

COMP2005

Digital Images and Point Processes

TODAY'S LECTURE ..

1. Digital Image Formation
2. Acquiring and Colour Images
3. Colour Spaces
4. Intensity Transform

Digital Image Formation & Acquisition

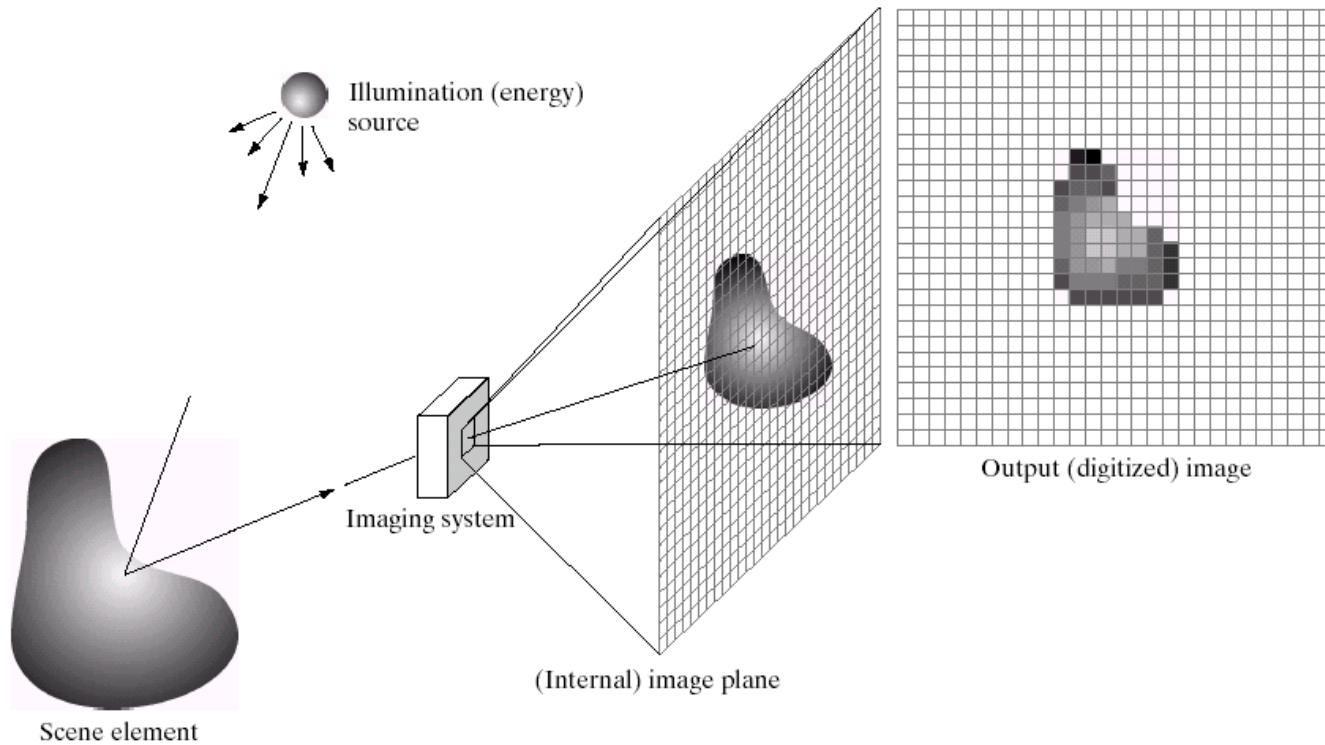
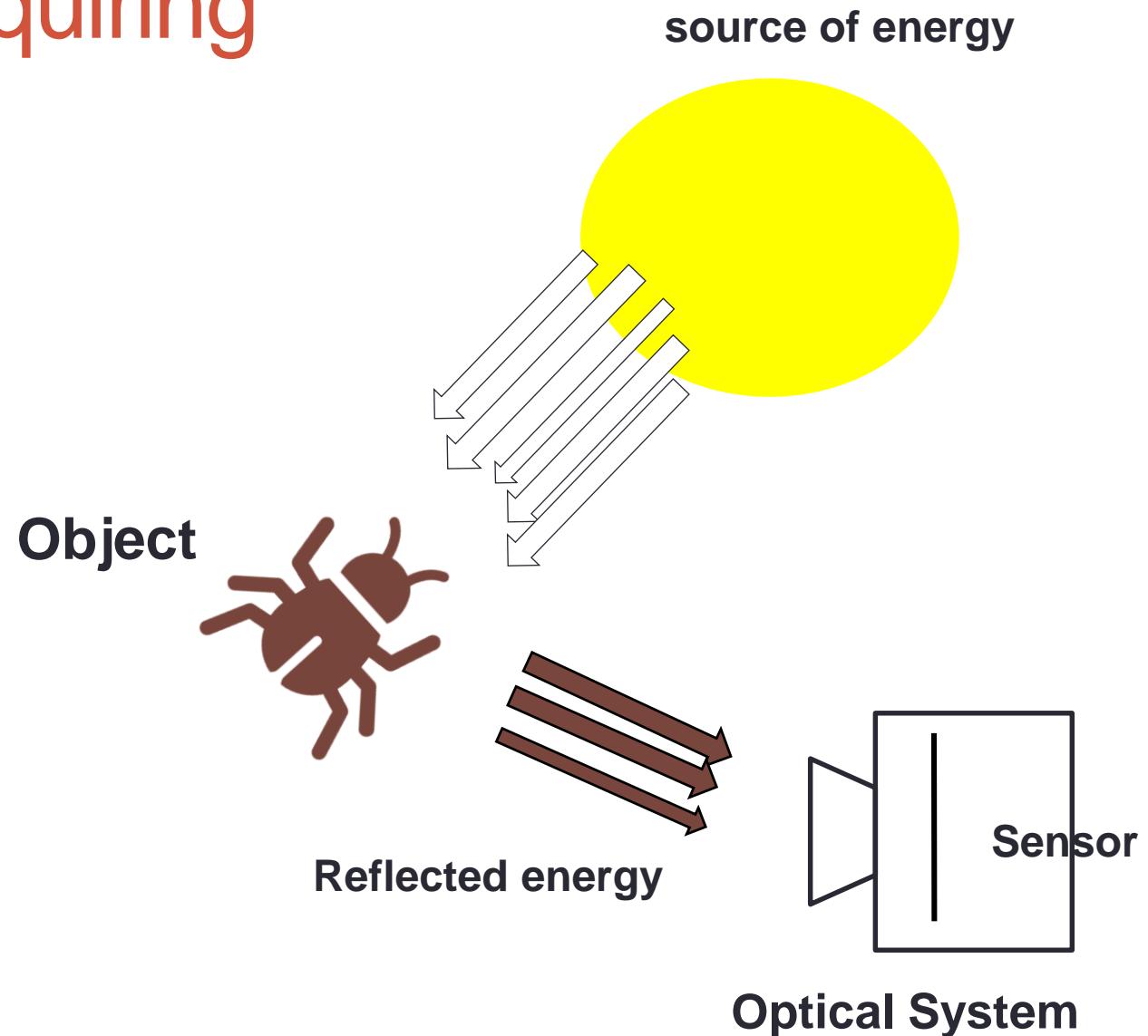
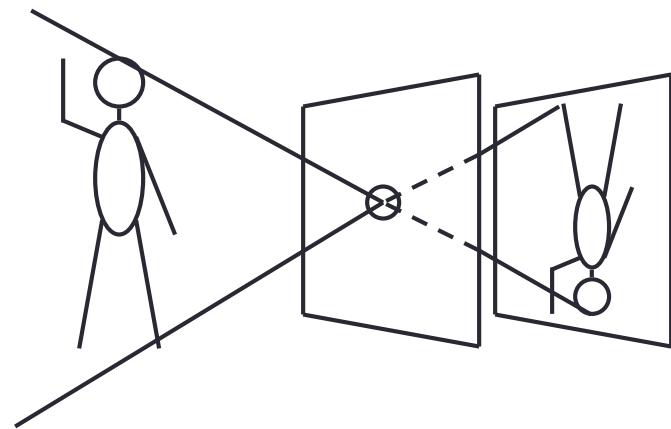


Image Acquiring



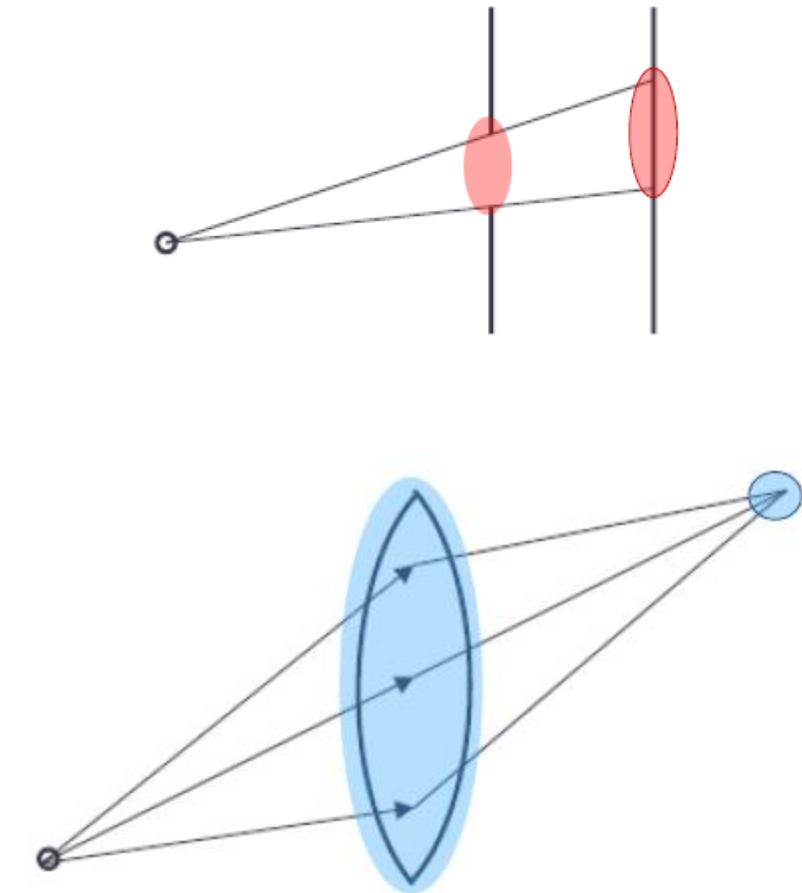
A Simple Camera Model

- A simple camera model is the **pinhole camera** model
 - Light passes through a small hole and falls on an **image plane**
 - The image is inverted and scaled
 - Early cameras used this approach



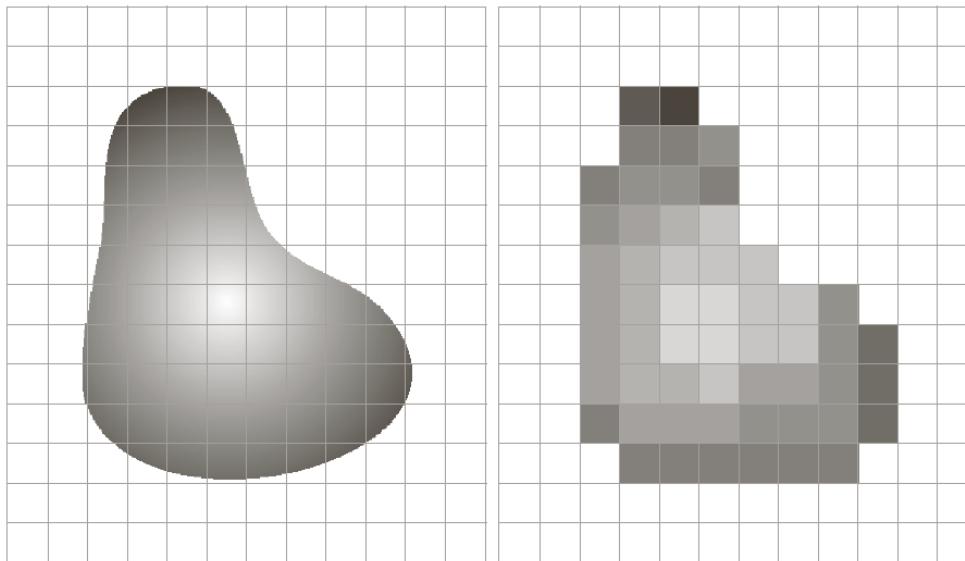
Real vs Pinhole Cameras

- Real cameras do not have ‘pinholes’
- A pinhole lets very little light in, leading to **long exposures**
- Larger holes lead to **blurred** images
- A **lens** can make a large hole act like a small one



Sampling and Quantisation

- Sampling : Digitisation of the **spatial coordinates**
- Quantisation : Digitisation of the **light intensity function**



Sampling determines
spatial resolution

Quantisation determines
***grey level, colour or
radiometric resolution***



How much of the light energy is quantized

Sampling

| How many samples to take? i.e. how many pixels in the image? |

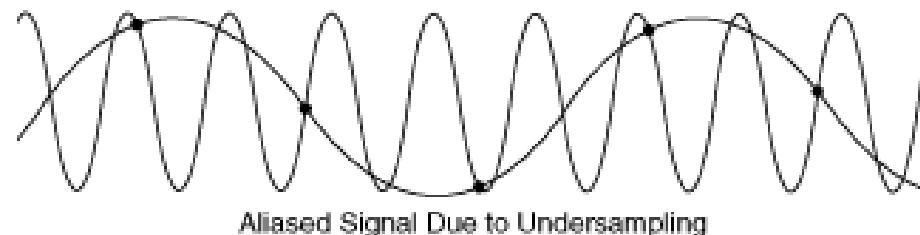
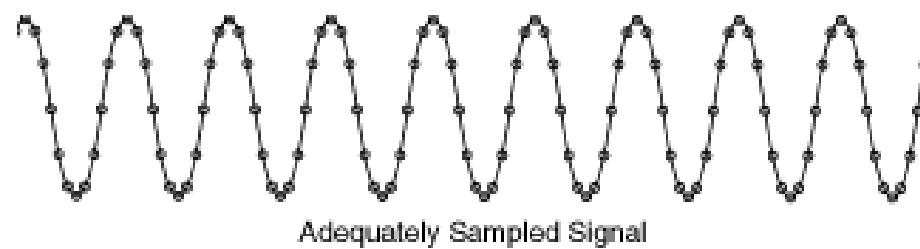
The Nyquist Rate

- Samples must be taken at a rate that *is twice the frequency of the highest frequency component to be reconstructed.*
- Under-sampling: sampling at a rate that is too coarse, i.e., is below the Nyquist rate.
- **Aliasing:** artefacts that result from under-sampling.



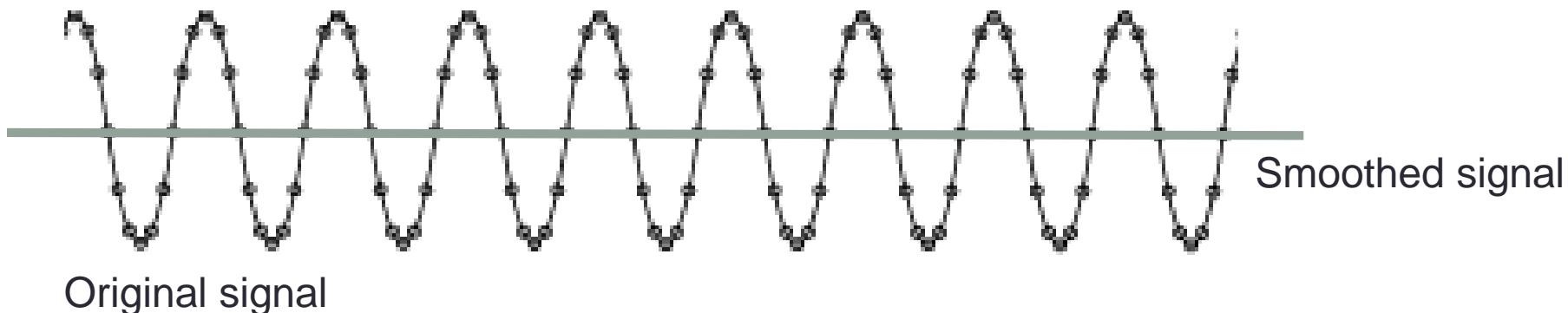
Aliasing

- Aliasing occurs when two signals (images) become indistinguishable when sampled
- In our case the two signals are the true image (the image that would be seen if there were no quantisation) and the one reconstructed by the human vision system from a sampled image



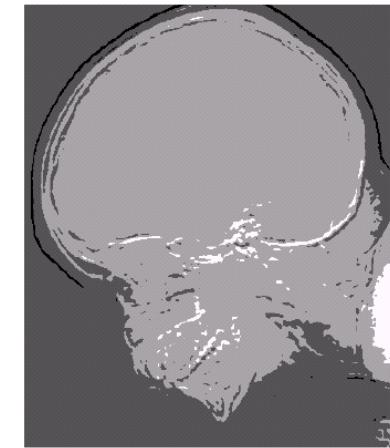
Anti-aliasing

- Aliasing can be introduced when an image is resampled, if the sampling rate of the new image is less than the Nyquist rate of the original
- Smooth out high frequency signals before sampling so its impossible to “see” the alias



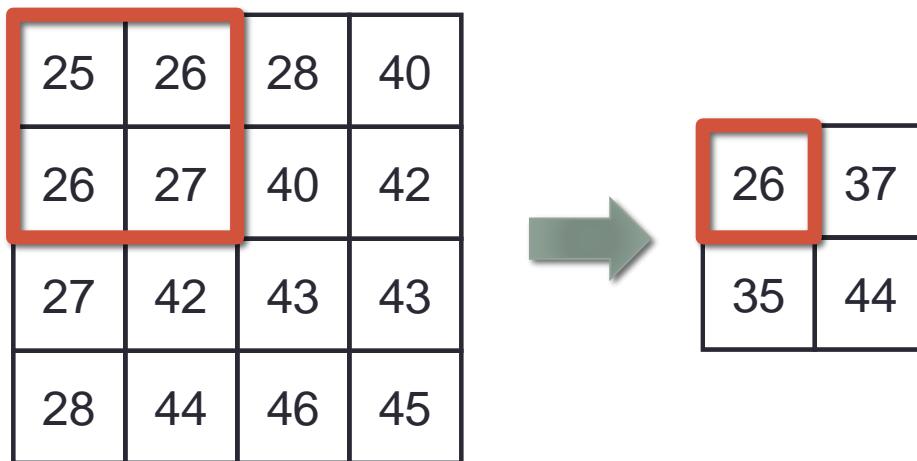
Quantisation

- How many grey levels to store?
- Determines the number of levels of color/intensity to be represented at each pixel
 - 256, 64, 16, 4
- Sampling and quantisation occur naturally during image acquisition, but can also be applied to existing images
 - e.g. during resizing and compression



Re-Sampling and Re-Sizing

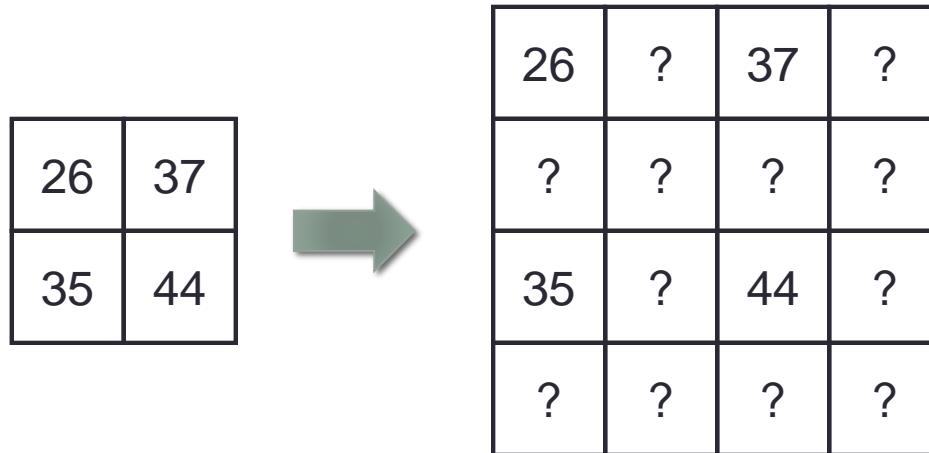
- Most basic form of image processing



- When ***downsampling***, need to compute a summary pixel value from each local area:

pick one, mean, weighted mean....

Re-Sampling and Re-Sizing



- When **upsampling**, need to **interpolate** from the known values **to produce an estimate at the unknown pixels**.
- Average the known values in a local region centred on each unknown pixel, fit some kind of function through known values....

