rgb2gray(I);       % convert colour image to grayscale
figure; imshow(P);     % display grayscale image

% iterate through each pixel and apply logarithmic transform
C = 125;
[M,N] = size(P);
for x = 1:M
        for y = 1:N
                m = double(P(x,y));
                Q(x,y) = C.*log10(1+m);
        end
end

figure; imshow(uint8(Q));      % display transformed image
```

UNSW SYDNEY

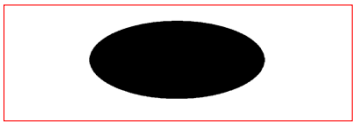
Basic Logic Operations

Operation	Definition
NOT	c = $\bar{a}$
OR	c = $a + b$
AND	c = $a.b$
XOR	c = $a \oplus b = a.\bar{b} + \bar{a}.b$

UNSW SYDNEY

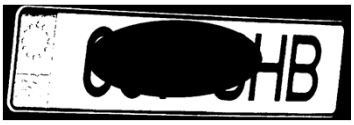
Logic Operations

a



b

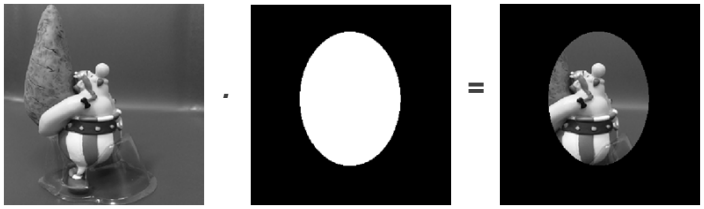
NOT(a)

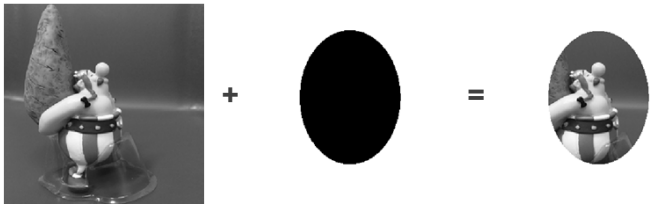


a . b

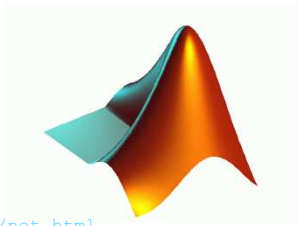
a + b

Logic Operations





MATLAB – Logic

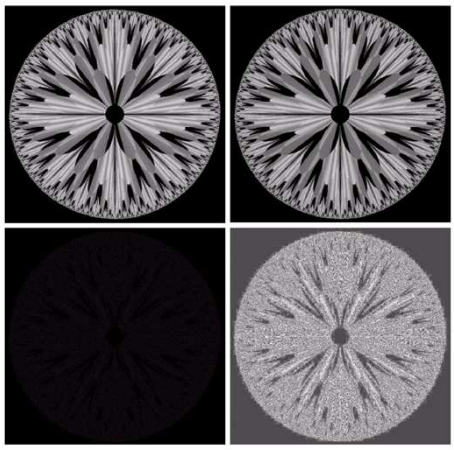


```
I = logical([0 0 1; 1 1 1]);
% I =     0 0 1
%         1 1 1
J = logical([1 0 1; 0 1 1]);
% J =     1 0 1
%         0 1 1

K = ~I;
K = not(I); % http://au.mathworks.com/help/matlab/ref/not.html
% K =     1 1 0
%         0 0 0

K = I | J;
K = or(I,J); % http://au.mathworks.com/help/matlab/ref/or.html
% K =     1 0 1
%         1 1 1

K = I & J;
K = and(I,J); % http://au.mathworks.com/help/matlab/ref/and.html
% K =     0 0 1
%         0 1 1

K = xor(I,J); % http://au.mathworks.com/help/matlab/ref/xor.html
% K =     1 0 0
%         1 0 0
```


Basic Arithmetic Operations

Operation	Definition
ADD	c = a + b
SUB	c = a - b
MUL	c = a * b
DIV	c = a / b
LOG	c = log(a)
EXP	c = exp(a)
SQRT	c = sqrt(a)
TRIG.	c = sin/cos/tan(a)
INVERT	c = $(2^B - 1) - a$

Image Subtraction

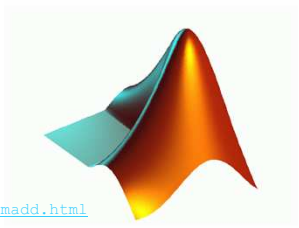


a b
c d

(a) original fractal image.
(b) Result of setting the four lower-order bit planes to zero.
(c) Difference between (a) and (b) .
(d) Histogram equalized difference image.

UNSW SYDNEY

MATLAB – Arithmetic



```
X = uint8([ 255 0 75; 44 225 100]);
% X =    255 0 75
%        44 255 100
Y = uint8([ 50 50 50; 50 50 50 ]);
% Y =    50 50 50
%        50 50 50

Z = imadd(X,Y); % http://au.mathworks.com/help/images/ref/imadd.html
% Z =    255 50 125
%        94 255 150

Z = imsubtract(X,Y); % http://au.mathworks.com/help/images/ref/imsubtract.html
% Z =    205 0 25
%        0 175 50

Z = imdivide(X,Y); % http://au.mathworks.com/help/images/ref/imdivide.html
% Z =    5 1 2
%        1 5 2