Re-quantisation

- Pixel values are integers in a fixed range
- Grey level resolution can be dropped by dividing each pixel value by a constant, but there is a side effect
- Can't increase grey level resolution of a single pixel
- **Super-resolution methods** exist that combine multiple exposures of the same scene, and so have more than one measurement of each pixel

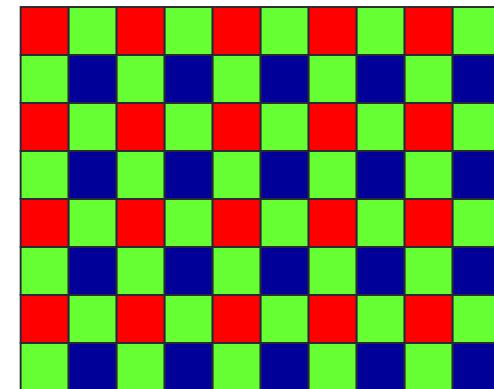
Key Points

- Sampling and quantisation are often under user control during image capture and have a significant effect on the image
- Both can be changed to some extent after capture, but the resulting images are often approximations
- Be aware of the Nyquist rule when choosing an image resolution; it can introduce artefacts that are difficult to remove

ACQUIRING COLOUR IMAGES

Acquiring Colour Images

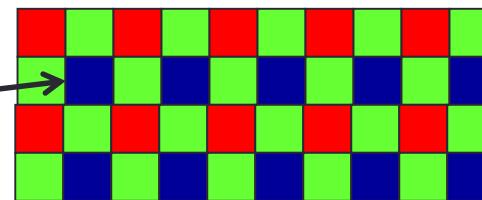
- Each pixel in a grey level image contains one value. Colour requires **three**
- **RGB** model inspired by the **retina** is an obvious choice for use in cameras
- Some (expensive) cameras use complex optics and **3 CCDs** to capture each colour channel independently
- Necessary for some scientific purposes, not for general use
- Single CCD colour cameras use the **Bayer Pattern**:



- Each CCD cell lies under a red, green or blue filter

Bayer Pattern

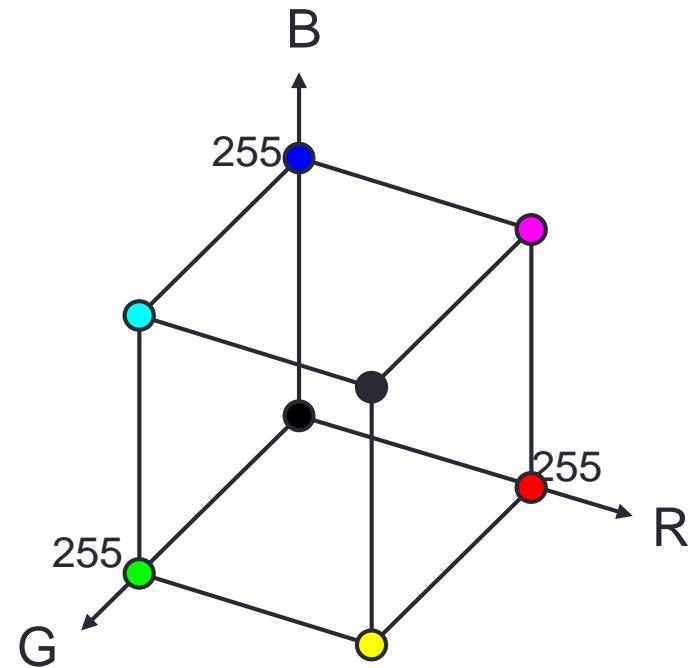
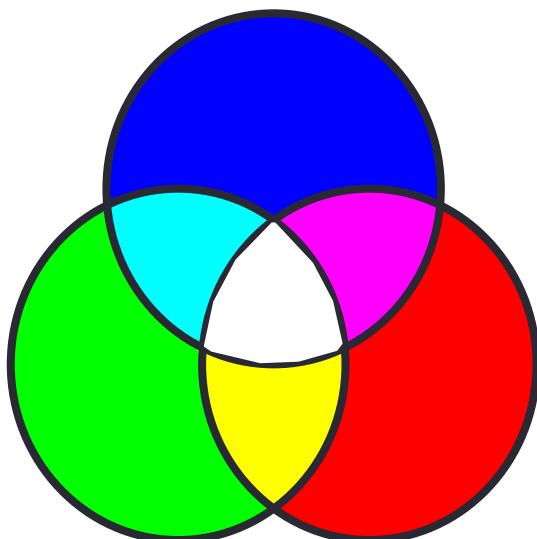
- One colour value is measured, but two are **estimated** at each pixel
- Here B is measured, G is the mean of the 4-neighbours, R is the mean of the diagonals
- Pattern is repetitive so there are a finite number of sets of filters that can surround each sensor



- More green elements as our eye is more sensitive to green light than red or blue

Representing Colour: RGB

- Most common starting point
- Retinal cells are sensitive to three primary colours R, G, B
- Light, not pigments



- RGB is additive, other colours are made by mixing these together

An RGB Example

- Colour images are 3D arrays
- Display software interprets them to produce a colour view



Colour Image

Red

Green

Blue

- RGB is used in image acquisition, but that doesn't mean we have to use it in image processing

Colour vs Greyscale

- Sometimes we want a single value at each pixel
 - Makes processing easier
 - Reduces the amount of information
 - Makes some of the theory simpler
- Many image processing methods were developed for single value images
 - This value is usually the intensity or grey level
 - Early CCDs only produced grey level; rare now
 - We can compute the grey value using a simple average of red, green, and blue
 - But our eyes are more sensitive to green light so...

Colour vs Greyscale

- We can convert an RGB image to greyscale using

$$i = 0.30r + 0.59g + 0.11b$$



Average

Original

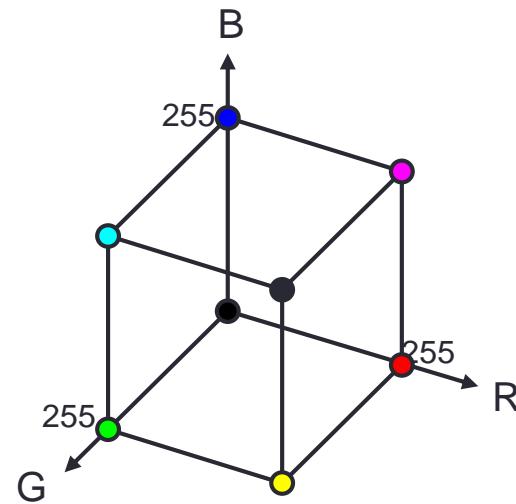
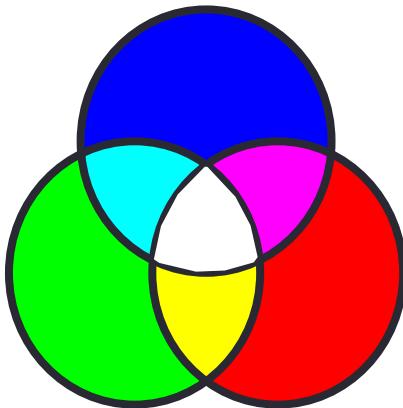
Weighted



COLOUR SPACES

Representing Colour: RGB

- Most common starting point
- Retinal cells are sensitive to three primary colours R, G, B
- Light, not pigments



- RGB is additive, other colours are made by mixing these together

Alternative Colour Spaces

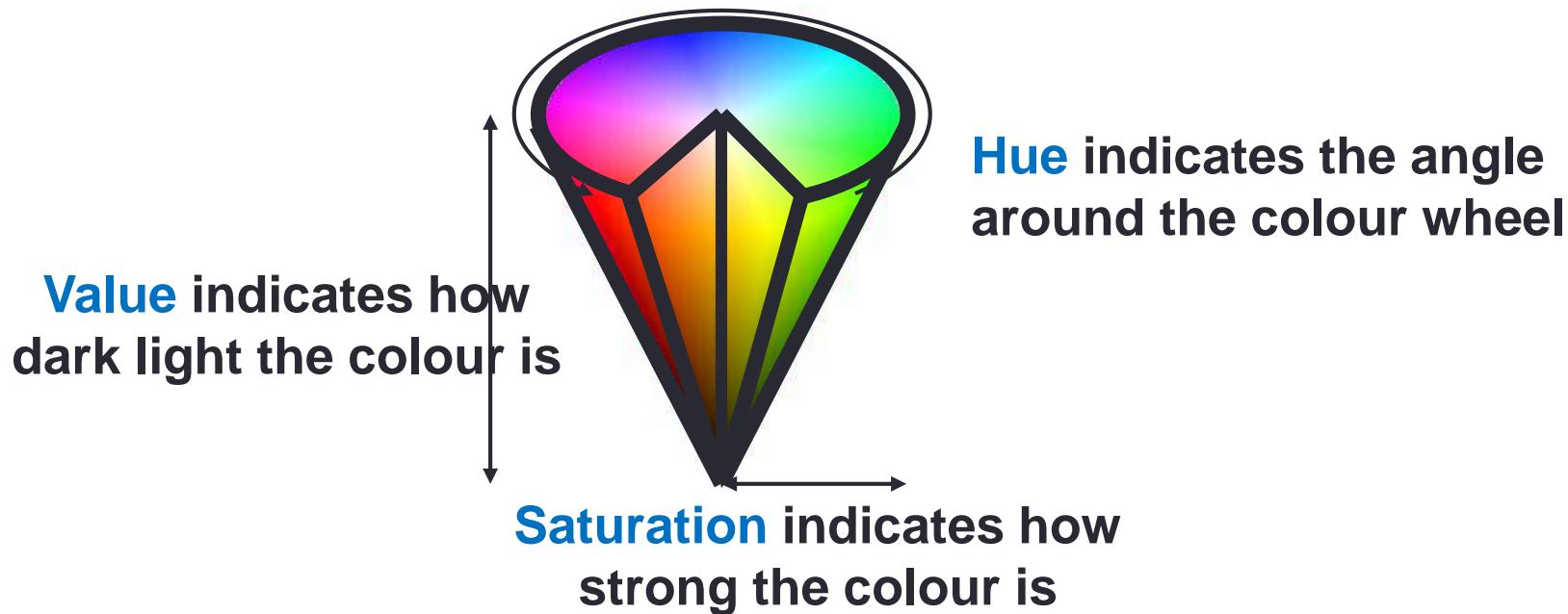
- RGB is OK, but there are other general and application-specific colour spaces.....
- If you want to work with plants you might use just G, or 'greenness':

$$G - (R + B)/2$$



- **HSV** is based on colour rather than light
 - **Hue** - what general colour is it
 - **Saturation** - how strongly coloured is it
 - **Value** - how bright or dark is it

HSV Space

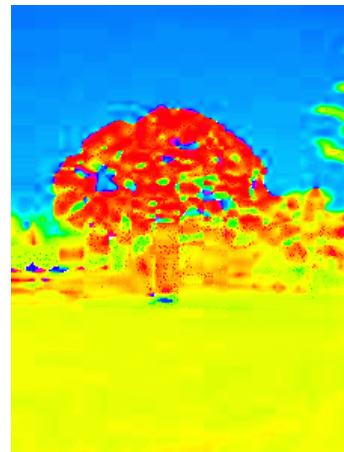


HSV Example

- HSV separates colour from intensity (value) making it less sensitive to illumination changes



Colour Image



Hue (presented as
a colour image)



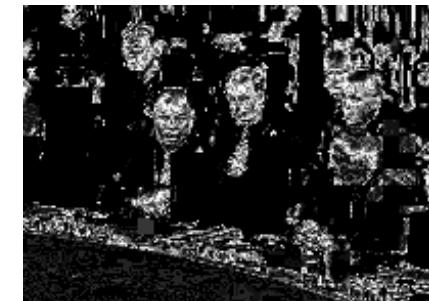
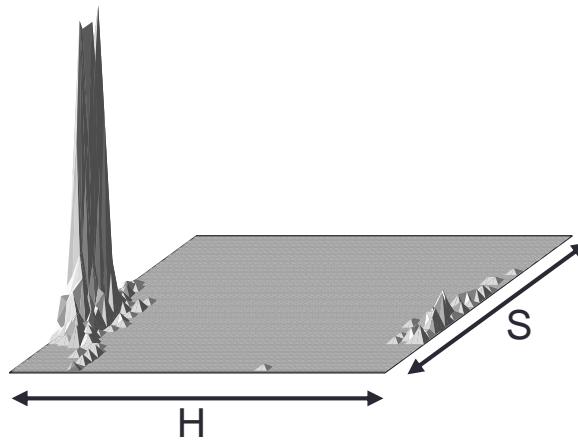
Saturation



Value

Hue, Saturation and Skin

- We don't always need all 3 colour values
- Human skin is tightly clustered in H,S space



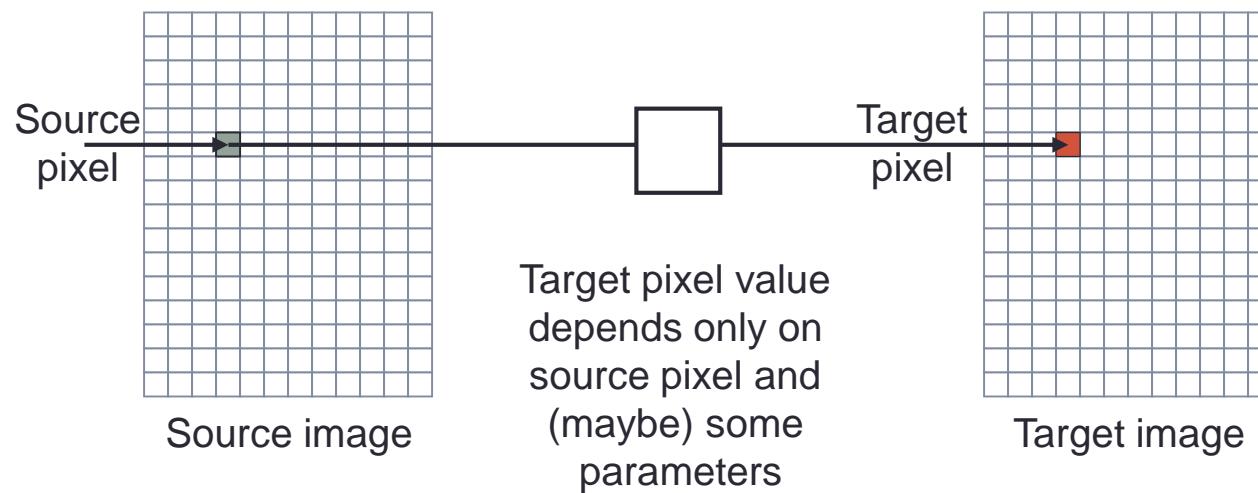
Histogram back-projection

Key Points

- Colour values are often interpolated; don't assume they are 100% reliable unless you have a 3 CCD camera
- Colour space transformation is an example of a point process
- Alternative 3D colour spaces exist
- Not all applications require all 3 colour planes to be considered, and application-specific colour spaces exist.

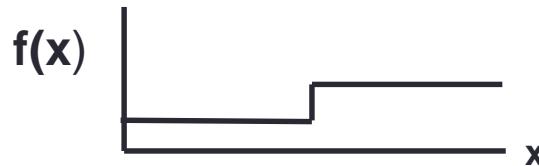
INTENSITY TRANSFORM

Intensity Transforms

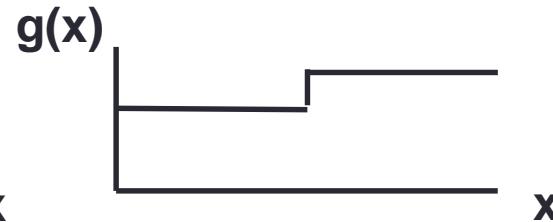
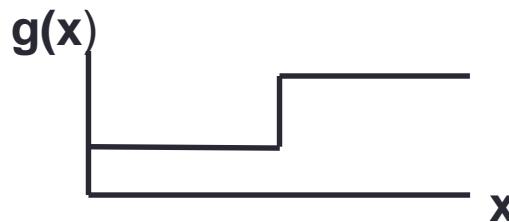


Linear Transforms

- Two commonly used point processes are multiplication by and addition of a constant, i.e. $g(x,y) = a.f(x,y) + b$

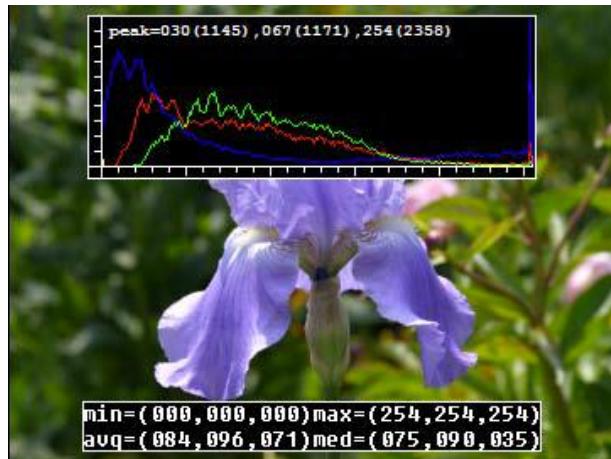


- a is the **gain**, and controls *contrast*, b is the **bias**, and controls *brightness*

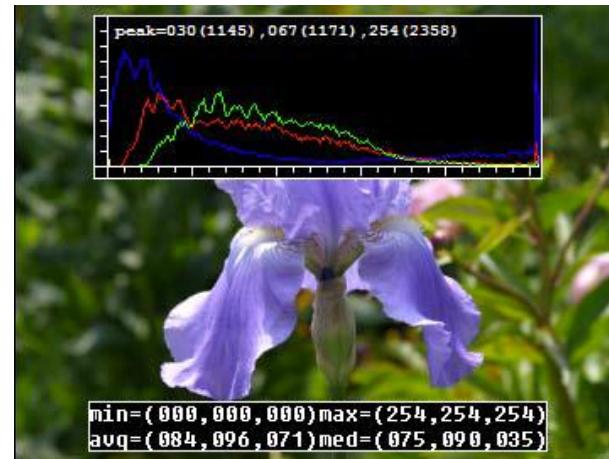


Gain

- $g(x,y) = 1.1 \cdot f(x,y) + 0$



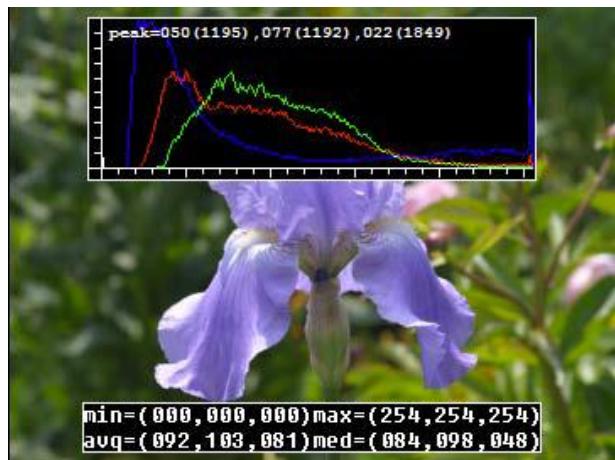
$f(x,y)$



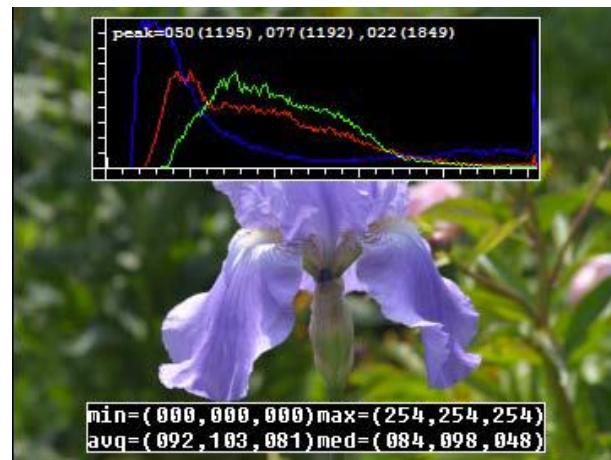
$g(x,y)$

Bias

- $g(x,y) = 1.f(x,y) + 16$



$f(x,y)$



$g(x,y)$

Negation

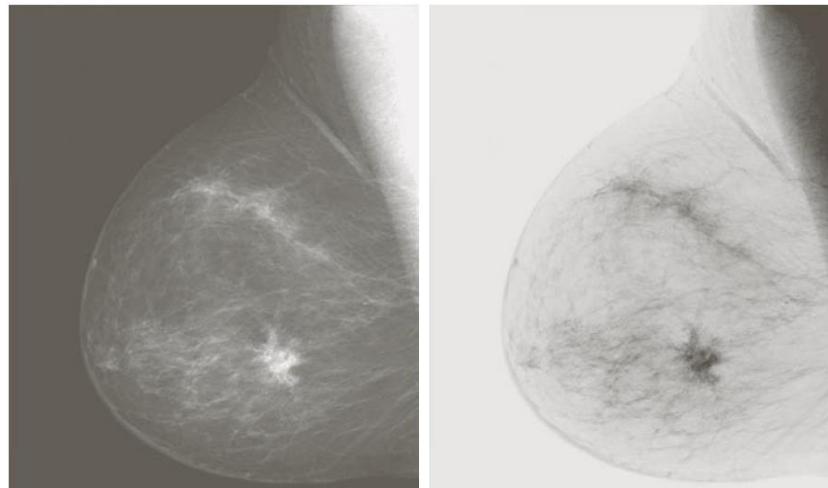
$$g(x,y) = f_{\min} + (f_{\max} - f(x,y))$$

$$= -f(x,y) + (f_{\min} + f_{\max})$$



Negation

- Often used to make fine details more visible,
e.g. in digital mammograms:



Dynamic Range

- Digital images are sampled – they contain a fixed number of data values
- Digital image representations can only store a fixed number of values
- Intensity transforms can produce values that are
 - outside that range and so can't be stored
 - clustered in a small part of that range and so are hard to distinguish
- Some intensity transforms need data in a particular range

Contrast Stretching

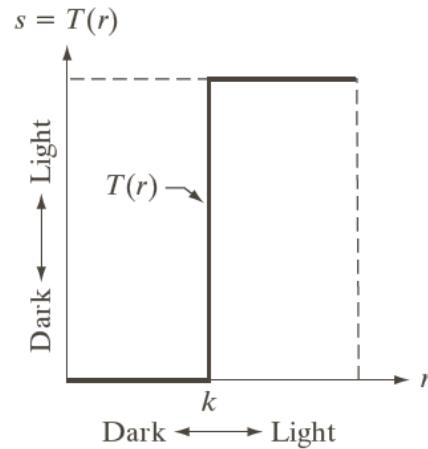
- To convert a source image in which intensities range from \min_s to \max_s to one in which they range from \min_t to \max_t

$$g(x,y) = \min_t + \frac{(f(x,y) - \min_s).(\max_t - \min_t)}{(\max_s - \min_s)}$$



Non-linear Transforms

- Thresholding

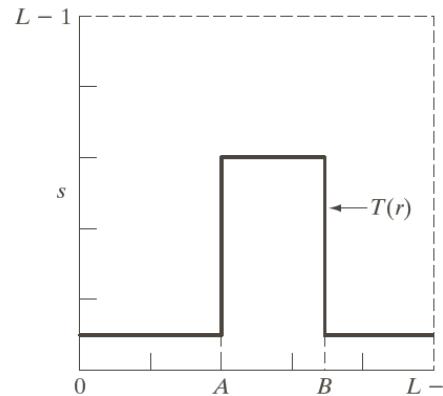


- Can be done to help segmentation

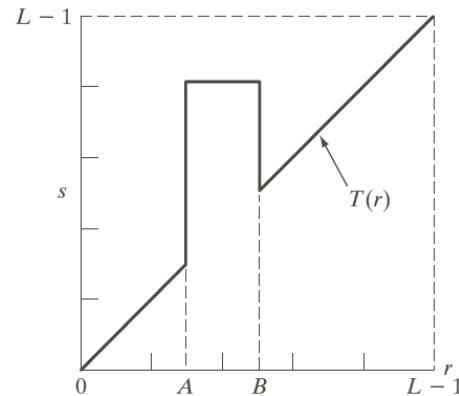
as the simplest form of image

Grey Level Slicing

- Highlights a specific range of intensities



Can reduce other grey levels to
a lower level.....



....or preserve them

Grey Level Slicing



An aortic angiogram



High intensities selected, others reduced to low level



Selected intensities set to black, others (blood vessels & kidneys) preserved

Gamma Correction

- When an image is displayed on a screen, the **hardware** used effectively applies an intensity transform
- You send a voltage proportional to the intensity of a pixel, the screen displays an intensity that is related to that, but not how you may think



$L \approx V^{2.5}$ depending on device

Gamma Correction

- We need to transform the image, so that it generates a voltage which will display what we want
- Create a new image in which

$$g(x,y) = f(x,y)^{1/2.5}$$

then

$$V = g(x,y)$$

$$L = (f(x,y)^{1/2.5})^{2.5} = f(x,y)$$

- **Gamma** Correction because the equation is usually written

$$g(x,y) = f(x,y)^{\gamma}$$

Key Points

- Point processes operate on each pixel independently
- Linear processes change the appearance of the whole image
- Non-linear processes can differentiate objects/image regions

Next Week :

Histogram (Whole Image Understanding)

COMP2005

Histogram Processing

Histograms

- The histogram of a digital image with grey levels in the range $[0, L-1]$ is a discrete function

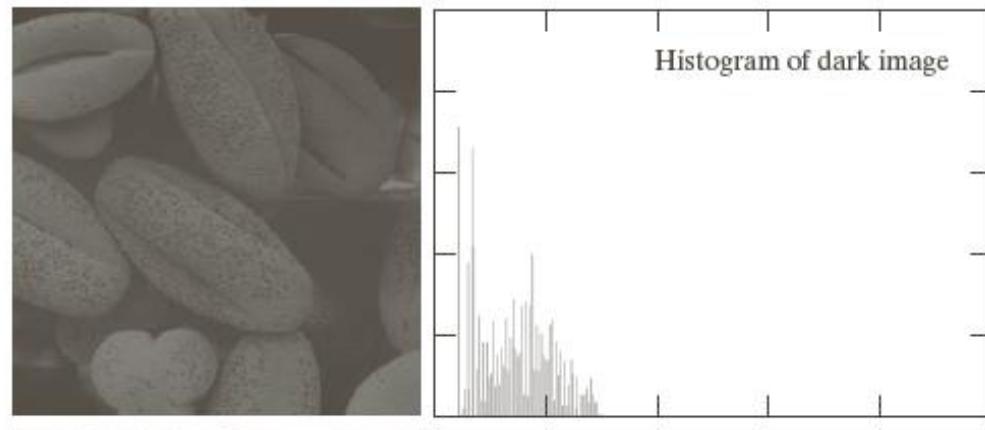
$$p(r_k) = n_k$$

where:

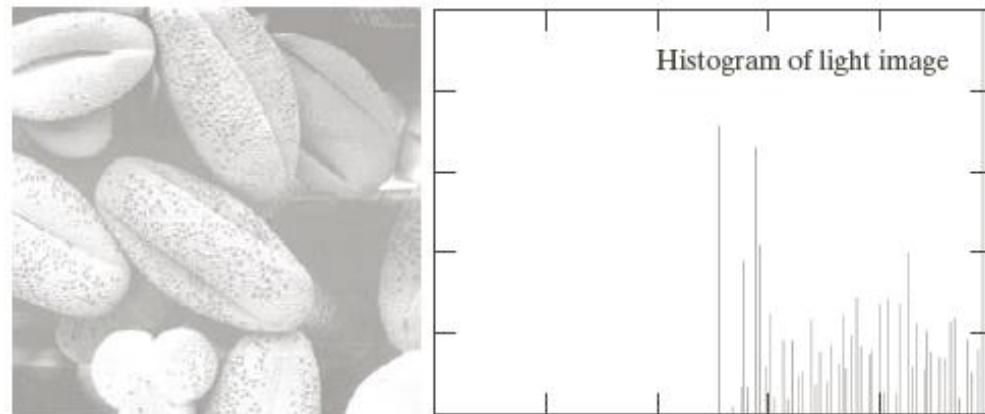
- r_k is the k th grey level,
- n_k is the number of pixels in the image with that grey level,
- $k = 0, 1, \dots, L-1$
- Histograms provide useful global information about the image, ease computation of some image properties, and can be manipulated to improve the image

Histograms

- Dark

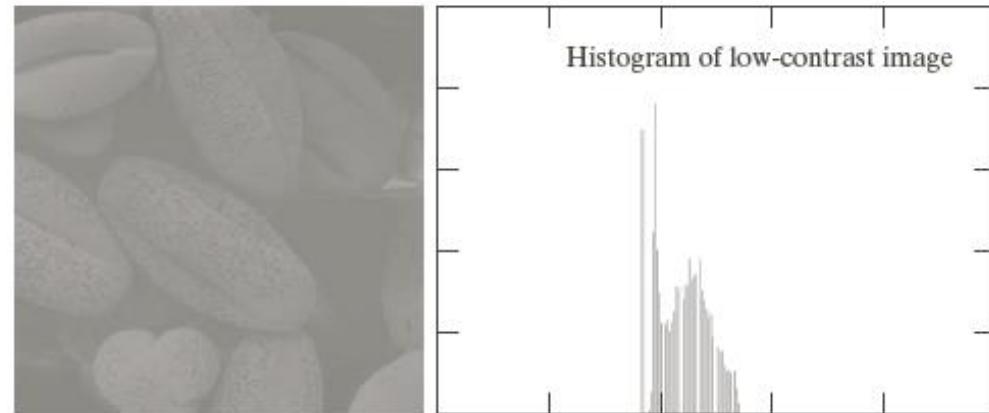


- Light

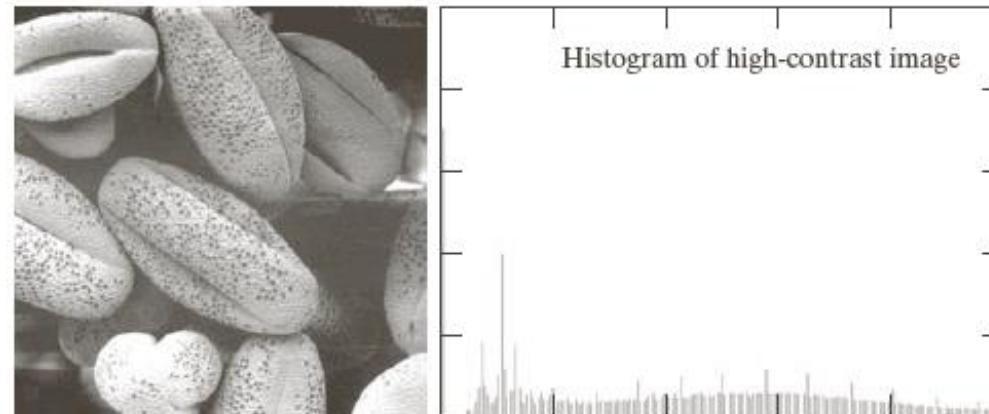


Histograms

- Low Contrast



- High Contrast



Normalised Histograms

- A normalised histogram is a discrete function

$$p(r_k) = n_k / n$$

where:

- $n = \text{width} \times \text{height}$ is the total number of pixels in the image
- The bins in a normalised histogram sum to one
- Each bin gives the probability of the corresponding grey level appearing in the image
- The probabilistic interpretation is valuable in e.g. contrast enhancement and automatic thresholding

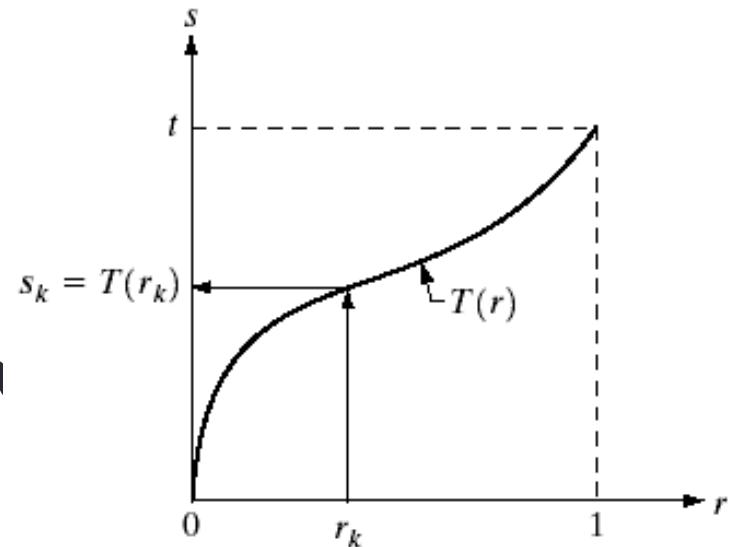
Histogram Equalisation

- Goal is to improve the contrast of an image
 - To transform an image in such a way that the transformed image has a nearly uniform distribution of pixel values
 - More general than linear or piecewise contrast stretching
- Histogram transforms
 - Map an input histogram r onto a new histogram s
 - Assume r has been normalized to the interval $[0,1]$, with $r = 0$ representing black and $r = 1$ representing white

$$s = T(r) \quad 0 \leq r \leq 1$$

Histogram Equalisation

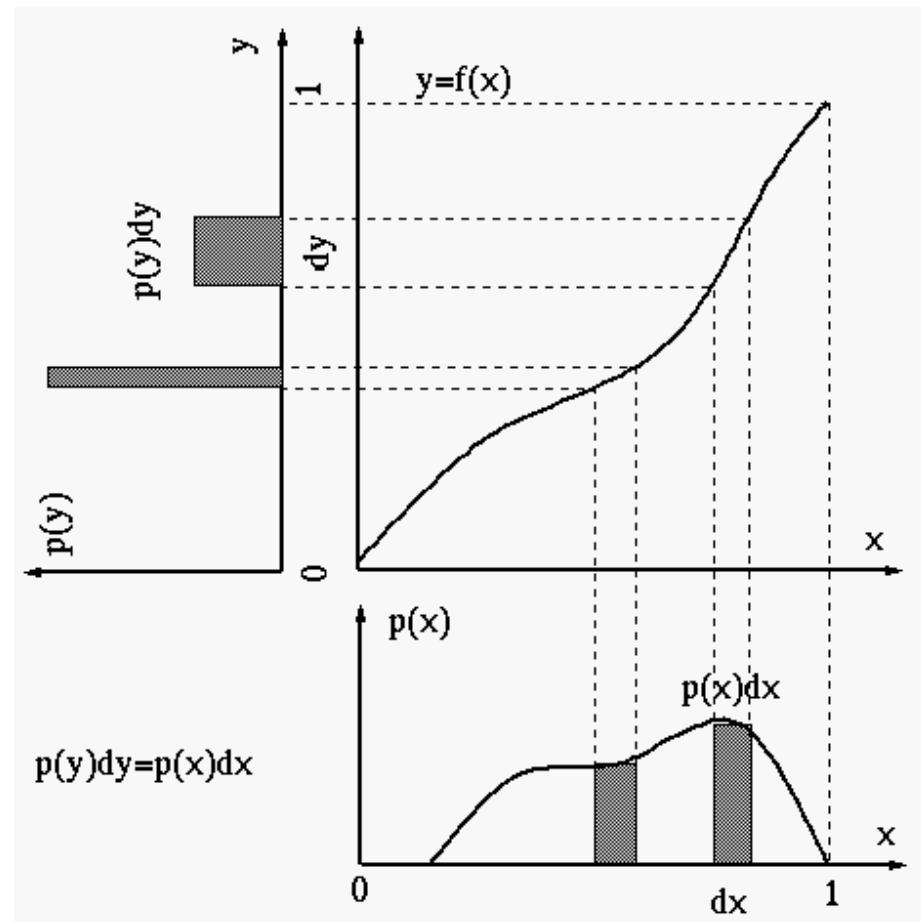
- The transformation function satisfies the following conditions
 - $T(r)$ is single-valued and strictly monotonically increasing in the interval $0 \leq r \leq 1$
 - $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$
- This means it is possible to invert the process
- It also gives us the relationship that allows the derivation of histogram equalisation



Histogram Equalisation

$$P(y).dy = P(x).dx$$

$$\text{So } P(y) = P(x) \frac{dx}{dy}$$



Histogram Equalisation

- In Gonzalez and Woods' notation....
- Let $p_r(r)$ and $p_s(s)$ denote the probability density function of random variables r and s , respectively
- If $p_r(r)$ and $T(r)$ are known, then the probability density function $p_s(s)$ of the transformed variable s can be obtained

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|$$

- $r = T^{-1}(s)$

Histogram Equalisation

- If we choose as the transformation function the cumulative distribution function or CDF:

$$T(r) = \int_0^r p_r(w) dw$$

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d}{dr} \int_0^r p_r(w) dw = p_r(r)$$

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| = p_r(r) \left| \frac{1}{p_r(r)} \right| = 1 \quad 0 \leq s \leq 1$$

$T(r)$ depends on $p_r(r)$, but the resulting $p_s(s)$ is always uniform.

Histogram Equalisation

- We have a discrete histogram, not a PDF of a continuous random variable
- The probability of occurrence of gray level r_k in an image is

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

- The transformation function is

$$s_k = T(r_k) = \sum_{j=0}^k p_r(r_j) = \sum_{j=0}^k \frac{n_j}{n} \quad k = 0, 1, 2, \dots, L - 1$$

- An output image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k .

In Practice

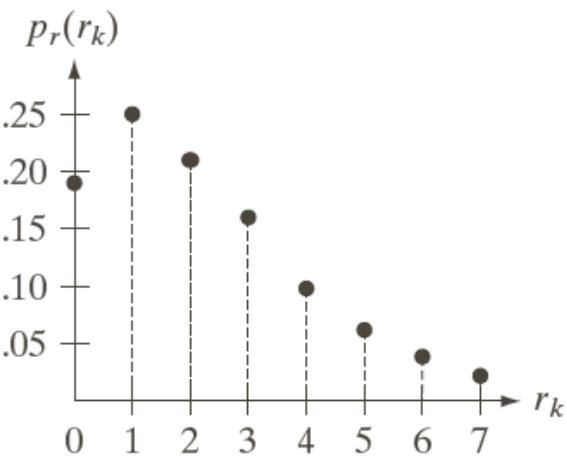
- To perform histogram equalisation
 - Compute the CDF of the input image.
 - For each pixel in the input image, the corresponding output pixel intensity is calculated by using the CDF as a look-up function.
 - CDF values will be in the range 0 - 1, scale the equalised image to fit the range supported by the output image format.
- The histogram of the output image will be approximately uniform

In Practice

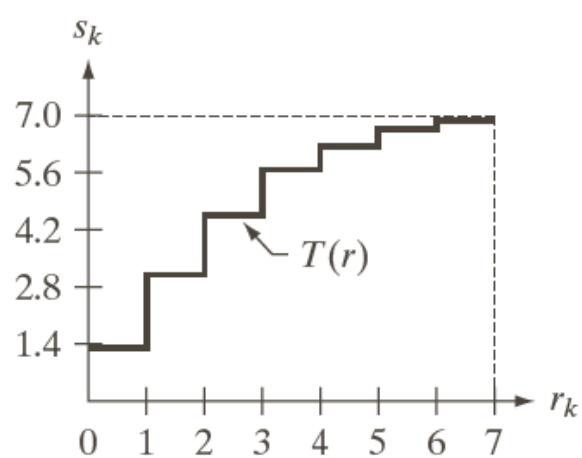
- Consider a 64×64 pixel, 3 bit (8 grey level) image
(Example from Gonzalez & Woods)

r_k	n_k	$P_r(r_k)$	$T(r_k)$	s_k	Round
0	790	0.19	0.19	1.33	1
1	1023	0.25	0.44	3.08	3
2	850	0.21	0.65	4.55	5
3	656	0.16	0.81	5.67	6
4	329	0.08	0.89	6.23	6
5	245	0.06	0.95	6.65	7
6	122	0.03	0.98	6.86	7
7	81	0.02	1.00	7.00	7

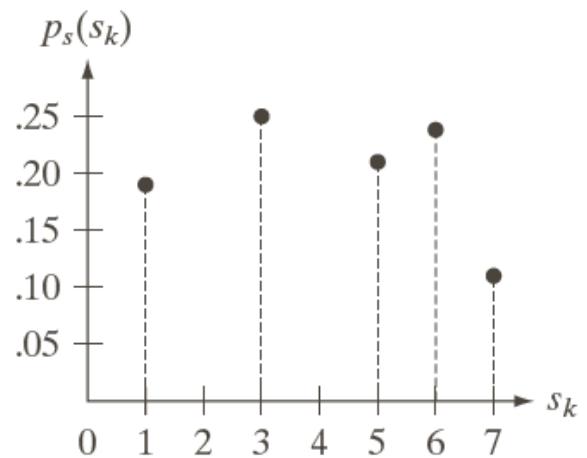
In Practice



Input histogram



$T(r_k)$, scaled back to
0 - 7



Equalised histogram

Results

- Light

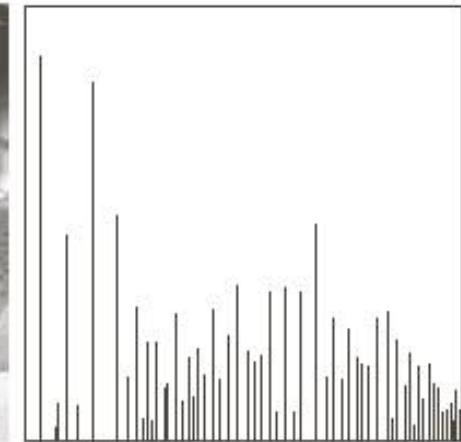
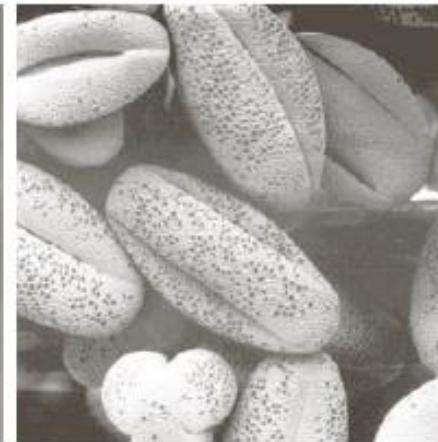