I = imread('moon.tif');
I16 = uint16(I);
J = immultiply(I16,I16); % http://au.mathworks.com/help/images/ref/immultiply.html
figure; imshow(I);
figure; imshow(J);
```

UNSW SYDNEY

Filtering

- An image may be “dirty” (with dots, speckles, stains)
- We want to remove noise from the image or enhance the image
- Filtering is a *neighbourhood operation,* in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighbourhood of the corresponding input pixel.
  - A pixel's neighbourhood is some set of pixels, defined by their locations relative to that pixel.

Filtering

- Noise removal: To remove speckles/dots on an image
  - Can be removed by taking mean or median values of neighboring pixels (e.g. 3x3 window)
  - Equivalent to low-pass filtering
- Problem with low-pass filtering
  - May blur edges
  - More advanced techniques: adaptive, edge preserving

Mean/Average Filter

- A type of linear filtering
- Replace each pixel by the average of pixels in a square window surrounding this pixel
- Smooths the image by averaging out sharp transitions
- Trade-off between noise removal and detail preserving:
  - Larger window can remove noise more effectively, but also blur the details/edges

Mean/Average Filter

Filter Kernel

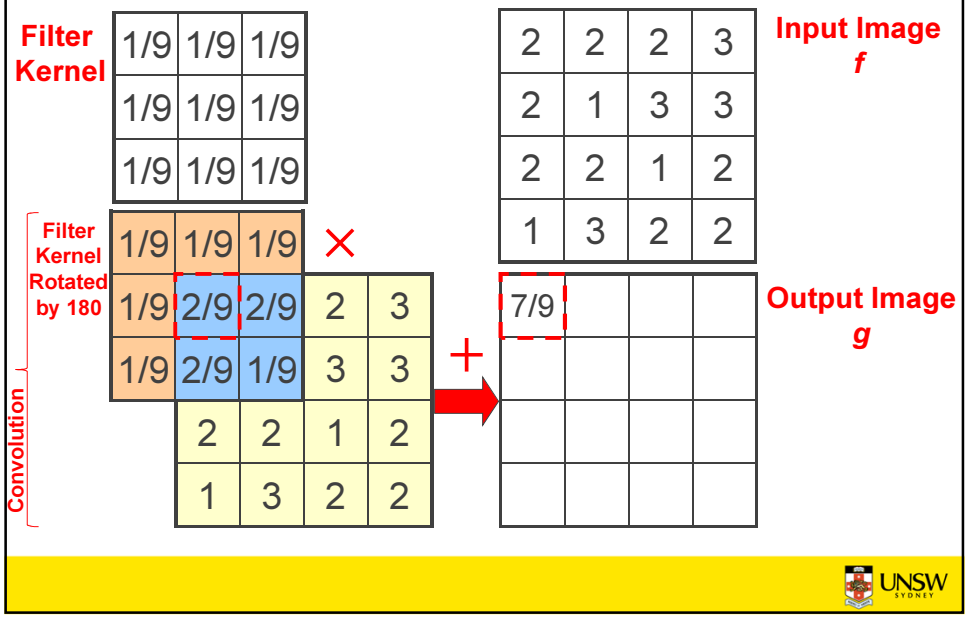
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

NB: Sum of elements is 1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

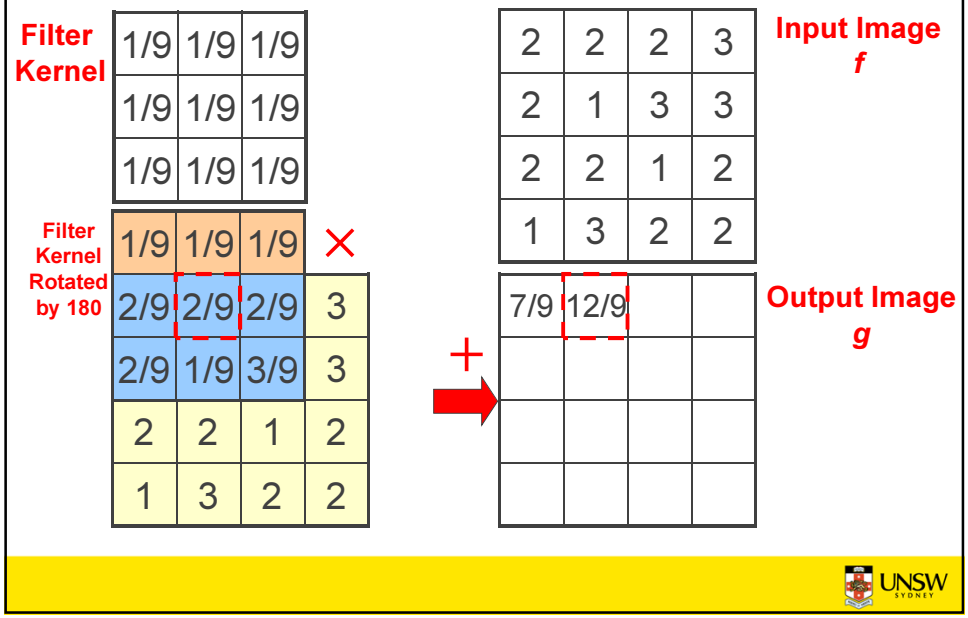
Input Image *f*

Mean/Average Filter



UNSW SYDNEY

Mean/Average Filter



UNSW SYDNEY

Mean/Average Filter

Filter Kernel

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Input Image ***f***

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Filter Kernel Rotated by 180

2	2	3	1
2	1	2	2
3	3	1/9	2/9	1/9
3	2	2/9	2/9	1/9
		1/9	1/9	1/9

×

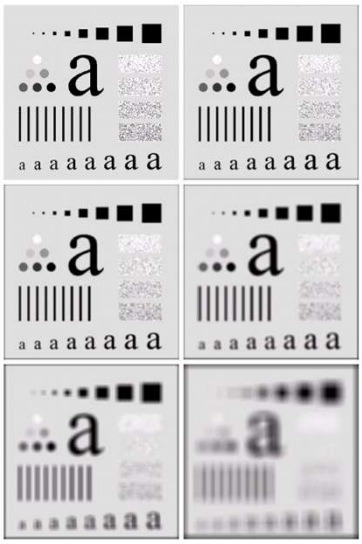
+

Output Image ***g***

7/9	12/9	14/9	11/9
11/9	17/9	19/9	14/9
11/9	17/9	19/9	13/9
8/9	11/9	12/9	7/9

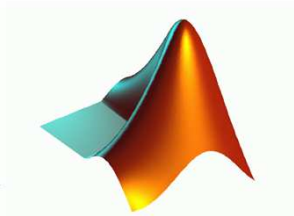
UNSW SYDNEY

Mean/Average Filter



UNSW SYDNEY

MATLAB – Mean/Average Filter



```
% read grayscale image
I = imread('coins.png')