- Dark

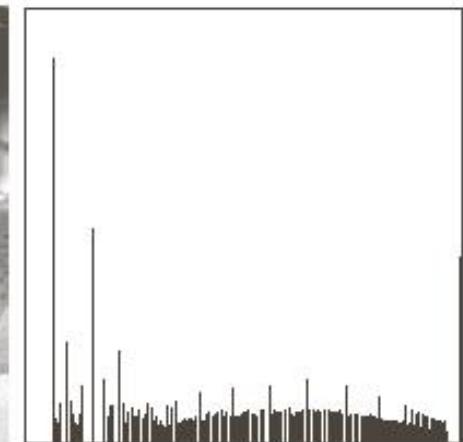


Results

- Low contrast

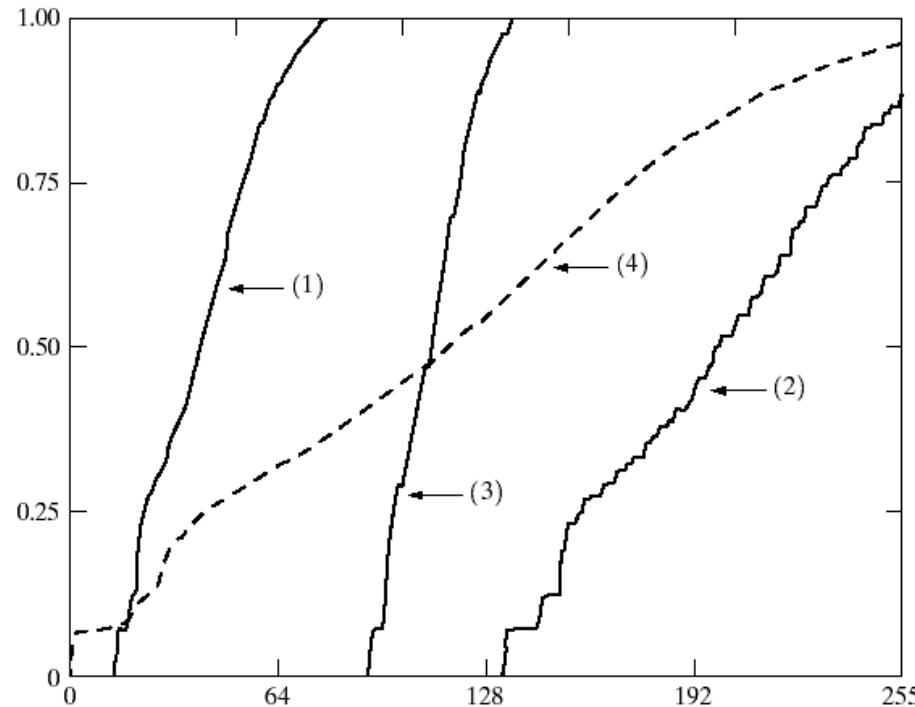


- High Contrast



Results

- Each of the four transformations above used a different transform, tuned to the input histogram



Strengths & Weaknesses

- Histogram equalisation works well when the input images
 - aren't too noisy
 - don't have large bright or dark areas



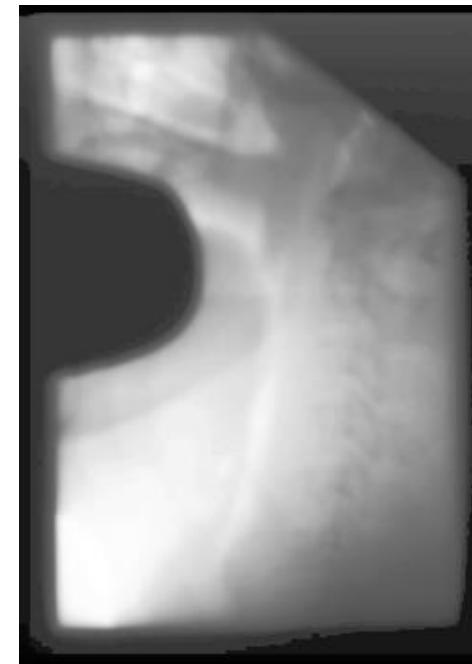
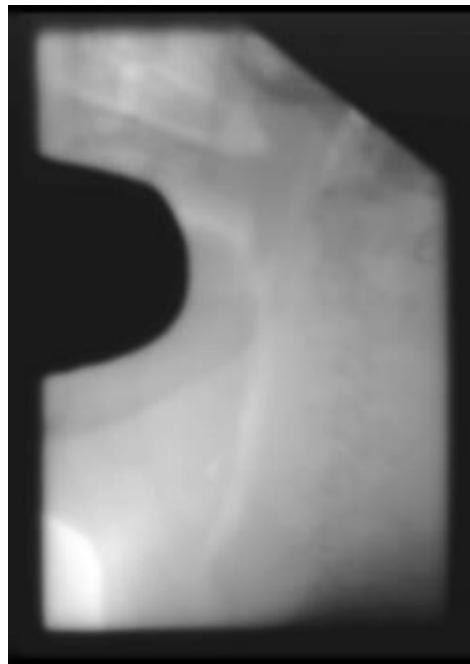
Strengths & Weaknesses

- Here the bright sky has dominated the process, equalisation has introduced an artificial boundary between sunlight and sky but not enhanced the three people

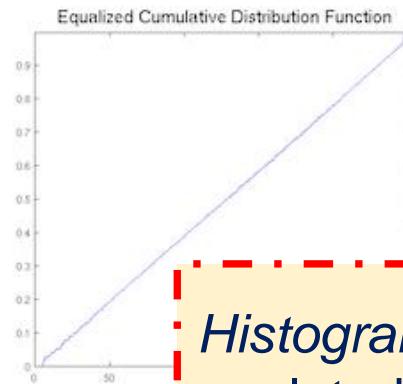
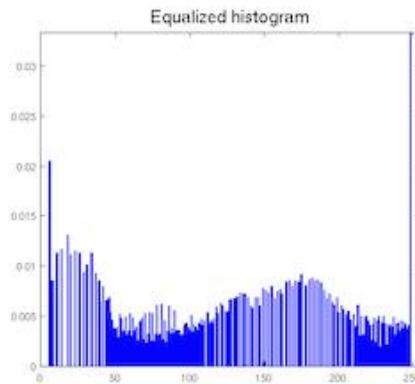
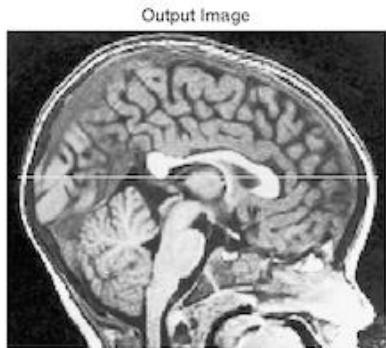
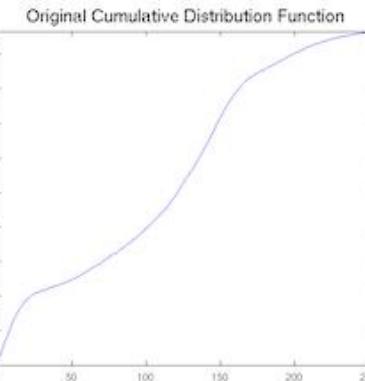
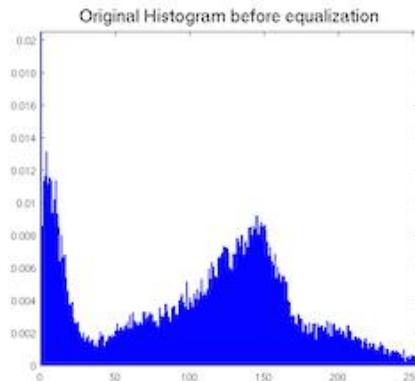
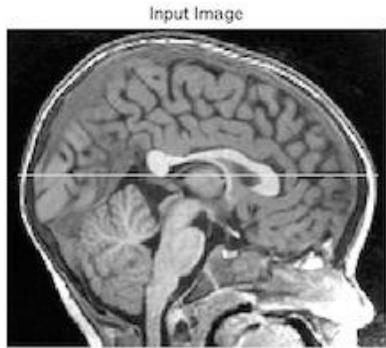


Strengths & Weaknesses

- Here the bright area of interest is enhanced, but the noise in the upper dark region is also more obvious



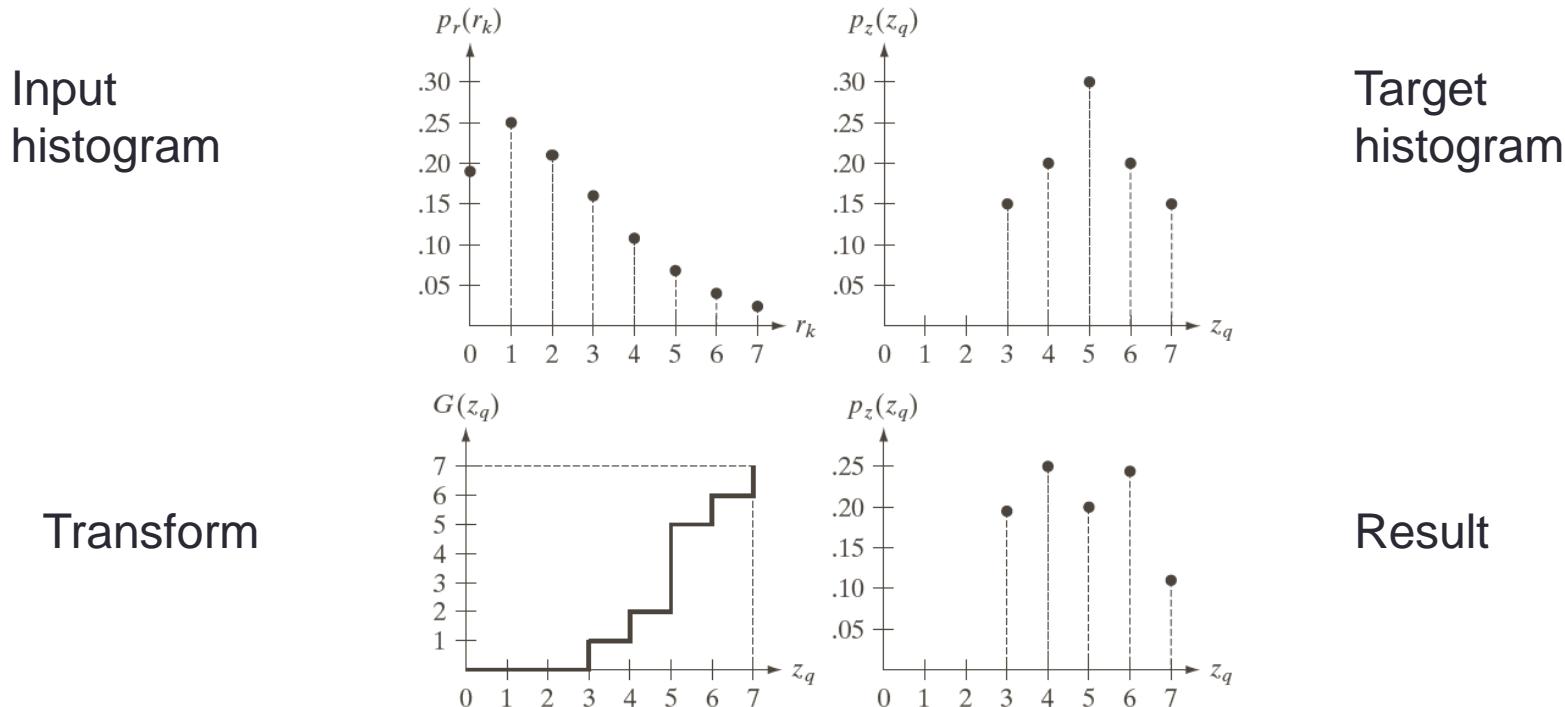
Medical Applications (MRI)



Histogram Specification is a related method which transforms an image's histogram so that it matches a target histogram

More Generally...

- Histogram Specification* is a related method which transforms an image's histogram so that it matches a target histogram



Conclusion

- Histograms, particularly normalised histograms, provide useful summary information
- Images can be manipulated by manipulating their histograms
- Histogram equalisation is a powerful and widely used image enhancement operation
- Global equalisation has drawbacks which can be addressed using local processing
- The method generalises to Histogram Specification

PART 2 : APPLICATIONS

Image Matching with Colour Histograms

Storing & Retrieving Images

- Given a large image database, find all the images containing e.g. horses
- We will focus on individual images, but many of the problems & methods discussed extend to video databases



Text-based Approaches

- Annotation: Relevant words/phrases are added to each image
- Retrieval is via text search
- But annotation is
 - subjective
 - laborious
 - unnecessary????

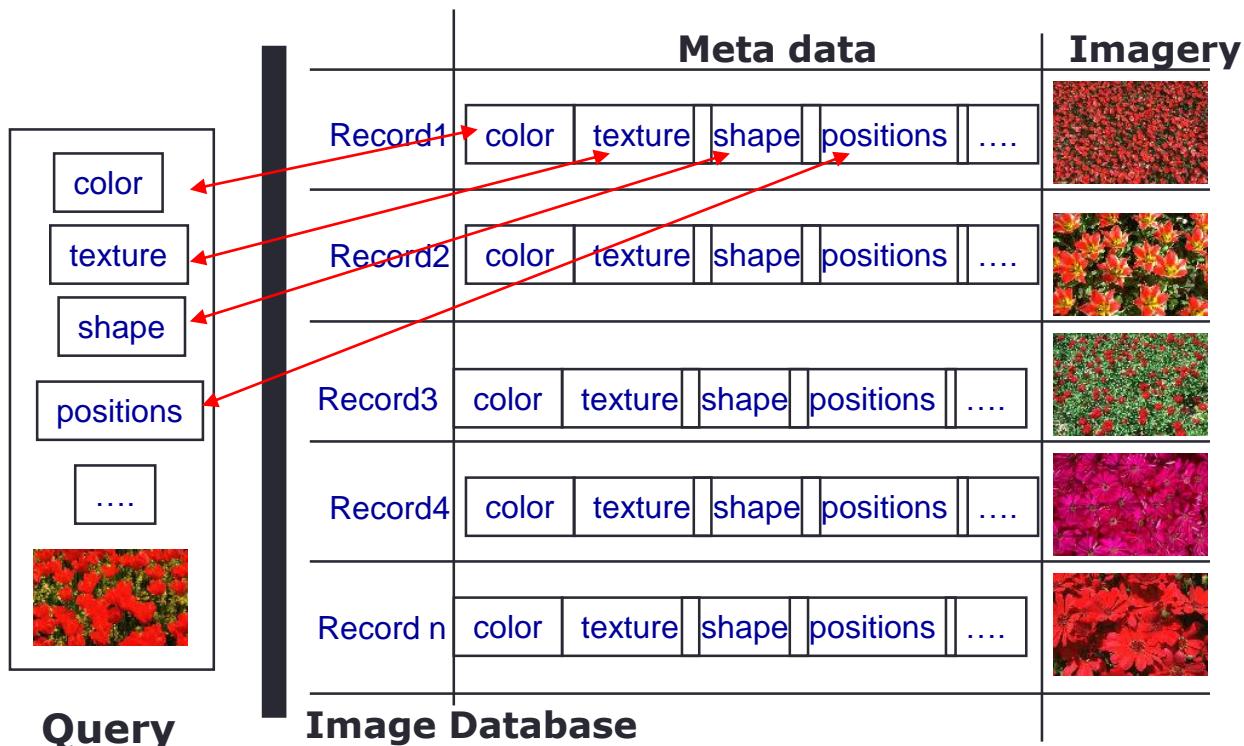


Mother, Child, Vegetable,
Yellow, Green, Purple ...

Content-based Retrieval

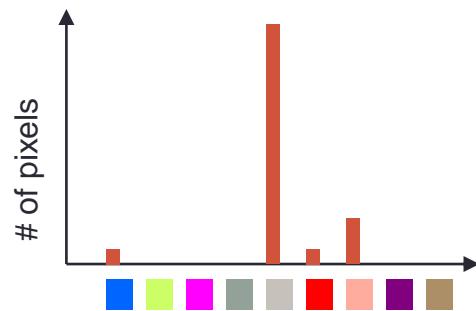
- Indexes the image database on visual features
 - colour
 - shape
 - Texture
 - Queries are expressed in those terms or via visual examples
- 
- Simple approaches compute metric distances between the query image and each image in the database
 - Advanced approaches use AI techniques, machine learning, etc., and may be interactive
 - Some simple measures and datastructures can be very useful

Content-based Retrieval

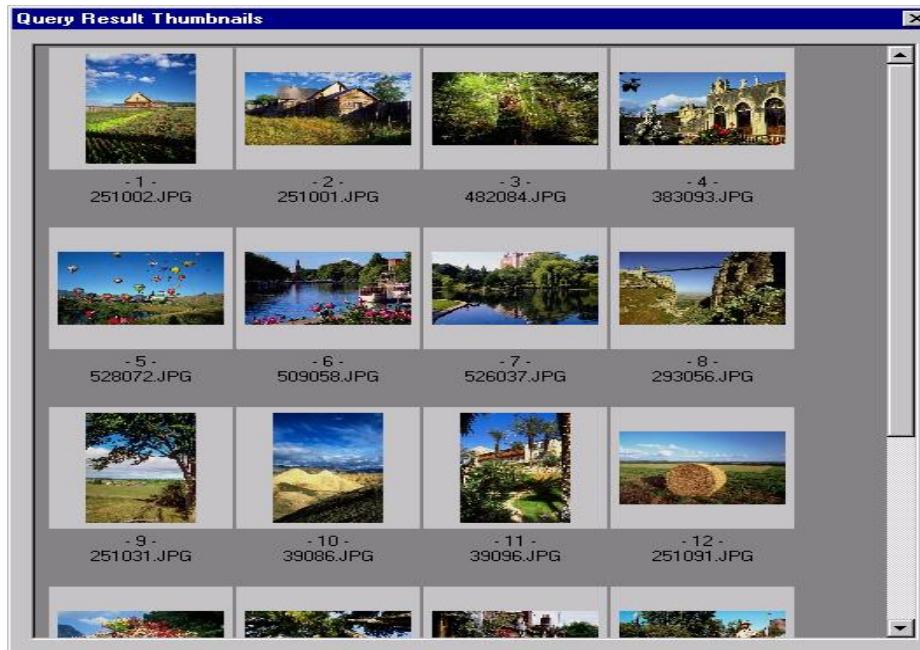


Colour Histograms

- Choose a colour space
 - RGB, HSV,....
- Divide the axes to create a reasonable number of divisions
 - Trade-off detail against memory/computational cost
- Build a histogram
- Normalise if images are different sizes or colour resolution



Why Colour Histograms?



- Images with similar colour distributions look similar:
colour distribution = colour histogram

Why Colour Histograms?

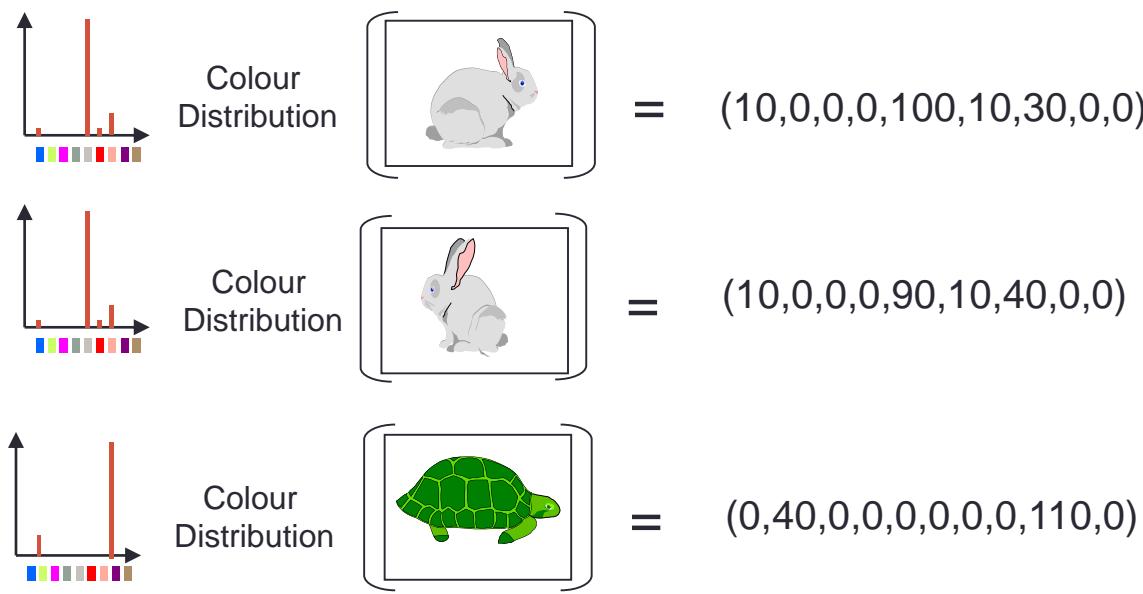
- Colour correlates well with class identity



- Human vision works hard to preserve colour constancy: presumably because colour is useful

- Histograms
 - Are invariant to translation and rotation
 - Change slowly as viewing direction changes
 - Change slowly with object size
 - Change slowly with occlusion
- Colour histograms summarise target objects quite well, and should match a good range of images

Colour Histograms



- Points in a high dimensional space or compact representations of images?