% define filter kernel
k = ones(3, 3) / 9; % 3x3 mean kernel

% perform convolution, keeping size of I
J = conv2(I, k,'same'); % http://au.mathworks.com/help/matlab
```

OR

```
% read grayscale image
I = imread('coins.png');

% create pre-defined 2-D filter
h = fspecial('average',[3 3]); % http://au.mathworks.com/help/images/ref/fspecial.html

% apply filter
J = imfilter(I, h);

% display images
figure; imshow(I);
figure; imshow(uint8(J));
```

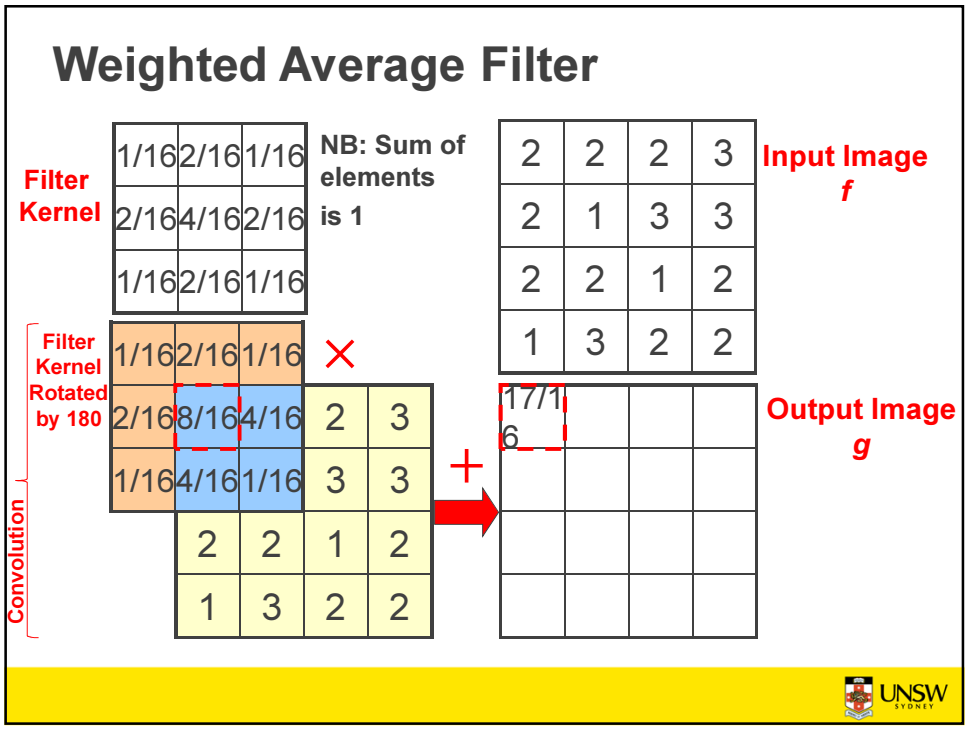
UNSW SYDNEY

Weighted Average Filter

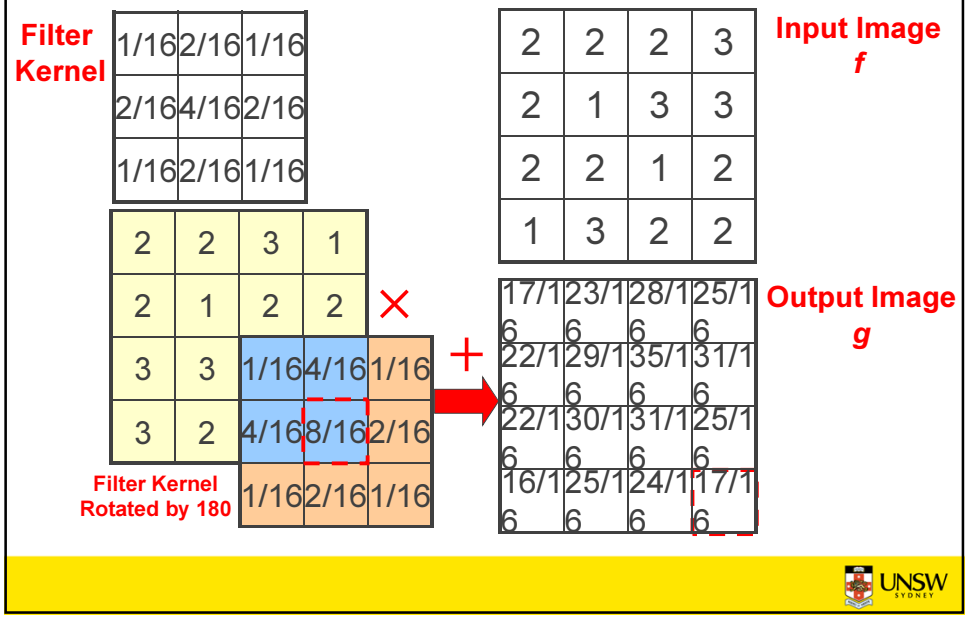
- Instead of averaging all the pixel values in the window, give the closer-by pixels higher weighting, and far-away pixels lower weighting
- This type of operation for arbitrary weighting matrices is generally called “2-D convolution or filtering”
- When all the weights are positive, it corresponds to weighted average
- Weighted average filter retains low frequency and suppresses high frequency = low-pass filter

UNSW SYDNEY

Weighted Average Filter



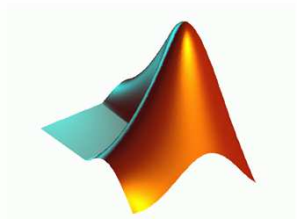
Weighted Average Filter



Weighted Average Filter

Noisy | Average | Weighted Average

MATLAB – Weighted Average Filter



```
% read grayscale image
I = imread('coins.png')

% display image
figure; imshow(I);

% define filter kernel
k = [1, 2, 1; 2, 4, 2; 1, 2, 1] / 16;

% perform convolution, keeping size of I
J = conv2(I, k,'same'); % http://au.mathworks.com/help/matlab/ref/conv2.html

% display image
figure; imshow(uint8(J));
```

Median Filter

- Problem with Averaging Filter –
  - Blur edges and details in an image –
  - Not effective for impulse noise (Salt-and-pepper)
- Median filter: Taking the median value instead of the average or weighted average of pixels in the window

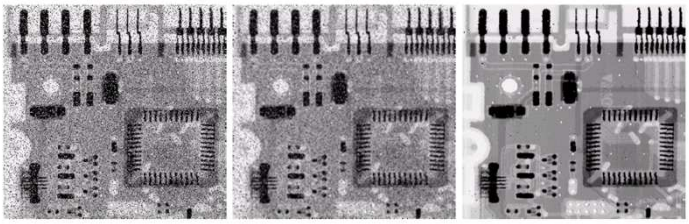
Median Filter

- Median: sort all the pixels in an increasing order, take the middle one
  - The window shape does not need to be a square – Special shapes can preserve line structures
- Order-statistics filter
  - Instead of taking the mean, rank all pixel values in the window, take the n^{th} order value.
  - E.g. max or min

Median Filter

- Use of a median filter to improve an image severely corrupted by defective pixels (pixels on a liquid crystal display that are not performing as expected)

Median Filter

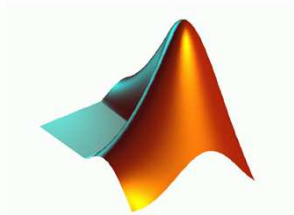


Corrupted by salt and pepper noise

Averaging filter

Median filter

MATLAB – Median Filter



```
% read grayscale image
I = imread('coins.png');
% add salt & pepper noise to image
I = imnoise(I,'salt & pepper',0.02);

% display image
figure; imshow(I);

% perform convolution, keeping size of I
J = medfilt2(I, [3,3]); % http://au.mathworks.com/help/matlab/ref/conv2.html

% display image
figure; imshow(J);
```

UNSW SYDNEY

Image Sharpening

- Sharpening : to enhance line structures or other details in an image
- Enhanced image = original image + scaled version of the line structures and edges in the image
- Line structures and edges can be obtained by applying a difference operator (=high pass filter) on the image
- Combined operation is still a weighted averaging operation, but some weights can be negative, and the sum=1.
- In frequency domain, the filter has the “highemphasis” character

UNSW SYDNEY

Image Sharpening – High Pass Filter

- Spatial operation: taking difference between current and averaging (weighted averaging) of nearby pixels
- Can be interpreted as weighted averaging
  - i.e. linear convolution
- Can be used for edge detection

Image Sharpening

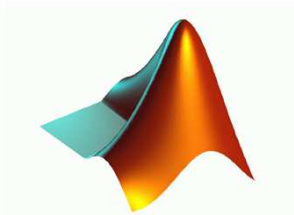
- Example high-pass filters
  - NB: Coefficients sum to zero

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

NB: Sum of elements is 0

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

MATLAB – Image Sharpening



```
% read grayscale image
I = imread('coins.png')

% display image
figure; imshow(I);

% define filter kernel
k = [0 -1 0; -1 4 -1; 0 -1 0];

% perform convolution, keeping size of I
J = conv2(I, k,'same'); % http://au.mathworks.com/help/matlab/ref/conv2.html

% display image
figure; imshow(uint8(J));
```

UNSW SYDNEY

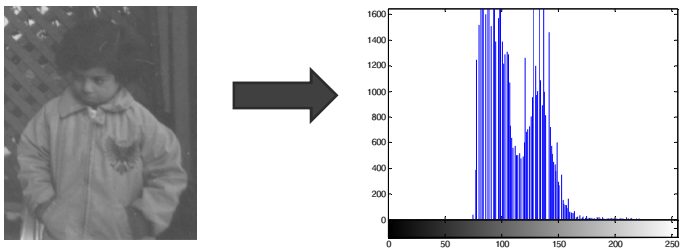
Image Histogram

- An **image histogram** is a type of histogram that acts as a graphical representation of the pixel intensity distribution in a digital image
- It plots the number of pixels for each pixel intensity.
- By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.

UNSW SYDNEY

Image Histogram

- The image is scanned in a single pass and a running count of the number of pixels found at each intensity value is kept. This is then used to construct a suitable histogram.



MATLAB – Image Histogram

```
% read grayscale image
I = imread('pout.tif');

% display image
figure; imshow(I);

% display image histogram
figure; imhist(I); % http://au.mathworks.com/help/images/ref/imhist.html
```

Contrast Stretching

- Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the contrast in an image by `stretching' the range of intensity values it contains to span a desired range of values,
  - *e.g.* the the full range of pixel values that the image type concerned allows.

Contrast Stretching

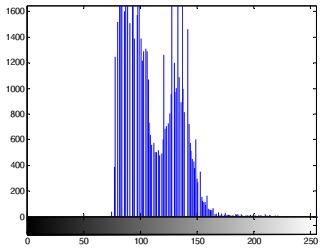
- Before the stretching can be performed it is necessary to specify the upper and lower pixel value limits over which the image is to be normalized.
- Often these limits will just be the minimum and maximum pixel values that the image type concerned allows.
  - For example for 8-bit graylevel images the lower and upper limits might be 0 and 255.
  - Call the lower and the upper limits *a* and *b* respectively.
- The simplest sort of normalization then scans the image to find the lowest and highest pixel values currently present in the image. Call these *c* and *d*. Then each pixel *P* is scaled using the following function:

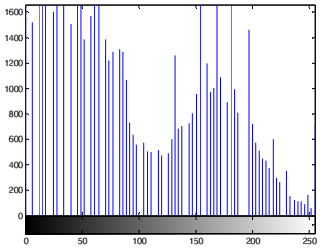
$$P_{new} = (P_{old} - c)\left(\frac{b-a}{d-c}\right) + a$$

Contrast Stretching

$$P_{new} = (P_{old} - c)\left(\frac{b-a}{d-c}\right) + a$$

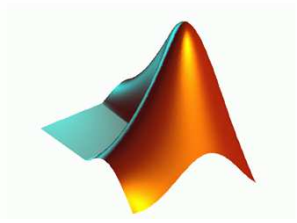
$$a = 0$$
$$b = 255$$
$$c = 78$$
$$d = 161$$





UNSW SYDNEY

MATLAB – Contrast Stretching