Common Distance Metrics

- Euclidean or straight-line distance or L2-norm, D^2

$$D^2(H^1, H^2) = \sqrt{\sum_i (H_i^1 - H_i^2)^2} = \|H^1 - H^2\|_2$$

*Root-mean square
error*

- City-block distance or L1-norm, D^1

$$D^1(H^1, H^2) = \sum_i |H_i^1 - H_i^2| = \|H^1 - H^2\|_1$$

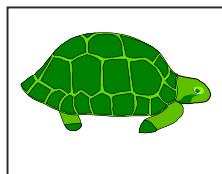
*sum of absolute
differences*

- *How far apart are two points?*

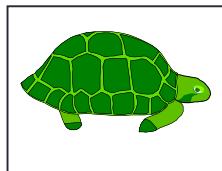
Some Problems

- Colour quantisation

- Noise and/or different camera responses can give similar images very different histograms



(0,40,0,0,0,0,0,110,0)



(0,0,40,0,0,0,0,0,110)

- Histogram resolution

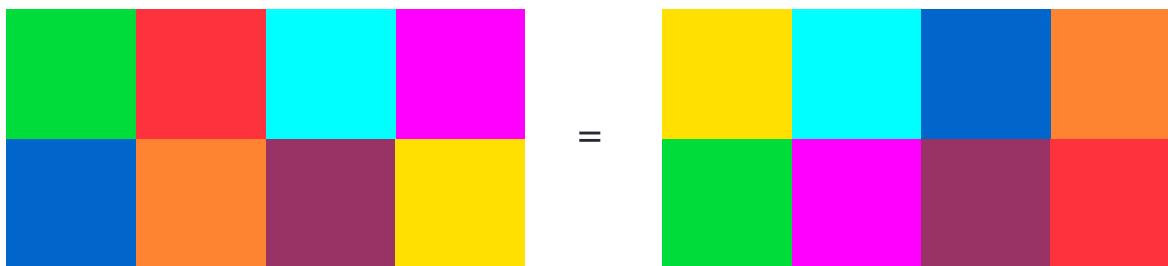
- May need many bins (4096) to accurately store colour distributions
 - Expensive

- The illumination may be coloured

- Same object may generate a different histogram under different lighting

Some Problems

- Colour histograms ignore spatial information
- More advanced methods take spatial relationships into account



- But the comparatively simple image processing methods and representations methods covered so far can do useful things

Histogram Intersection

- Measures how much of the query may be present in the target image (and vice-versa)
- A bin in the target histogram can have a larger value than the corresponding query bin (and vice-versa)

$$\text{HI}(H^1, H^2) = \sum_{i=1}^n \min(H_i^1, H_i^2)$$

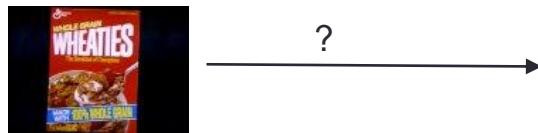
$H^1 = (10, 0, 0, 0, 100, 10, 30, 0, 0)$
 $H^2 = (0, 40, 0, 0, 0, 6, 0, 110, 0)$

$$\begin{aligned}\text{HI}(H^1, H^2) &= 0 + 0 + 0 + 0 + 0 + 6 + 0 + 0 \\ &= 6\end{aligned}$$

- *How much do two representations overlap?*

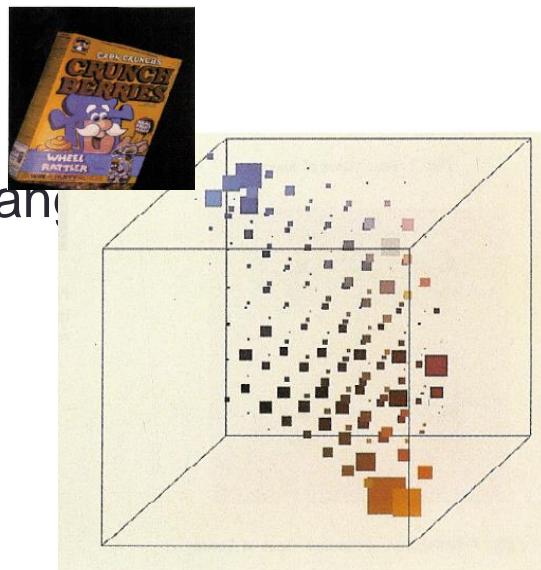
Histogram Intersection

- In the first histogram intersection paper (Ballard and Swain 1991):
 - A database of 66 colour histograms
 - 32 query images
- Recognition rate was almost 100%



Histogram Intersection

- Ballard and Swain used opponent colour axes
 - $RG = R - G$
 - $BY = 2 * B - R - G$
 - $WB = R + G + B$ (intensity)
 - & matched images under a range of conditions
 - Normal condition
 - Varying in view
 - Varying in image resolutions
 - Occlusion (of bottom 1/3 and/or side 1/3 of image)
 - Varying in bin resolutions
 - Varying in light intensity



Histogram Intersection

Condition	Placement			
	1st	2nd	3rd	>3rd
Full size	29	3	0	0
128 x 90	29	3	0	0
64 x 45	27	5	0	0
32 x 22	14	7	1	0
16 x 11	15	6	4	0
8 x 5	4	4	3	21
Bottom occluded	27	4	1	0
Bottom & side occluded	22	6	5	0

Good until
images are
very small



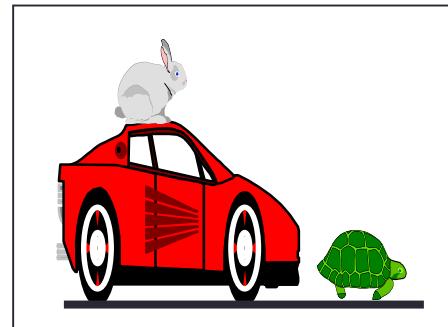
Not bad when
object only
partly visible



Using Histograms: Object Location

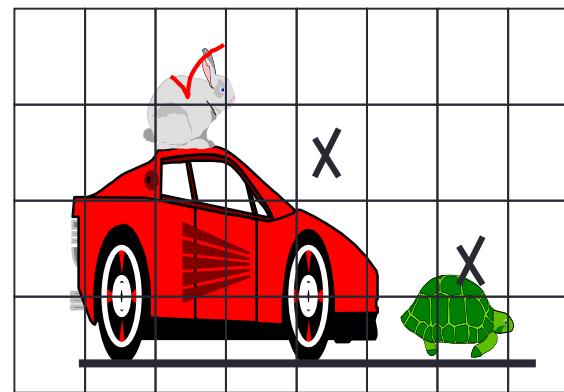
Object Location

- Matching whole images isn't always appropriate.....
 - e.g. if the target object is only expected to fill part of the image or you want to know where it is



Region/Object-based Queries

- We know the rabbit is mostly grey
- Divide the image into windows and see how grey each window is
 - Highlight pixels in the image that are similar to those in the query
 - Look for regions with lots of these pixels



✓ Similar colours
✗ Dissimilar colours

The Histogram Ratio

- BUT: the image is usually much bigger than the query region

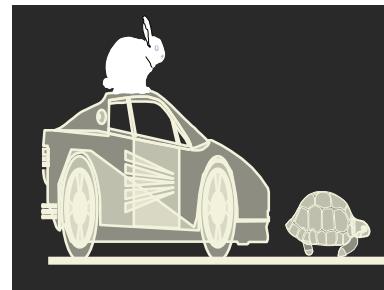
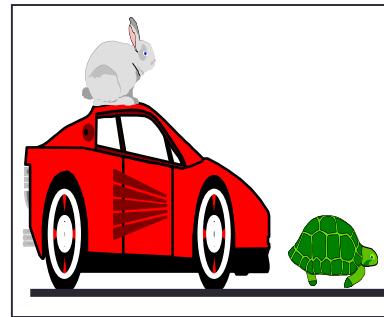
- Other objects in the image may be the same colour as parts of the query object
- So some colours are not reliable cues: if you're looking for a zebra at night look for white pixels

$$R_j = \min\left(\frac{M_j}{I_j}, 1\right)$$

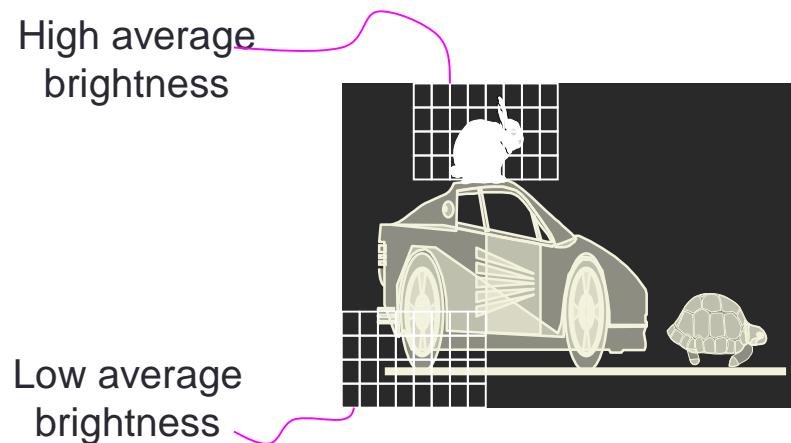
- Compute the ratio of corresponding Model and Image histogram bins
- If the image has many more pixels of a given colour R_j is small and that colour is not useful
- If the model has more, R_j is 1 and that colour is useful

Backprojection

- The greater the value of R_j , the more valuable the colour(s) represented by bin j
 - Consider each image pixel
 - If that pixel maps to histogram bin k , replace the pixel value with a grey value = R_k
- This is still image processing: we've processed an image to create an image
 - The pixel values are related to the likelihood of their showing the target



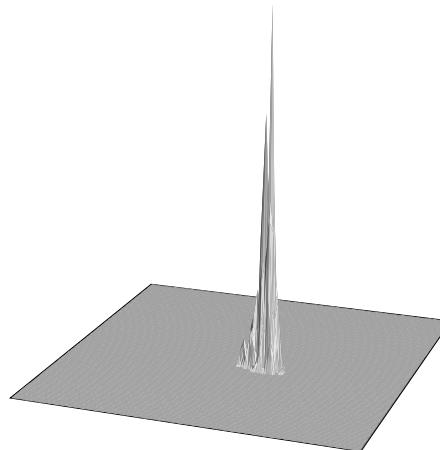
Backprojection



- Regions with high average brightness are likely to contain the rabbit – its more complex, but still a histogram matching approach (see Ballard and Swain)

We've seen Backprojection before

- Skin shows a **very** clear peak and narrow range in U,V

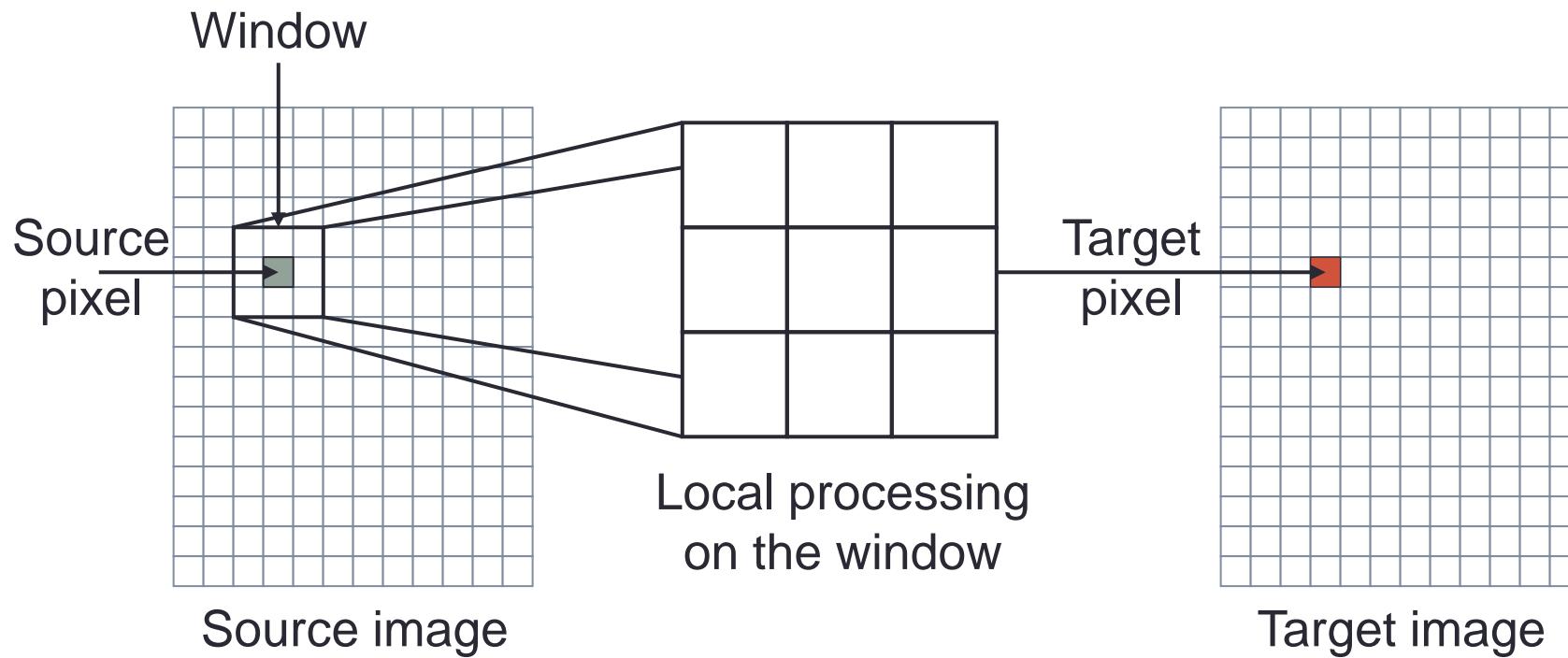


COMP2005

Linear Filters :

- Convolution, Mean Filtering and Noise
- Gaussian Filtering

Spatial Filtering



Why?

- Intensity Transforms read and affect only a single pixel, their power is limited
- Images are spatially organised data structures, many important attributes vary slowly across the image
 - Object identity
 - Viewed surface orientation, colour, etc
 - Illumination
- Processes restricted to a small, compact area have access to more information but are still likely to consider a single object, surface, illumination pattern, etc.

Image Noise

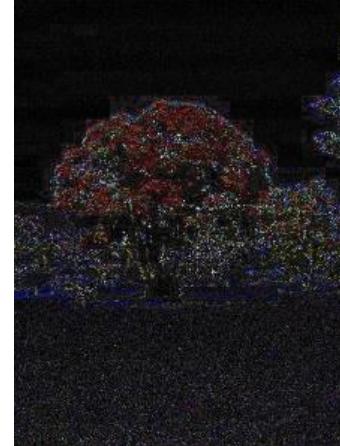
- Noise = small errors in image values
- Imperfect sensors introduce noise
- Image compression methods are lossy:
repeated coding & decoding adds noise



Original



JPEG



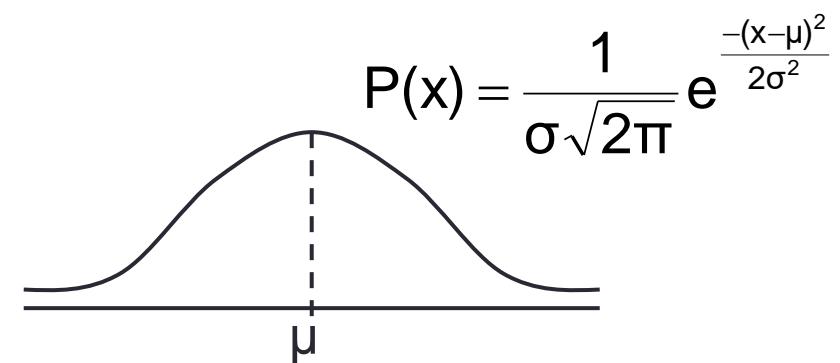
Difference
(Enhanced)

- Noise is often modelled as additive:

*Recorded value = true value +
random noise value*

Gaussian Noise

- Sensors often give a measurement a little off the true value
 - On **average** they give the right value
 - They tend to give values near the right value rather than far from it
- We model this with a **Gaussian**
 - Mean (μ) = 0
 - Variance (σ^2) indicates how much noise there is



Gaussian Noise

The level of noise is related to the Gaussian parameter σ



$$\sigma = 1$$



$$\sigma = 10$$



$$\sigma = 20$$

Image with varying degrees of Gaussian noise added

Noise Reduction

- If you have multiple images, taking the mean value of each pixel will reduce noise
 - Noise is randomly added to each value
 - Mean value added is 0
 - If you average a large set of estimates of the same pixel the random noise values will cancel out

42 43 44 41 40 42 42 44 40

→ 42

- Given only a single image, averaging over a local region has a similar effect

42 43 44

41 40 42 → 42

42 44 40

- Ideally, we would choose the region to only include pixels that should have the same value
- **We need a spatial filter.....**

Spatial Filtering: Convolution

- Many filters follow a similar pattern - multiplying each image value by a corresponding filter entry, and summing the results

$F_{(-1,-1)}$	$F_{(0,-1)}$	$F_{(+1,-1)}$
$F_{(-1,0)}$	$F_{(0,0)}$	$F_{(+1,0)}$
$F_{(-1,+1)}$	$F_{(0,+1)}$	$F_{(+1,+1)}$

Filter Window

$P_{(x-1,y-1)}$	$P_{(x,y-1)}$	$P_{(x+1,y-1)}$
$P_{(x-1,y)}$	$P_{(x,y)}$	$P_{(x+1,y)}$
$P_{(x-1,y+1)}$	$P_{(x,y+1)}$	$P_{(x+1,y+1)}$

Picture Window

$$\begin{aligned}
 & F_{(-1,-1)} \times P_{(x-1,y-1)} \\
 & + F_{(0,-1)} \times P_{(x,y-1)} \\
 & + F_{(+1,-1)} \times P_{(x+1,y-1)} \\
 & + F_{(-1,0)} \times P_{(x-1,y)} \\
 & + \dots \\
 & + F_{(+1,+1)} \times P_{(x+1,y+1)}
 \end{aligned}$$

Result

Filtering

- More generally, with a filter with radius r
 - $p_{x,y}$ is the original image value at (x,y)
 - $p'_{x,y}$ is the new image value at (x,y)

$$p'(x,y) = \sum_{dx=-r}^{+r} \sum_{dy=-r}^{+r} f_{dx,dy} \times p_{x+dx, y+dy}$$

- Many, though not all, filters work this way, e.g. **the mean filter:**

3×3
mean filter

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

5×5
mean filter

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

The Mean Filter



Original



Gaussian



Key Points

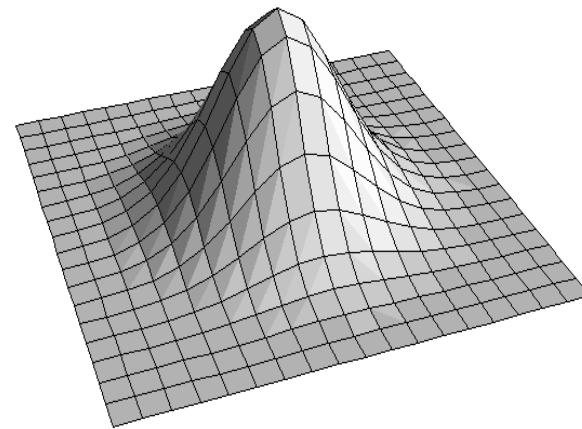
- Spatial filters operate on local image regions
- Many can be formulated as convolution with a suitable mask
- Noise reduction via mean filtering is a classic example

COMP2005

Gaussian Filtering

Gaussian Filters

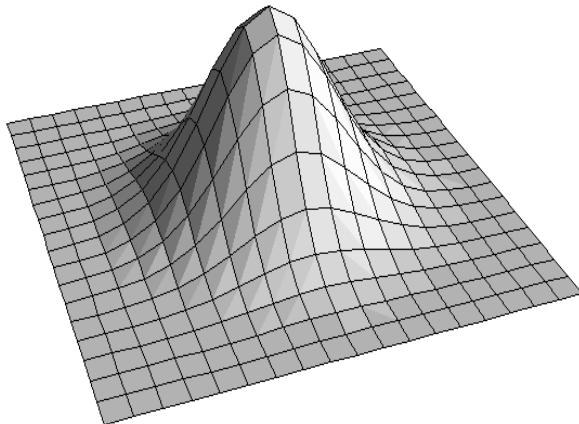
- Convolution with a mask whose weights are determined by a 2D Gaussian function
- Higher weight is given to pixels near the source pixel
- These are more likely to lie on the same object as the source pixel



$$P(x,y) = \frac{1}{\sigma^2 2\pi} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Discrete Gaussian Filters

- The Gaussian
 - Extends **infinitely** in all directions, but we want to process just a **local** window
 - Has a volume underneath it of 1, which we want to maintain
 - We can approximate the Gaussian with a discrete filter
 - We restrict ourselves to a square window and **sample** the Gaussian function
 - We **normalise** the result so that the filter entries add to 1



Example

- Suppose we want to use a 5×5 window to apply a Gaussian filter with $\sigma^2 = 1$
 - The centre of the window has $x = y = 0$
 - We sample the Gaussian at each point
 - We then normalise it