```
% read grayscale image
I = imread('pout.tif');

% display image
figure; imshow(I);

% display image histogram
figure; imhist(I);

% determine min and max pixel intensity
[minI,maxI] = stretchlim(I); % http://au.mathworks.com/help/images/ref/stretchlim.html

% perform contrast stretching
J = imadjust(I,[minI,maxI],[]); % http://au.mathworks.com/help/images/ref/imadjust.html

% display image
figure; imshow(J);

% display image histogram
figure; imhist(J);
```

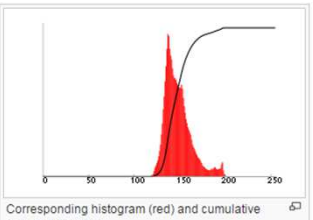
UNSW SYDNEY

Histogram Equalisation

- **Histogram equalization** is a method in image processing of contrast adjustment using the image's histogram.
- The idea of histogram equalization is that the pixels should be distributed evenly over the whole intensity range, *i.e.* the aim is to transform the image so that the output image has a *flat* histogram.

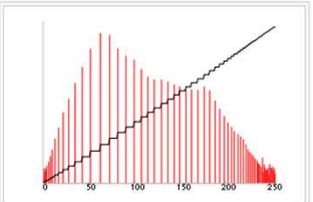
Histogram Equalisation

An unequalized image



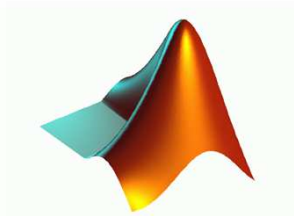
Corresponding histogram (red) and cumulative histogram (black)

The same image after histogram equalization



Corresponding histogram (red) and cumulative histogram (black)

MATLAB – Histogram Equalisation



```
% read grayscale image
I = imread('tire.tif');
% display image
figure; imshow(I);
% display image histogram
figure; imhist(I);

% perform histogram equalisation
J = histeq(I); % http://au.mathworks.com/help/images/ref/histeq.html

% display image
figure; imshow(J);
% display image histogram
figure; imhist(J);
```

UNSW SYDNEY

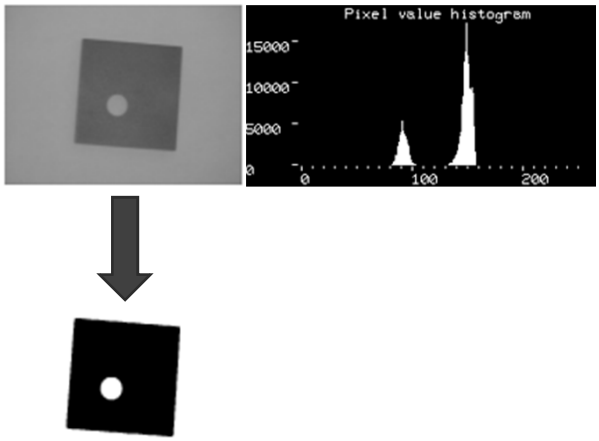
Thresholding – Conversion to Binary

- Histograms have many uses
- One of the more common is to decide what value of threshold to use when converting a grayscale image to a binary one by thresholding.
- If the image is suitable for thresholding then the histogram will be *bi-modal* --- *i.e.* the pixel intensities will be clustered around two well-separated values.
- A suitable threshold for separating these two groups will be found somewhere in between the two peaks in the histogram.
- If the distribution is not like this then it is unlikely that a good segmentation can be produced by thresholding.

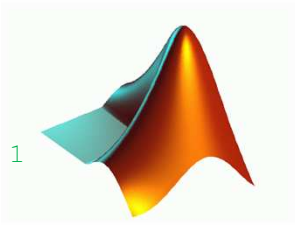
UNSW SYDNEY

Thresholding – Conversion to Binary

- Image with histogram with good bi-modal distribution.
- A threshold of around 120 segments the picture nicely.



MATLAB – Thresholding for Binary



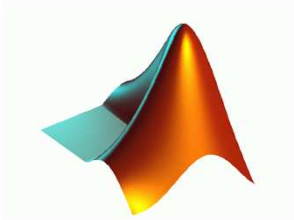
```
% read grayscale image
I = imread('coins.png');

% determine a threshold between 0 and 1
level = 93/255;

% convert grayscale image to binary image
BW = im2bw(I,level); % http://au.mathworks.com/help/images/ref/im2bw.html

% display image
figure; imshow(BW);
```

Otsu Thresholding

- Otsu's method computes a global threshold (level) that can be used to convert an intensity image to a binary image
- Otsu's method, chooses the threshold to minimize the intraclass variance of the black and white pixels.
- Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.

UNSW SYDNEY

MATLAB – Otsu Thresholding



```
% read grayscale image
I = imread('coins.png');

% determine Otsu threshold
level = graythresh(I); % http://au.mathworks.com/help/images/ref/graythresh.html

% convert grayscale image to binary image
BW = im2bw(I,level); % http://au.mathworks.com/help/images/ref/im2bw.html

% display image
figure; imshow(BW);
```

UNSW SYDNEY

Morphological Operations

- Binary images may contain numerous imperfections. In particular, the binary regions produced by simple thresholding are distorted by noise and texture.
- Morphological image processing pursues the goals of removing these imperfections by accounting for the form and structure of the image
- These techniques can be extended to grayscale images

Morphological Operations

- **Morphological image processing** is a collection of non-linear operations related to the shape or morphology of features in an image
- A morphological operation on a binary image creates a new binary image in which the pixel has a non-zero value only if the test is successful at that location in the input image

Morphological Operations

- Morphological techniques probe an image with a small shape or template called a structuring element
- The structuring element is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one:
  - The matrix dimensions specify the size of the structuring element
  - The pattern of ones and zeros specifies the shape of the structuring element
  - An origin of the structuring element is usually one of its pixels, although generally the origin can be outside the structuring element

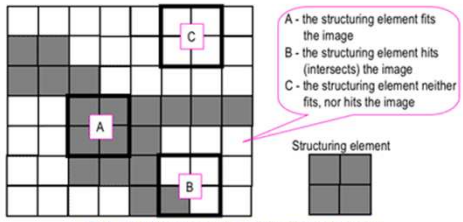
UNSW SYDNEY

Morphological Operations

- The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels
- Some operations test whether the element "fits" within the neighbourhood, while others test whether it "hits" or intersects the neighbourhood

UNSW SYDNEY

Morphological Operations



Probing of an image with a structuring element
(white and grey pixels have zero and non-zero values, respectively).

$s_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

$s_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

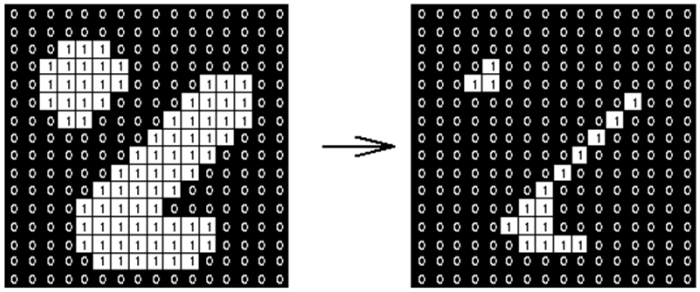
		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Fitting and hitting of a binary image with structuring elements s_1 and s_2.

UNSW SYDNEY

Morphological Erosion

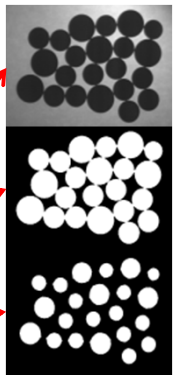
- Tests whether the structuring element “fits”
- Example with 3x3 square structuring element



UNSW SYDNEY

Morphological Erosion

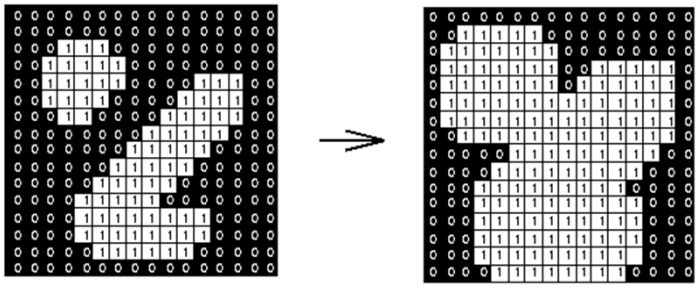
- A number of specialist uses for erosion
- One common use is to separate touching objects in a binary image for counting using a labelling algorithm
  - E.g. Coins silhouetted against a light background
  - Image thresholded at a pixel value of 90
  - Eroding the thresholded image twice with a disk shaped structuring element 11 pixels in diameter
    - coins nicely separated and shape largely preserved
  - Can now use a labelling algorithm to count the coins and the relative sizes of the coins can be used to distinguish the various types by the area of each distinct region



UNSW SYDNEY

Morphological Dilation

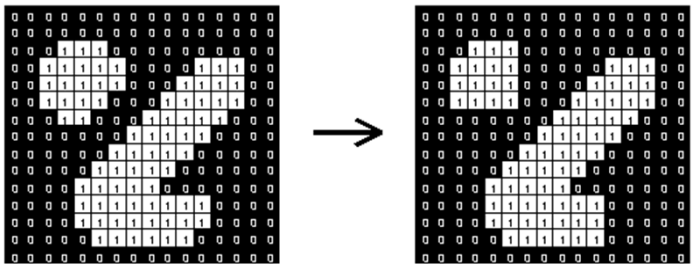
- Tests whether the structuring element "hits"
- Example with 3x3 square structuring element



UNSW SYDNEY

Morphological Opening

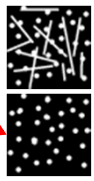
- Morphological erosion followed by dilation with the same structuring element
- Example with 3x3 square structuring element



UNSW SYDNEY

Morphological Opening

- Separate out the circles from the lines
  - Opening with a disc shaped structuring element 11 pixels in diameter
  - The lines have been almost completely removed while the circles remain almost completely unaffected

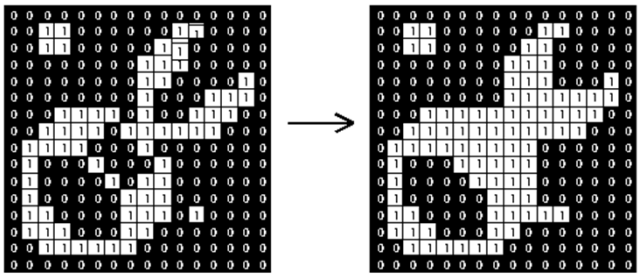


- Separate horizontal and vertical lines
  - Opening with a 3x9 vertically oriented structuring element gives the vertical lines
  - Opening with a 9x3 horizontally oriented structuring element gives the horizontal lines
    - There are a few glitches where the diagonal lines cross vertical lines, but this could be eliminated using a slightly longer structuring element

UNSW SYDNEY

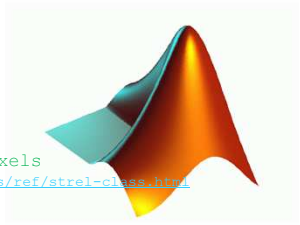
Morphological Closing

- Morphological dilation followed by erosion with the same structuring element
- Example with 3x3 square structuring element



UNSW SYDNEY

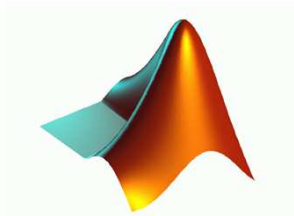
MATLAB – Morphological Operations



```
% read binary image
originalBW = imread('circles.png');
figure; imshow(originalBW); % display image

% create a disk structuring element with radius 15 pixels
se = strel('disk', 15); % http://au.mathworks.com/help/images/ref/strel-class.html
figure; imshow(se.Neighborhood);

% dilate using structuring element se
dilateBW = imdilate(originalBW,se); % http://au.mathworks.com/help/images/ref/imdilate.html
figure; imshow(dilateBW); % display image

% erode using structuring element se
imerodeBW = imerode(originalBW,se); % http://au.mathworks.com/help/images/ref/imerode.html
figure; imshow(imerodeBW); % display image