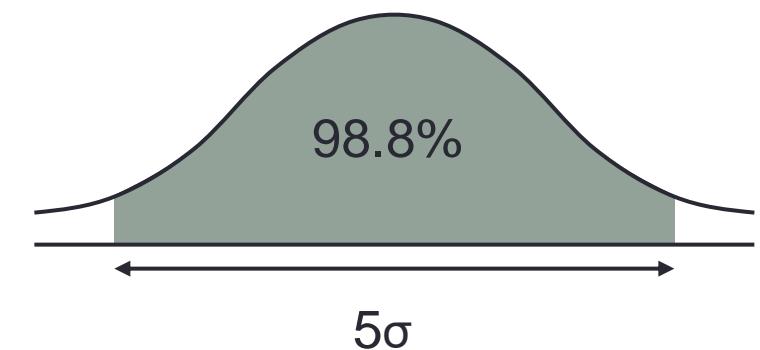
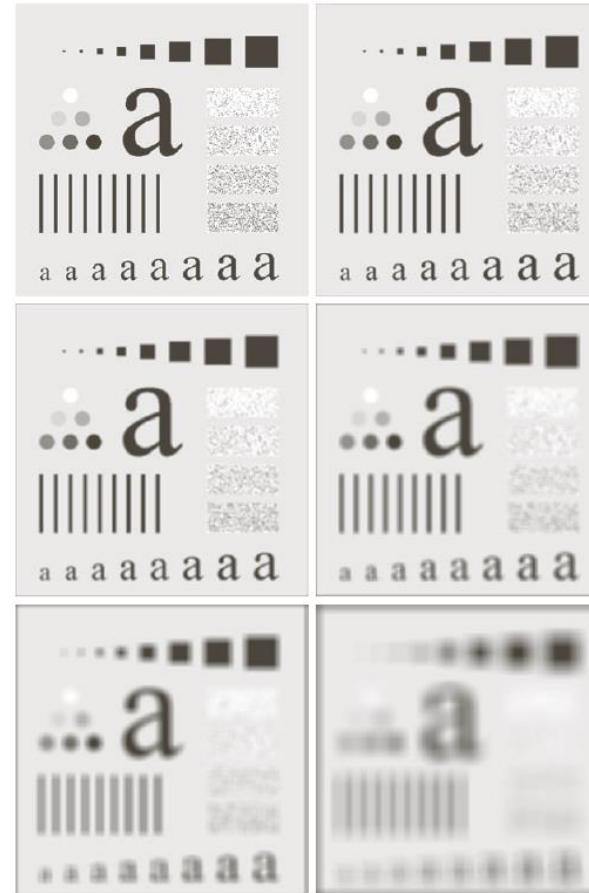
-2	0.00	0.01	0.02	0.01	0.00
-1	0.01	0.06	0.10	0.06	0.01
0	0.02	0.10	0.16	0.10	0.02
1	0.01	0.06	0.10	0.06	0.01
2	0.00	0.01	0.02	0.01	0.00

-2 -1 0 1 2

$\times \frac{1}{0.96}$

Gaussian Filters

- How big should the filter window be?
 - With Gaussian filters this depends on the variance (σ^2)
 - Under a Gaussian curve 98% of the area lies within 2σ of the mean
 - A filter width of 5σ gives more than 98% of the values we want



The Gaussian Filter



Original

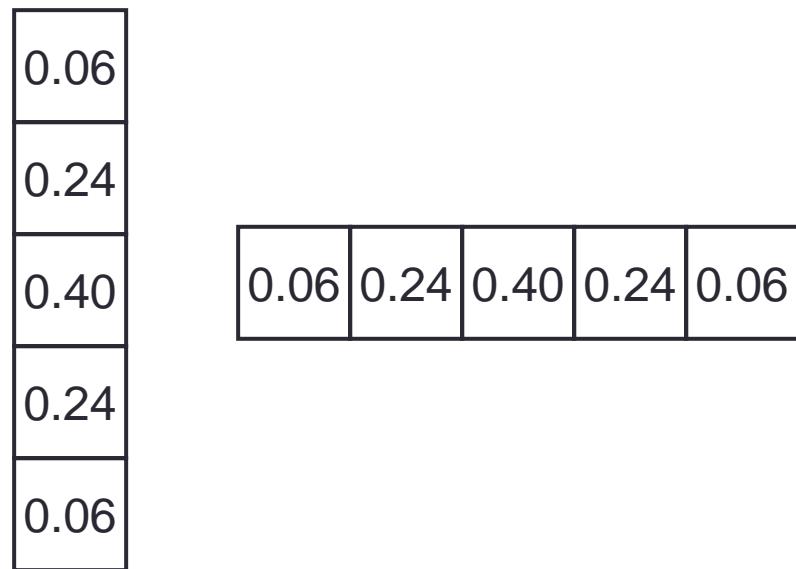


Gaussian



Separable Filters

- The Gaussian filter is separable
 - A 2D Gaussian is equivalent to two 1D Gaussians
 - First you filter with a ‘horizontal’ Gaussian
 - Then with a ‘vertical’ Gaussian.....



$$\begin{aligned}
 P(x,y) &= \frac{1}{\sigma^2 2\pi} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \\
 &= \frac{1}{\sigma\sqrt{2\pi}} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} e^{\frac{-y^2}{2\sigma^2}} \\
 &= \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \right) \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-y^2}{2\sigma^2}} \right) \\
 &= P(x) \times P(y)
 \end{aligned}$$

Separable Filters

- The separated filter is more efficient
 - Given an $N \times N$ image and a $n \times n$ filter we need to do $O(N^2n^2)$ operations
 - Applying two $n \times 1$ filters to a $N \times N$ image takes $O(2N^2n)$ operations
- Example
 - A 600×400 image and a 5×5 filter
 - Applying it directly takes around 6,000,000 operations
 - Using a separable filter takes around 2,400,000 - less than half as many

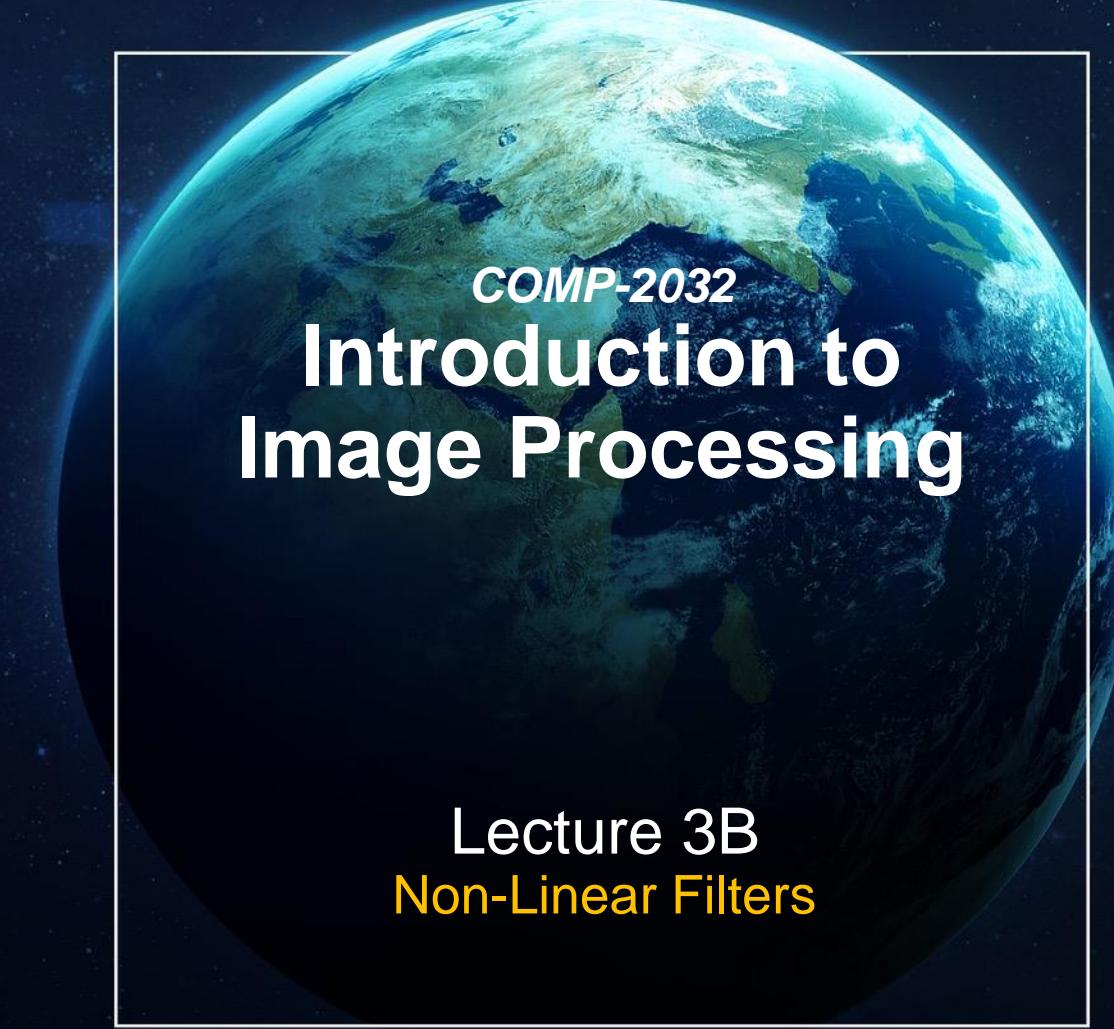
Key Points

- Like mean filtering, and Gaussian smoothing can be used to remove additive noise
- Gaussian smoothing emphasises pixels near the source pixel, where it is more likely image properties are fixed
- Gaussian smoothing is separable, and so efficient



University of
Nottingham

UK | CHINA | MALAYSIA

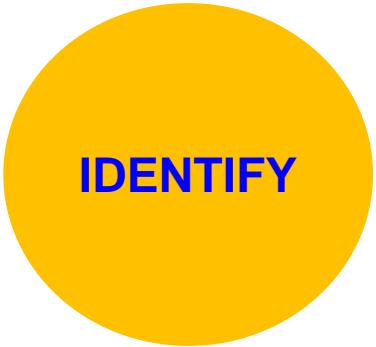


COMP-2032
**Introduction to
Image Processing**

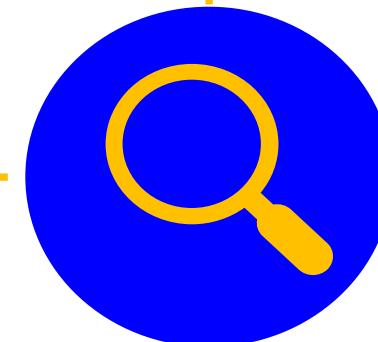
Lecture 3B
Non-Linear Filters



Learning Outcomes



- 1. Median Filtering
- 2. Anisotropic Diffusion
- 3. Bilateral Filtering





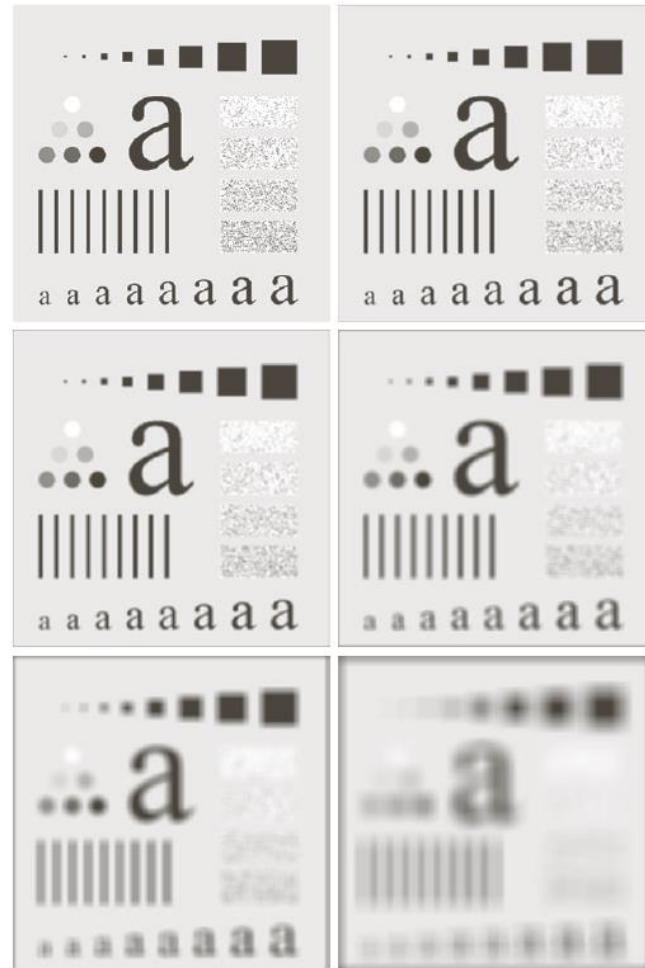
Non-Linear Filters

- Convolution with a mask of weights compute a linear function of a set of pixel values
- Many operations can be implemented this way, but not all:

- *Median filtering*
- *Anisotropic diffusion/Bilateral filtering*

Linear filters smooth sharp image changes, **nonlinear filters** tend to preserve or even enhance them

Difference



Gaussian Smoothing



Median Filtering

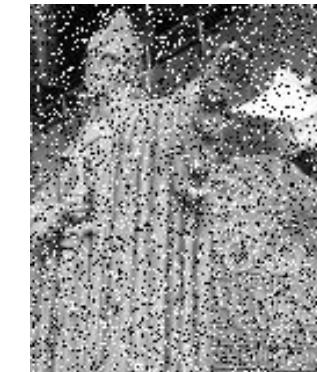


Salt and Pepper Noise

Sometimes sensors either fail to respond or saturate in error



- A **false saturation** gives a **white spot** in the image (**salt**)
- A **failed response** gives a **black spot** in the image (**pepper**)
- Sometimes called **speckle noise**



1%

10%

20%

An image with varying amounts of salt and pepper noise added



Reducing Salt and Pepper Noise





The Median Filter

Alternative: Median Filter

- Statistically the median is the middle value in a set
- Each pixel is set to the median value in a local window
- Result is a real pixel value, not a combination
- Noise pixels are outliers
- Noise would have to affect $>1/2$ the pixels to appear in the output

123	124	125
129	127	9
126	123	131

123	124	125	129	127	9	126	123	131
-----	-----	-----	-----	-----	---	-----	-----	-----

Find the values in a local window

9	123	123	124	125	126	127	129	131
---	-----	-----	-----	-----	-----	-----	-----	-----

Sort them

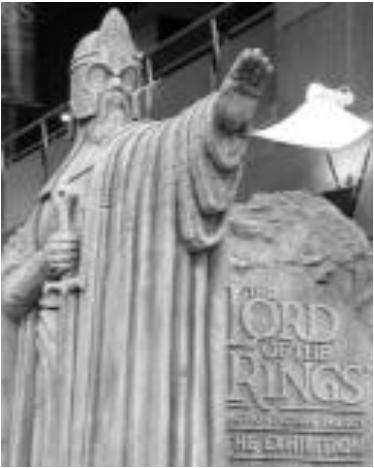
9	123	123	124	125	126	127	129	131
---	-----	-----	-----	-----	-----	-----	-----	-----

Pick the middle one

A mean filter would give 113



The Median Filter



Original



Salt & Pepper



Gaussian



ACK: Prof. Tony Pridmore, UNUK



Anisotropic Diffusion



Anisotropic Diffusion

- Median filtering is good given small regions of speckle noise, less good if edges are important
- There exist explicit edge-preserving smoothing ops

Diffusion

- Spreading out
- Mean and Gaussian filters can be seen as diffusion processes

Anisotropic

Not the same in all directions

Basic IDEA

- Mean and Gaussian filters make each pixel more like its neighbours
- Anisotropic diffusion makes each pixel more like those neighbours that it is already similar to



Anisotropic Diffusion

We have a **similar function**, $s(p,q)$

- $s(p,q)$ has values in the range from 0 to 1
- If the pixels p and q are **similar** then $s(p,q)$ is close to 1
- If the pixel p and q are **different** then $s(p,q)$ is close to 0

We use $s(p,q)$ to compute a weighted average of pixel values

- The new value at a pixel p , is based on all its neighbours, q

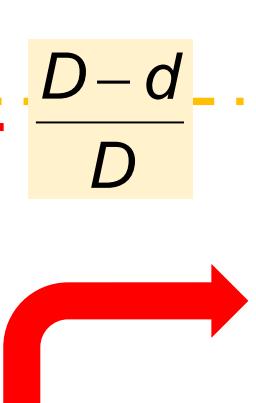
$$p' = \frac{\sum q \times s(p,q)}{\sum s(p,q)}$$



The Similarity Function

- The smoothing function, $s(p,q)$ needs to be found
- If d is the difference between p and q and D is the maximum possible difference we can use:

$$\frac{D-d}{D}$$


$$s(p,q) = e^{\left(\frac{p-q}{K}\right)^2}$$

Other functions often used include:


$$s(p,q) = \frac{1}{1 + \left(\frac{p-q}{K}\right)^2}$$

K determines the amount of smoothing



Anisotropic Diffusion

The examples here used the similarity function:

$$s(p,q) = \frac{1}{1 + \left(\frac{p - q}{K} \right)^2}$$



With $K = 25$



Salt & Pepper



Gaussian





Anisotropic Diffusion

A higher value of **K** gives greater smoothing, but
edges are still (quite) sharp



$K = 5$



$K = 25$



$K = 50$



$K = 100$



Anisotropic Diffusion

We can apply the filter repeatedly to give greater smoothing



Original



1 Iteration



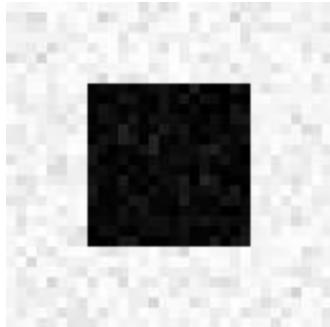
2 Iteration



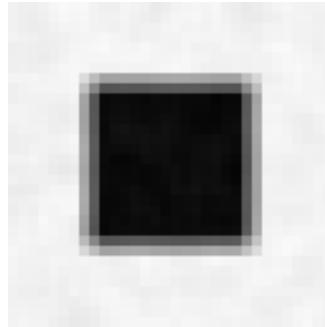
3 Iteration



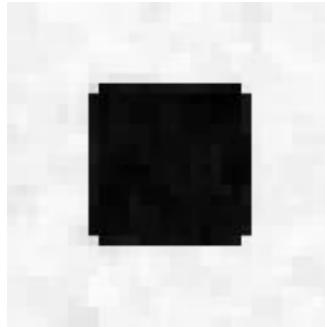
Reducing Noise near Edges



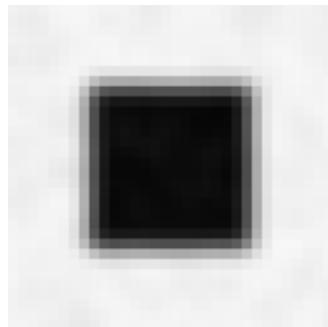
Noise Image



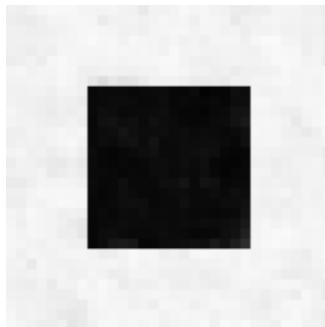
Mean



Median



Gaussian



Anisotropic Diffusion

|-----|
| Anisotropic diffusion
| is to mean filtering
| as ??? Is to
Gaussian filtering...



Bilateral Filtering



Bilateral Filtering

- Anisotropic Diffusion is related to mean filtering
- If the similarity function is always 1 we get a mean filter

$$p' = \frac{\sum q \times s(p,q)}{\sum s(p,q)}$$

Sums pixel values in a region

Counts pixel values in a region

Bilateral filters modify Gaussian smoothing in a similar way

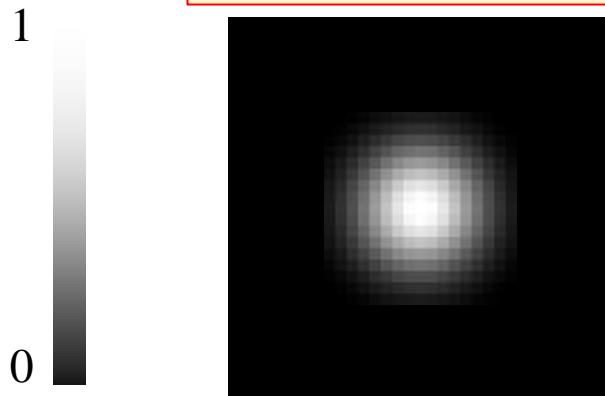
- One Gaussian weights pixels that are near the source
- Another Gaussian weights pixels that have similar intensity to the source pixel



Gaussian Smoothing Again

$$GB[I]_p = \sum_{q \in S} G_\sigma(\| p - q \|) I_q$$

normalized
Gaussian function





Bilateral Filtering

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(|I_p - I_q|) I_q$$

The equation is annotated with three colored arrows pointing to specific components:

- A pink arrow points to $\frac{1}{W_p}$, labeled "normalization factor".
- An orange arrow points to $G_{\sigma_s}(\| p - q \|)$, labeled "space weight".
- A blue arrow points to $G_{\sigma_r}(|I_p - I_q|)$, labeled "range weight".

Below the annotations is a diagram showing a grayscale image patch with a central white pixel, representing the space weight distribution. To the right is a plot of intensity I versus position, showing a wavy curve with a vertical double-headed arrow indicating the range filter's receptive field.