% morphologically open using structuring element se
imopenBW = imopen(originalBW,se); % http://au.mathworks.com/help/images/ref/imopen.html
figure; imshow(imopenBW); % display image

% morphologically close using structuring element se
imcloseBW = imclose(originalBW,se); % http://au.mathworks.com/help/images/ref/imclose.html
figure; imshow(imcloseBW); % display image
```

UNSW SYDNEY

Region Properties



```
stats = regionprops(BW,properties);
```

Returns measurements for the set of properties specified by properties for each connected component (object) in the binary image, `BW`.

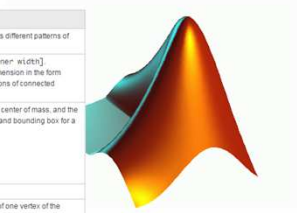
`stats` is struct array containing a struct for each object in the image.

You can use `regionprops` on contiguous regions and discontiguous regions

http://au.mathworks.com/help/images/ref/regionprops.html

Region Properties

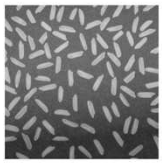
Property Name String	Description
'Area'	Returns a scalar that specifies the actual number of pixels in the region. (This value might differ slightly from the value returned by bwarea, which weights different patterns of pixels differently.)
'BoundingBox'	Returns the smallest rectangle containing the region, specified as a 1-by-Q*2 vector, where Q is the number of image dimensions, for example, [ul_corner width]. ul_corner specifies the upper-left corner of the bounding box in the form [x y z ...]. width specifies the width of the bounding box along each dimension in the form [x_width y_width ...]. regionprops uses ndims to get the dimensions of label matrix or binary image, ndims(L), and numel to get the dimensions of connected components, numel(CC.ImageSize).
'Centroid'	Returns a 1-by-Q vector that specifies the center of mass of the region. The first element of Centroid is the horizontal coordinate (or x-coordinate) of the center of mass, and the second element is the vertical coordinate (or y-coordinate). All other elements of Centroid are in order of dimension. This figure illustrates the centroid and bounding box for a discontiguous region. The region consists of the white pixels; the green box is the bounding box, and the red dot is the centroid.
'ConvexArea'	Returns a scalar that specifies the number of pixels in 'ConvexImage'.
'ConvexHull'	Returns a p-by-2 matrix that specifies the smallest convex polygon that can contain the region. Each row of the matrix contains the x- and y-coordinates of one vertex of the polygon.
'ConvexImage'	Returns a binary image (logical) that specifies the convex hull, with all pixels within the hull filled in (set to on). The image is the size of the bounding box of the region. (For pixels that the boundary of the hull passes through, regionprops uses the same logic as roipoly to determine whether the pixel is inside or outside the hull.)
'Eccentricity'	Returns a scalar that specifies the eccentricity of the ellipse that has the same second-moments as the region. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases. An ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.)
'EquivDiameter'	Returns a scalar that specifies the diameter of a circle with the same area as the region. Computed as sqrt(4*Area/pi).
'EulerNumber'	Returns a scalar that specifies the number of objects in the region minus the number of holes in those objects. This property is supported only for 2-D label matrices. regionprops uses 8-connectivity to compute the Euler number measurement. To learn more about connectivity, see Pixel Connectivity.
'Extent'	Returns a scalar that specifies the ratio of pixels in the region to pixels in the total bounding box. Computed as the Area divided by the area of the bounding box.
'Extrema'	Returns an 8-by-2 matrix that specifies the extrema points in the region. Each row of the matrix contains the x- and y-coordinates of one of the points. The format of the vector is [top-left top-right right-top right-bottom bottom-right bottom-left left-bottom left-top]. This figure illustrates the extrema of two different regions. In the region on the left, each extrema point is distinct. In the region on the right, certain extrema points (e.g., top-left and left-top) are identical.



… and more

Cell (or Rice) Counting

```
% read image
I = imread('rice.png');
% display image
figure; imshow(I);

% approximate the background - morphological
opening with 15 pixel radius disk
background = imopen(I,strel('disk',15));

% subtract the background from original image
I2 = I - background;

% display the background subtracted image
figure; imshow(I2);
```



Cell (or Rice) Counting

```
% increase the image contrast
I3 = imadjust(I2); % default parameters
% display image
figure; imshow(I3);

% convert to b+w using Otsu threshold
level = graythresh(I3);
bw = im2bw(I3,level);
% remove objects with area < 50 pixels
bw = bwareaopen(bw, 50);
% display image
figure; imshow(bw);
```

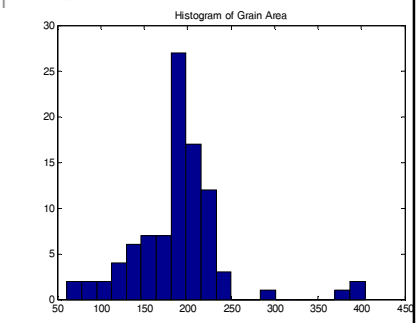
Cell (or Rice) Counting

```
% get area, centroid location and
bounding box of each grain of rice
graindata = regionprops(bw, 'basic');

% get vector of all areas
grain_areas = [graindata.Area];

% plot histogram of areas
nbins = 20;
figure; hist(grain_areas, nbins);
title('Histogram of Grain Area');
```

graindata <93x1 struct>

graindata(1, 1) <1x1 struct>

Field	Value	Min	Max
Area	138	138	138
Centroid	[5.9855,35.0870]	5.9855	35.0870
BoundingBox	[0.5000,22.5000,13,22]	0.5000	22.5000

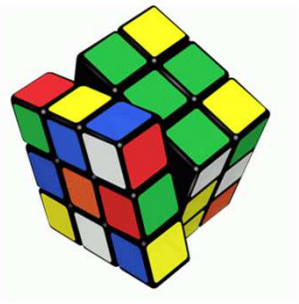


UNSW SYDNEY

Cell (or Rice) Counting

http://au.mathworks.com/help/images/image-enhancement-and-analysis.html

Cell counting is slightly more complicated than the counting rice example, but we will attempt this in the tutorial…



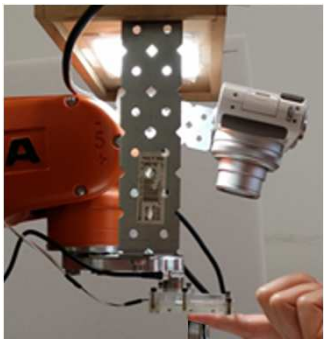
UNSW SYDNEY

Fingerprint Tracking

- Understanding the biomechanics of the human finger while manipulating objects
- Track the fingerprint (stretch, compression, deformation) when in contact with glass
- The light source and camera are set up in such a way that due to relative refractive indices of air, glass and skin, light is reflected from the glass when skin out of contact, and refracts into the glass and is absorbed by the skin when in contact
  - i.e. when fingerprint ridges come into contact with the glass, they appear dark, and the fingerprint valleys (and other skin) which are not in contact with the glass, appear bright

UNSW SYDNEY

Fingerprint Tracking

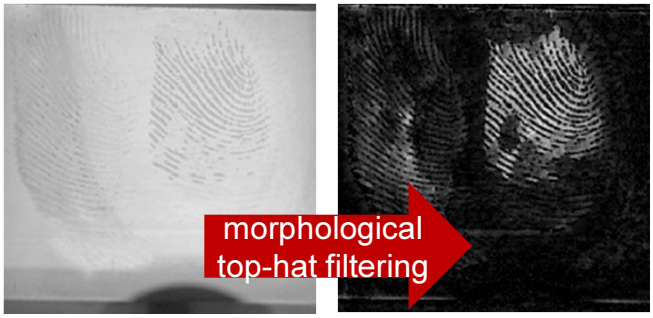


- Where is the contact area?
- Shouldn't track points that aren't in contact

UNSW SYDNEY

Where is the Contact Area?

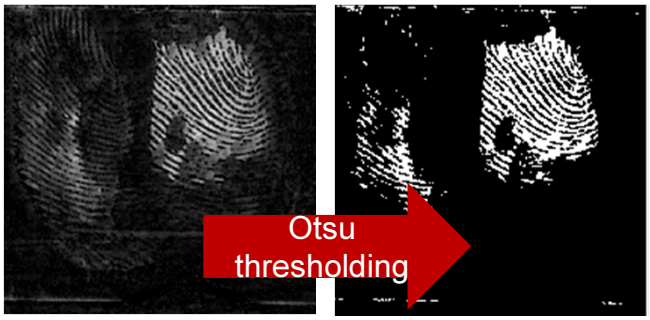
- Top-hat filtering computes the morphological opening of the image and then subtracts the result from the original image
- `IM2 = imtophat(IM,SE)`
- http://au.mathworks.com/help/images/ref/imtophat.html



UNSW SYDNEY

Where is the Contact Area?

- Otsu thresholding



UNSW SYDNEY

Where is the Contact Area?

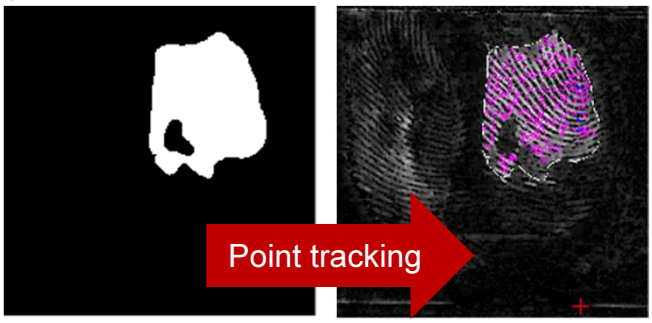
- Morphological closing then opening to fill in holes
- Median filter to soften the border

Where is the Contact Area?

- Remove residual areas
- bwareaopen(flatBorderFrame,pixelsThreshold)

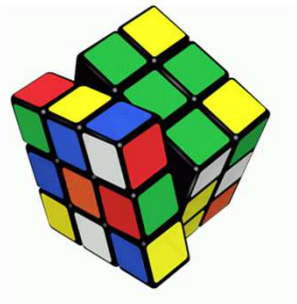
Now We Can Track Points

- For more information:
  http://au.mathworks.com/help/vision/ref/vision.pointtracker-class.html



Fingerprint Tracking - Contact Area

We will attempt this with our own fingerprint images in the tutorial



Malnutrition Assessment Challenges

- Skin detection to identify arms and face
- Crude colour matching:

R > 95 AND G > 40 AND B > 20

AND

v = [R,G,B]
(max(v)-min(v)) > 15

AND

abs(R-G) > 15 AND R > G AND R > B

UNSW SYDNEY

Detecting Skin

We will attempt this in the tutorial to answer the question…

"How much fake-tan does it take before skin is unrecognisable?"

UNSW SYDNEY

Tutorial Preparation

- The tutorial for this class is in week 11
- Tutorial instructions will be online soon
- Do the online MATLAB tutorial:
  - http://moodle.telt.unsw.edu.au/course/view.php?id=24200
- Practice running the code snippets in this lecture on various images to become familiar with the effects of each of the image processing techniques
- Please bring a laptop to the tutorial, ideally with MATLAB installed or access to MATLAB Online in your browser via a Mathworks account
- sign-up with your UNSW email
  http://au.mathworks.com/products/matlab-online

UNSW SYDNEY

Some Helpful Resources

- Some tutorials
  - http://homepages.inf.ed.ac.uk/rbf/HIPR2/index.htm
  - https://sisu.ut.ee/imageprocessing/book/1
  - http://www.tutorialspoint.com/dip/
  - http://www.imageprocessingplace.com/root_files_V3/tutorials.htm
  - http://www.cs.dartmouth.edu/farid/downloads/tutorials/fip.pdf
    - Math heavy
- General
  - https://en.wikipedia.org
  - http://www.google.com.au
- MATLAB has very extensive documentation
  - http://au.mathworks.com/help/
  - Easier to google and then choose the google result on the mathworks website
- MATLAB forums are also very helpful for answers and code
  - https://au.mathworks.com/matlabcentral/
  - Again, easier to google, then choose google result on the forum website

UNSW SYDNEY