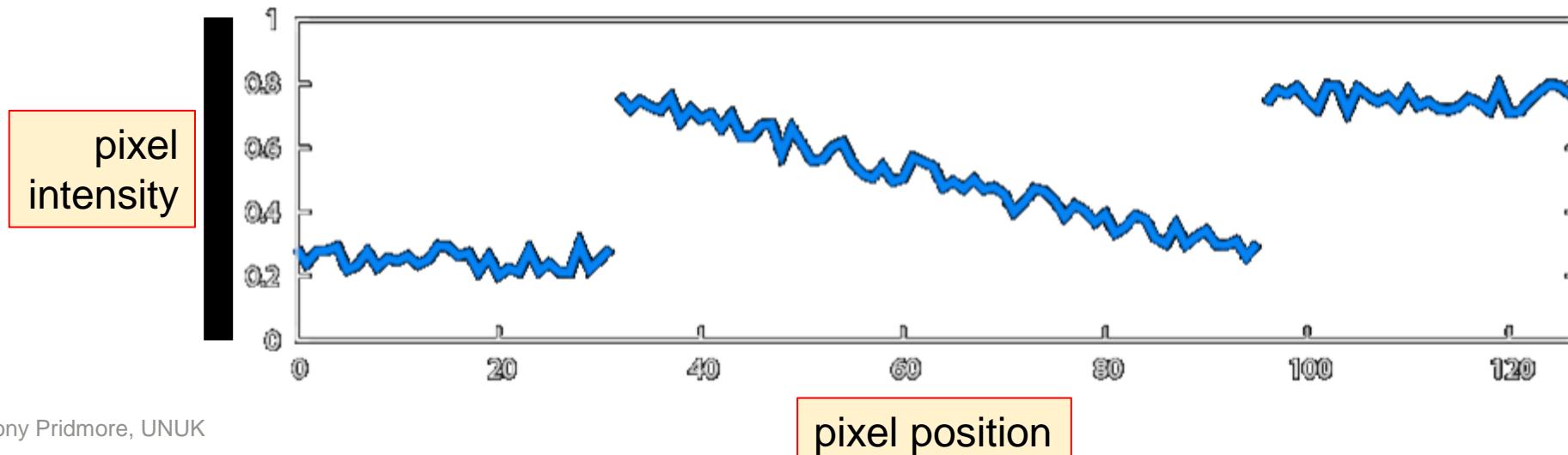


A One Dimensional Example

1D image = line of pixels



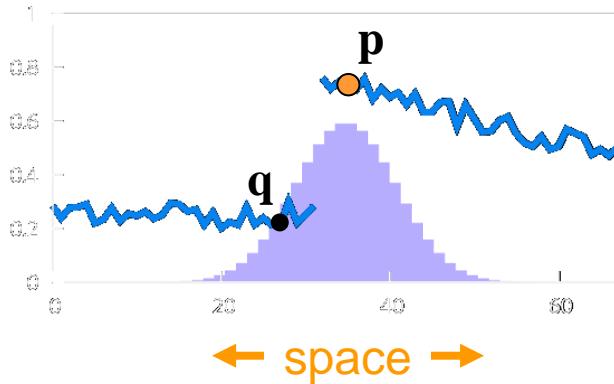
Better visualised as a plot



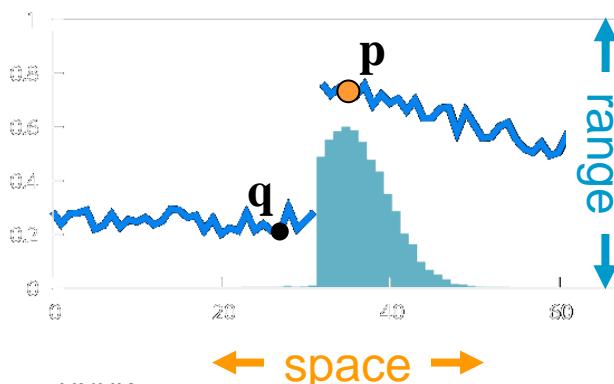


Gaussian & Bilateral Smoothing

Gaussian blur



Bilateral filter



$$GB[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q$$

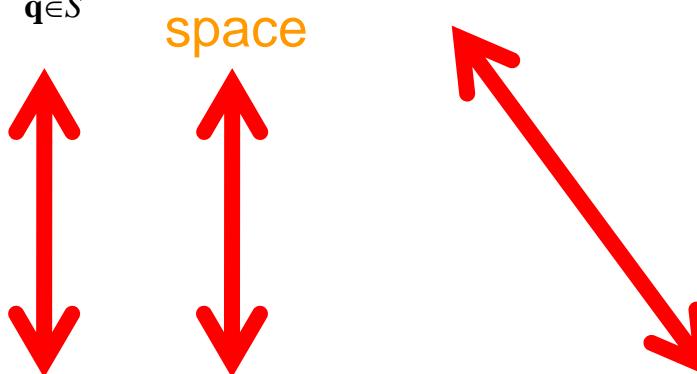
space

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

space

range

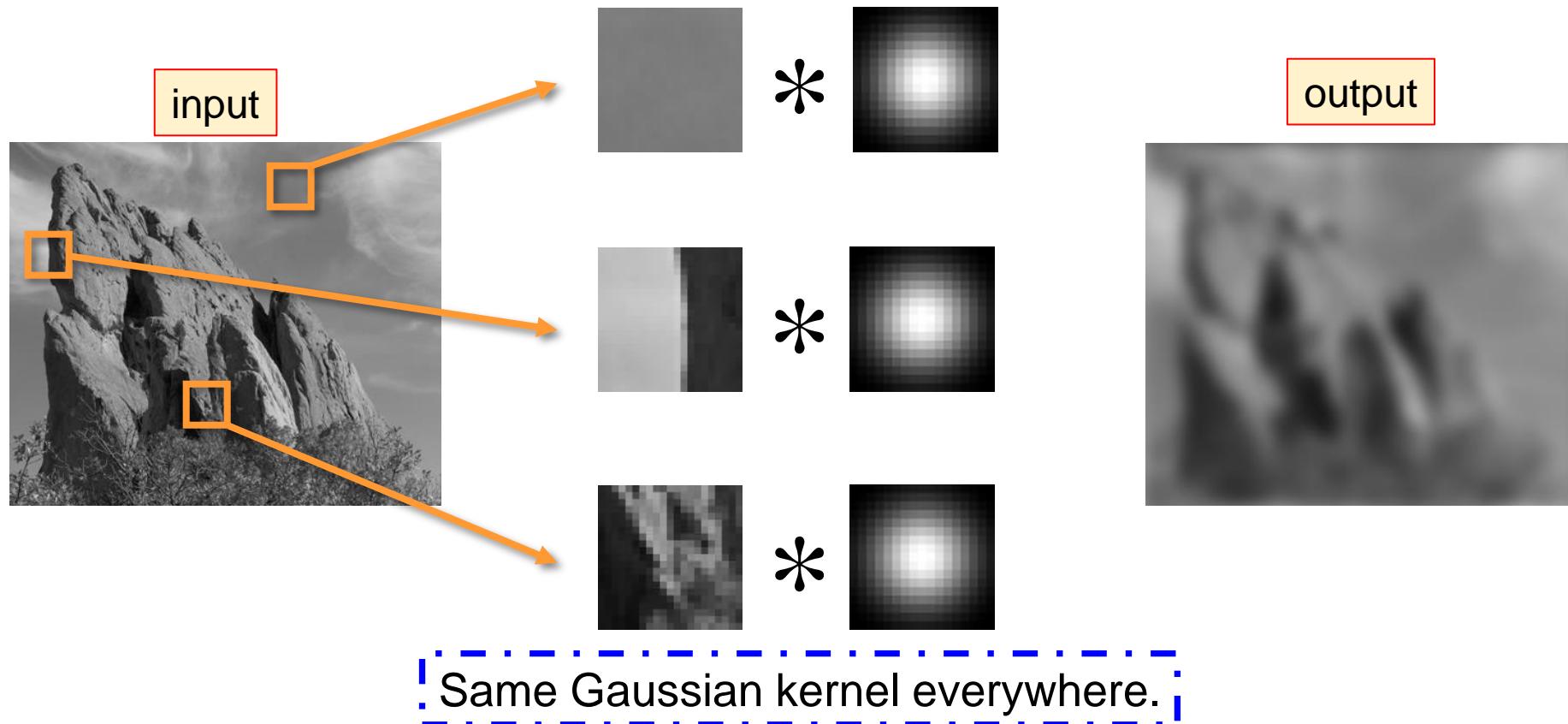
normalization



ACK: Prof. Tony Pridmore, UNUK

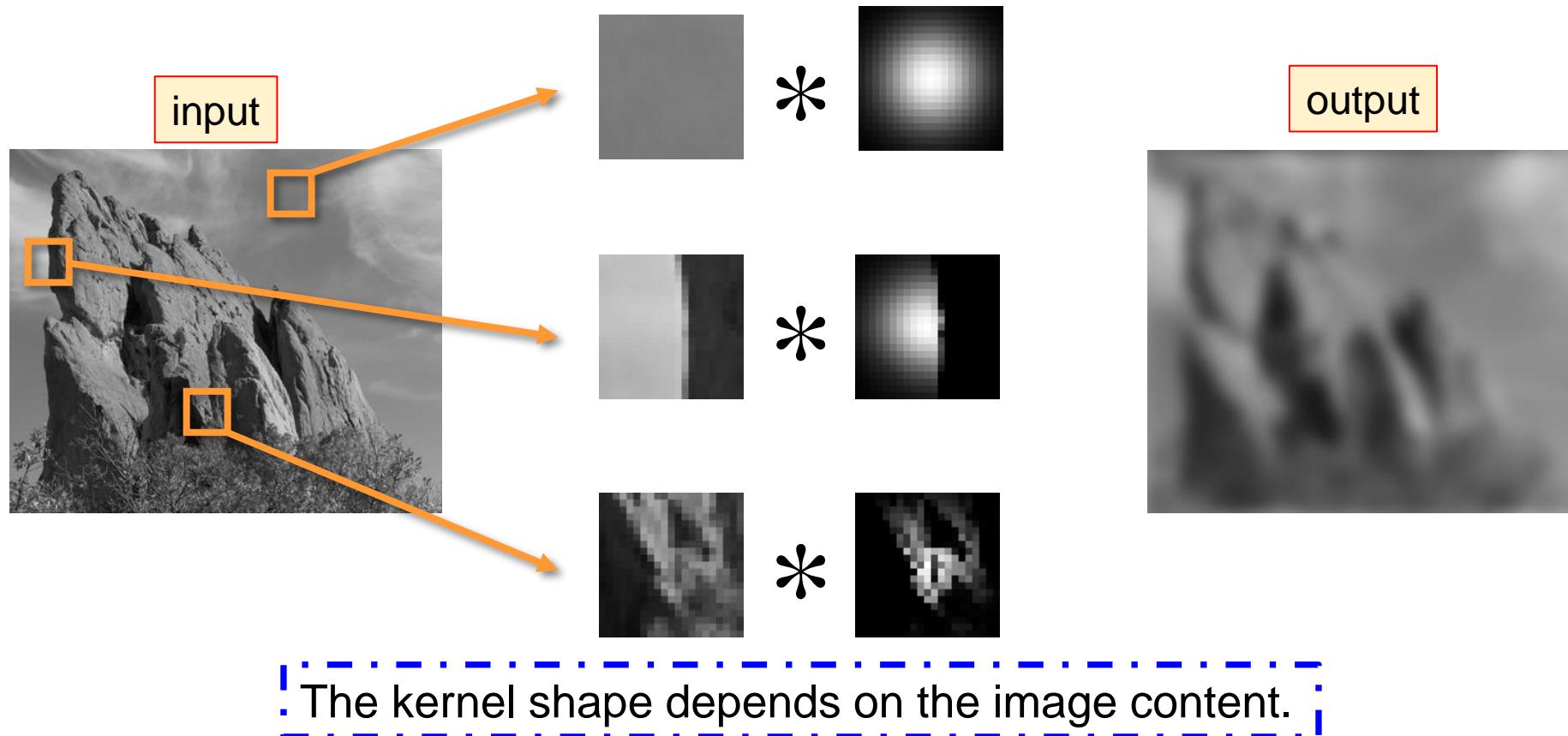


In 2D: Gaussian Smoothing



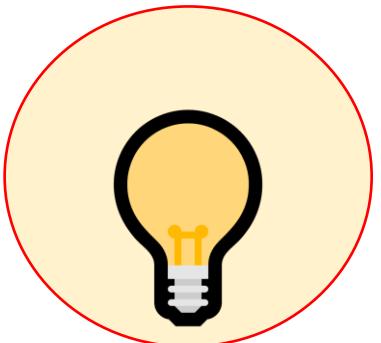


In 2D: Bilateral Filtering

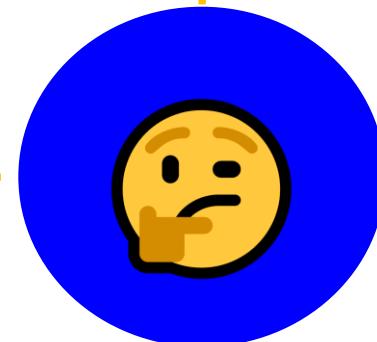




Summary



1. Median Filtering
2. Anisotropic Diffusion
3. Bilateral Filtering





University of
Nottingham

UK | CHINA | MALAYSIA

Questions



University of
Nottingham

UK | CHINA | MALAYSIA

NEXT:

Thresholding
&
Binary Images

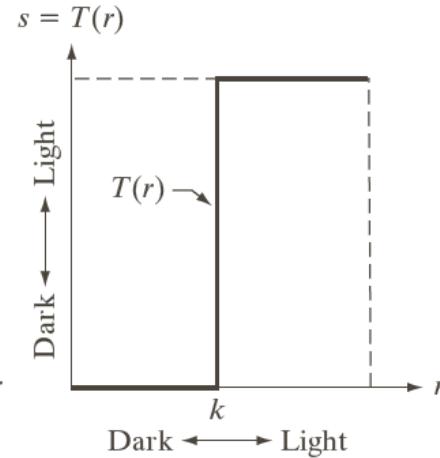
COMP2005

Thresholding & Binary Image Processing

1. What is Thresholding
2. Adaptive Thresholding

Binary Image Processing

- Many image processing operations make a decision:
 - is this the colour/object/edge I'm interested in?
- The result is a binary image
 - Pixels can have only two values: 0 or 1
 - Binary images also need noise removal, enhancement, etc.



Binarisation: Thresholding

- A dark object on a light background in a grey-level image
- Choose a threshold value, T
- Consider each pixel in turn
 - If the brightness at a pixel is less than T , that pixel is object
 - Otherwise it is part of the background
- Basic idea extends to colour; define sets of colour values that correspond to objects



Threshold, $T = 96$

Too Simple?

- The value of the threshold is very important
 - If it is too high, background pixels will be classified as foreground
 - If it is too low, object pixels will be considered background
- Assumes there are exactly two regions, with no overlap in their brightness – *is that true?*



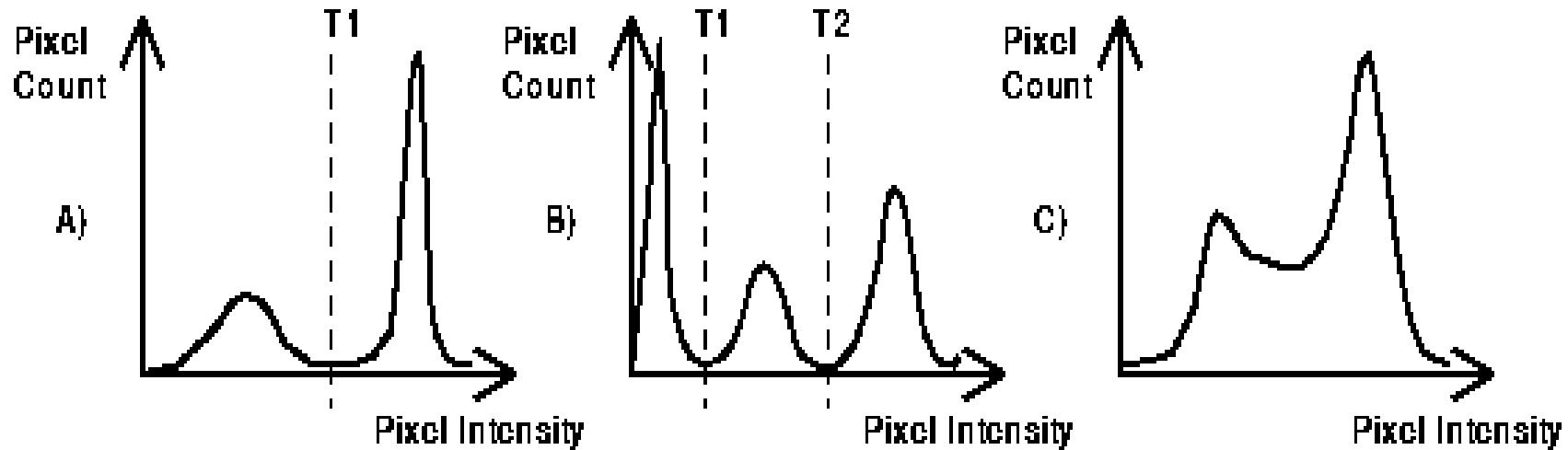
$T = 128$



$T = 64$

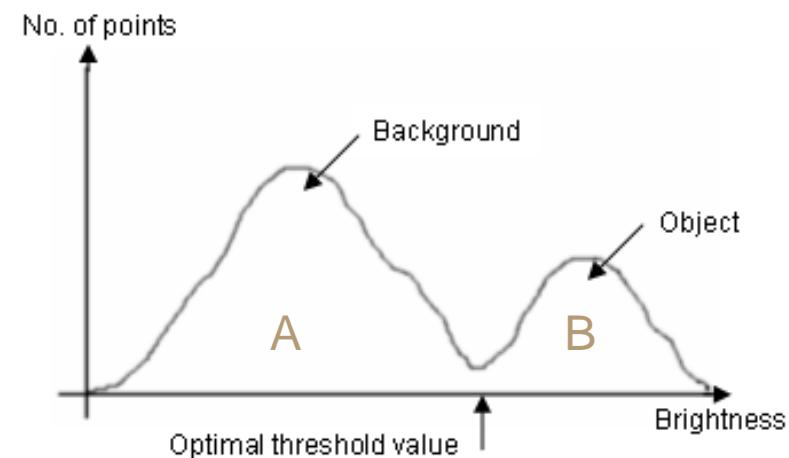
Adaptive Thresholds

- If the user chooses t for each of a set of images there is no guarantee the results will be consistent
- Automatic methods choose a threshold based on image properties: histograms are commonly used



Otsu Thresholding

- Assumes histograms are *bimodal*; two regions can be separated by one threshold
- Think of the histogram as made of two normal (Gaussian) distributions, described by their means and deviations
- If a threshold is wrong it will include histogram bins from peak A in peak B
 - Peak A's deviation will be too small
 - Peak B's deviation will be too big
 - Peak A's size (area) will be too small
 - Peak B's size will be too big



Otsu Thresholding

- Find the threshold which minimises a weighted sum of the variations of the two regions that threshold produces
 - Weights are the areas of the histogram assigned to each region

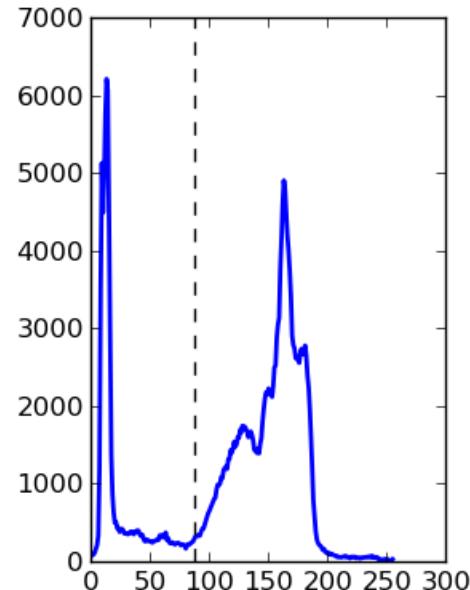
$$S_w^2(t) = q_1(t)S_1^2(t) + q_2(t)S_2^2(t)$$

$$q_1(t) = \bigodot_{i=1}^t P(i) \quad q_2(t) = \bigodot_{i=t+1}^I P(i)$$

- This is small when the two regions are **both physically small and have low deviations**

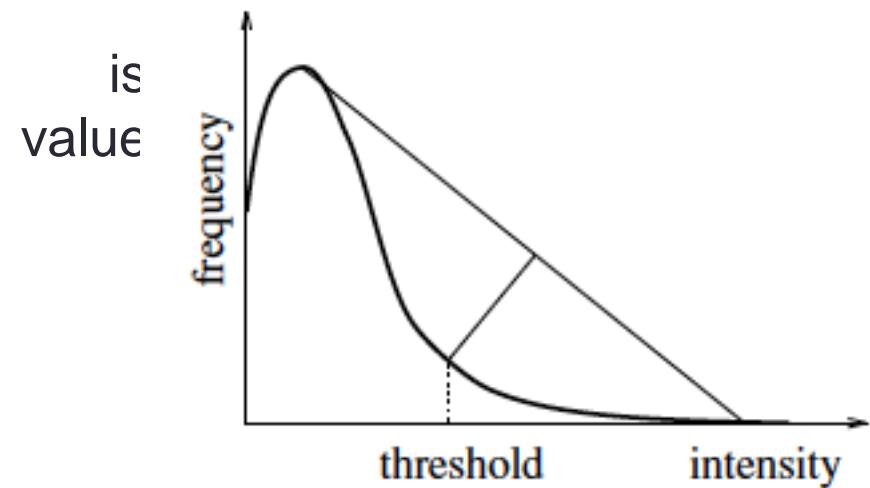
Otsu Thresholding

- The Algorithm
 - consider all possible threshold values (0 - 255)
 - compute weighted sum
 - pick t with smallest value
- A recursive version exists that is very efficient



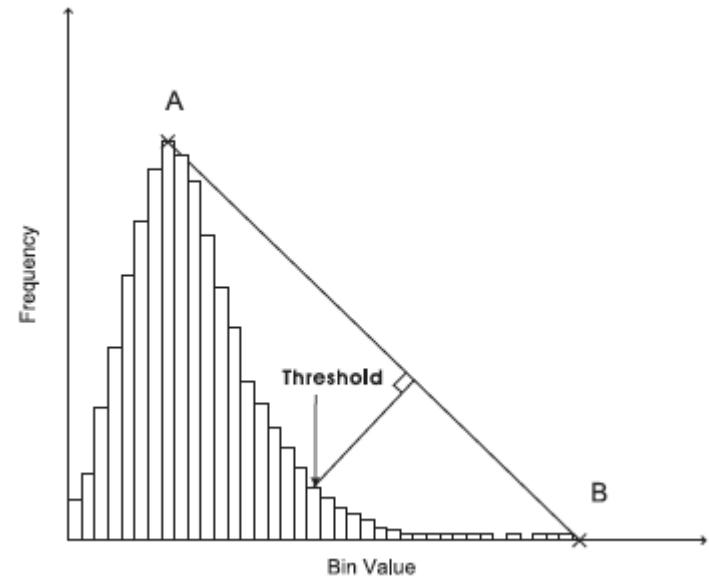
Unimodal Thresholding

- Many histograms are not bimodal, there is often only one peak e.g. text is mainly white, with a small amount of black
- Rosin's unimodal method
 - Finds the peak
 - Draws a line from there to the top of the furthest bin
 - Finds the top of the bin that



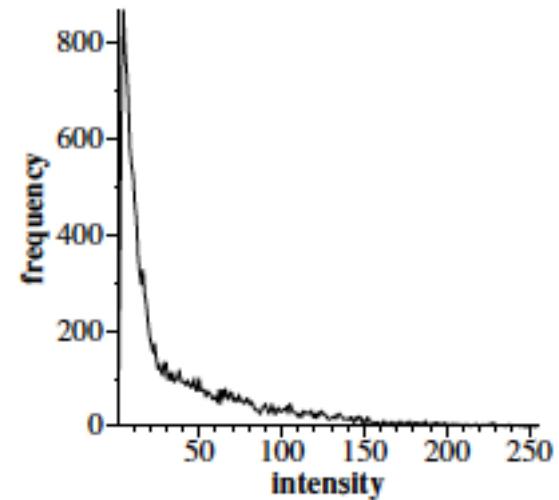
Unimodal Thresholding

- Many histograms are not bimodal, there is often only one peak e.g. text is mainly white, with a small amount of black
- Rosin's unimodal method
 - Finds the peak
 - Draws a line from there to the top of the furthest bin
 - Finds the top of the bin that is furthest from this line; that bin value is the threshold
 - The threshold is selected at the point of the histogram that **maximizes** the perpendicular distance from the histogram to the straight line.



Unimodal Threshold

- Can be applied to any suitable image, e.g. intensity gradients



Unimodal Thresholding



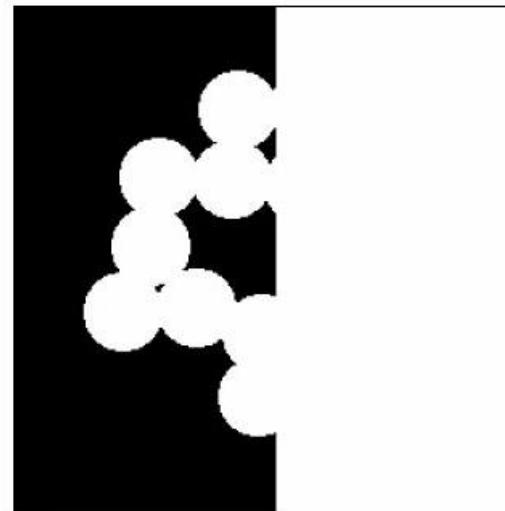
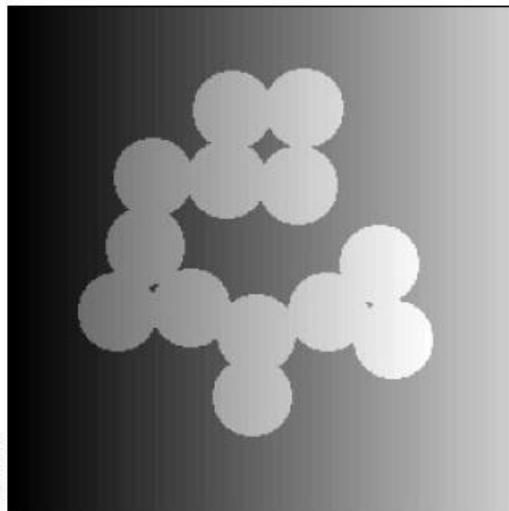
- Rosin



- Otsu

Local Adaptive Methods

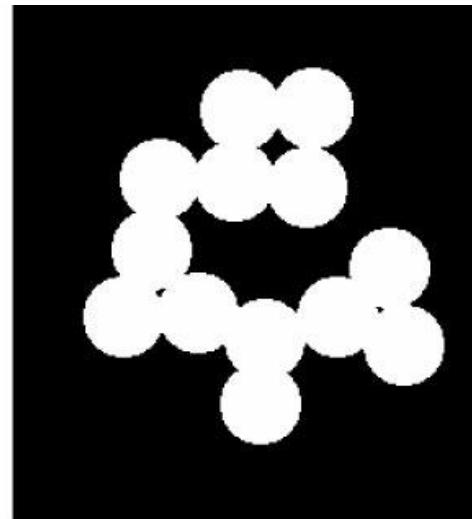
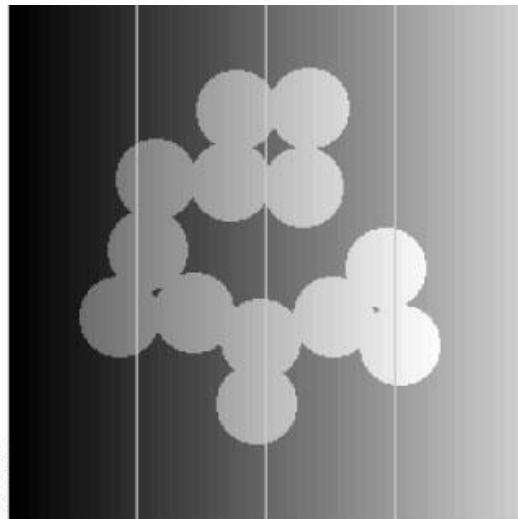
- Imaging conditions and object properties can vary within a single image as well as across sets of images
- Histograms can be too complex for any method's assumption to be true



■ Otsu

Local Adaptive Methods

- Assumptions about histograms may, however, be true for local areas of the image
- Divide image into subregions, apply a threshold selection method independently to each



- The histograms of each vertical strip of this image are bimodal
- Otsu can be applied to each strip

Conclusion

- Binary images contain only two values, but they are still images and can be processed
- Thresholding is the most common way to produce them
 - fixed vs adaptive
 - global vs local

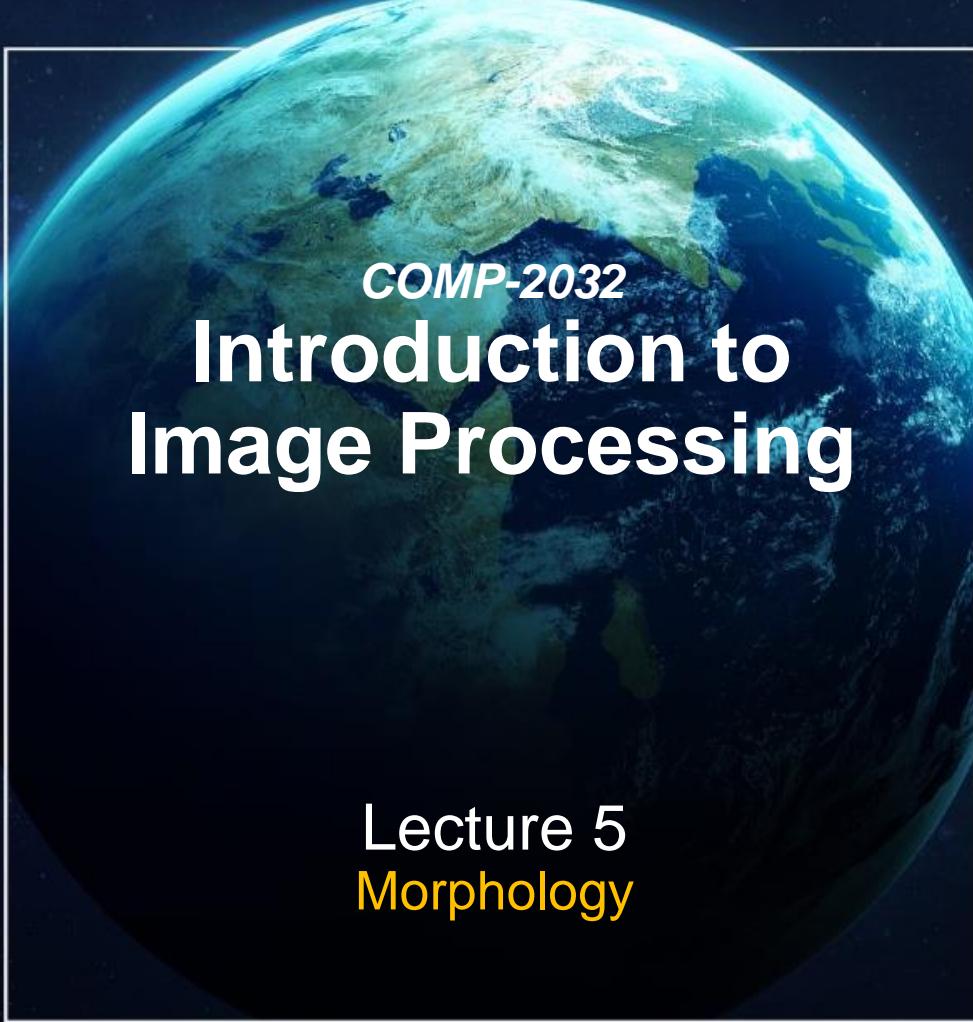
Next Week

Morphological Approaches



University of
Nottingham

UK | CHINA | MALAYSIA



COMP-2032
**Introduction to
Image Processing**

Lecture 5
Morphology



Univers
Nottin
UK | CHIN

COMP2005: Introduction to Image Processing

Week 23 – 10:00am Friday – 01 March 2024

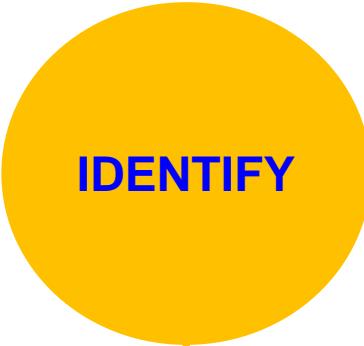


valid for 65 minutes from 9:55am
generated 2024-01-20 03:09

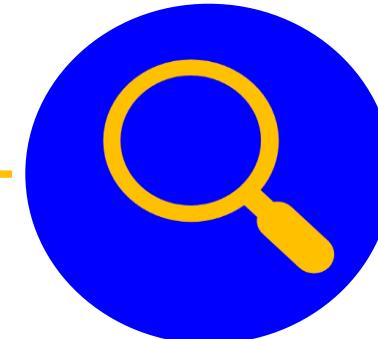




Learning Outcomes



1. Connected Components
2. What is Mathematical Morphology
3. Erosion and Dilation
4. Opening and Closing
5. ROIs and Masks



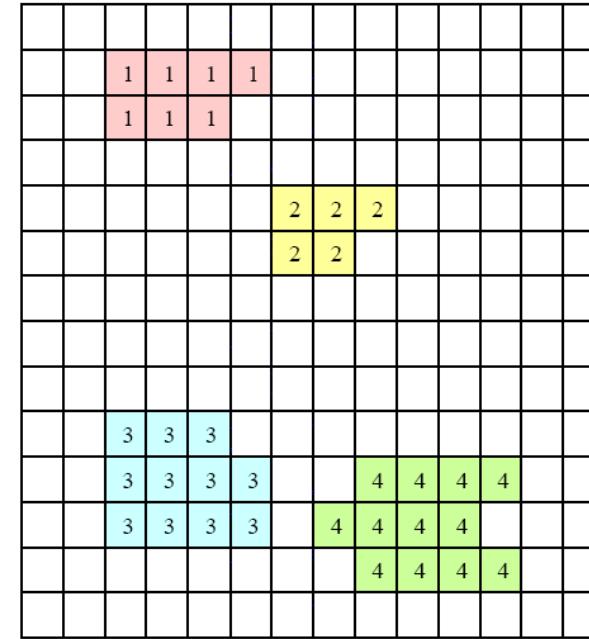
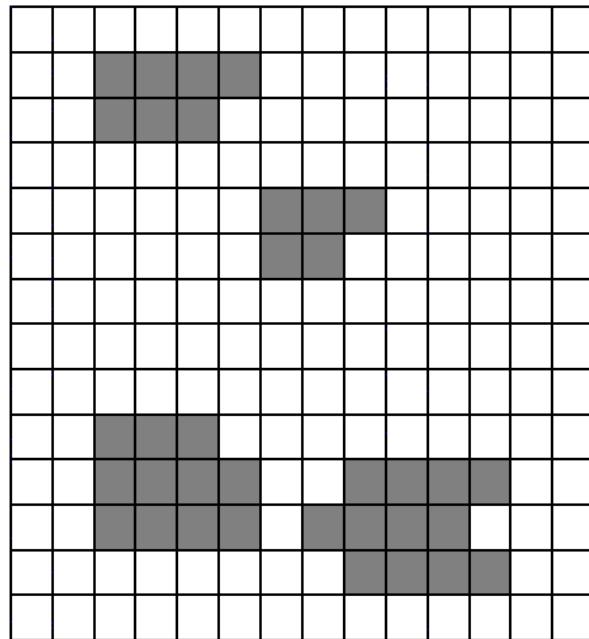


Connected Components



Connected Components

- If black pixels are objects & white are background, give each connected set of black pixels a different label



- Connected can mean 4- or 8- neighbours

REMEMBER



Connected Components

- Connected component algorithms are **slow** and often a **bottleneck**

- Keep a separate output array for labels
- Two passes over the image

A sequential algorithm using 4-neighbours

- Scan image top left to bottom right
- Look at top and left neighbours (already given labels)
- Can we assign either of their labels to the current pixel?

First pass



Connected Components

- If the current pixel is **foreground**, there are 3 cases:

1. left = top = background → assign current pixel a new label
2. one of left and top is background, the other foreground → assign current pixel the foreground pixel's label
3. Left = top = foreground → assign current pixel one of their labels and note that their labels are equal in an *equivalence table*

- Step 3 detects mergers between two previously separate components



Connected Components

- Consider each equivalent pair of labels, set all instances of the higher label to the lower

Second pass

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	0	0	0	0
0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1st Scan

			a	a	a		b	b		c	c							
		d	d	d		a	a	a	a									
e	e		d	d	d	a	a	a	a	a	a							

2nd Scan : Replace b & d by a

			a	a	a		a	a		c	c							
		a	a	a		a	a	a	a									
e	e		a	a	a	a	a	a	a	a	a							

$d=a$, $b=a$
 $d=a$
:equivalence table

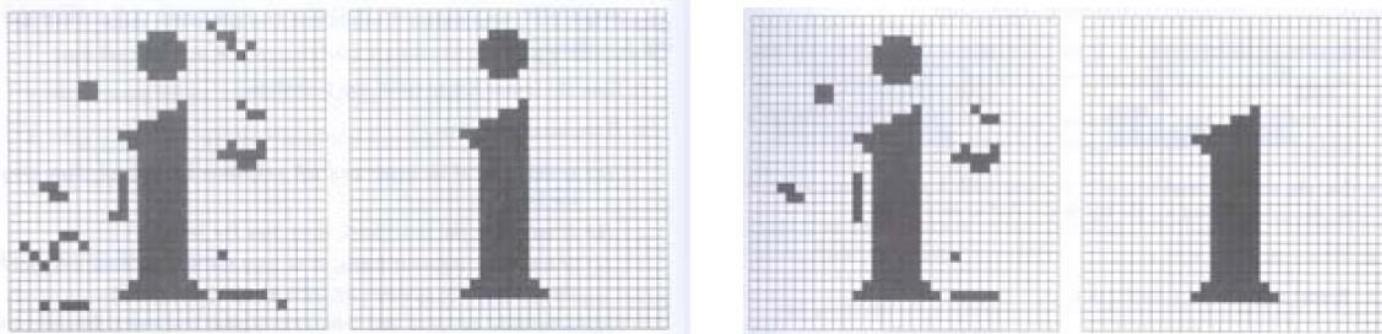
Using 8 neighbours makes the algorithm (a little) more complex and gives (slightly) different results



Connected Components

- Can compute features of and apply tests to components to process the underlying image

- e.g., apply a (size) threshold T to the number of pixels in the component



$T=10$

$T=25$

- Shape is captured by measures like area/boundary length

- Many application-specific features and tests exist



What is Mathematical Morphology



Mathematical Morphology

- A branch of image processing which treats images as sets of pixels and uses set theoretic operations to process them
- Developed for binary images, extended to grey level images
- Elements of sets are (x,y) coordinates of black (or white) pixels
- Perform operations by combining two sets:

- A patch of the binary image to be processed
- A Structuring Element, similar to the mask in a convolution process

Underlying mathematics is beyond the scope of COMP2032.

Instead, we will focus on how they work in practice and what they do

DISCLAIMER



Structuring Elements

- Binary masks, c.f. filter masks but identifying rather than weighting pixels
- Larger structuring elements produce more extreme effects
- Very similar effects can be achieved by repeated operations using a smaller but similarly shaped structuring element
- With larger structuring elements, it is quite common to use an approximately disk-shaped structuring element
- Need not be square, origin need not be in the centre

1	1	1
1	①	1
1	1	1

	1	
1	①	1
	1	

	1	1	1	1	
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1



Morphological Operations

- | Expands a foreground (or background) object A using structuring element B

Dilation

Erosion

- | Shrinks a foreground (or background) object A using structuring element B

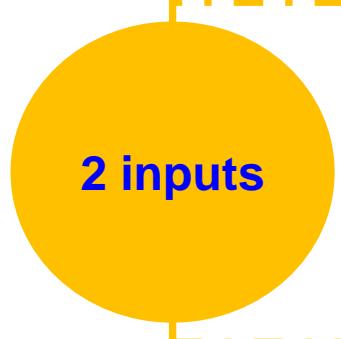
- The boundaries between foreground and background are often smoothed in the process
- The amount and the way objects grow and shrink depend upon the choice of the structuring element
- Dilating or eroding without specifying the structural element makes no sense than trying to filter an image without specifying the filter



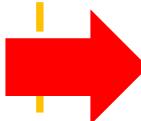
Erosion & Dilation



Dilation



- Binary image to dilate
- Set of points to be considered; a *structuring element*
- Origin is the central element



1	1	1
1	1	1
1	1	1

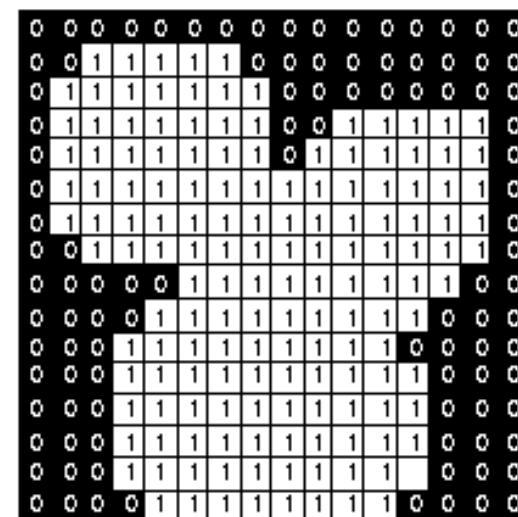
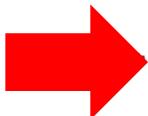
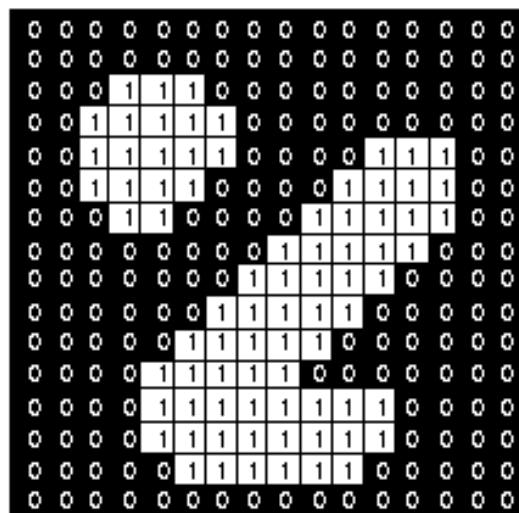
- The structuring element is superimposed on each of the **background pixels** such that origin of the structuring element coincides with the input pixel position
- If any of the '1' pixels in the structuring element overlap (intersect) the **foreground** then the **background** pixel is also set to **foreground**

Algorithm
(dilate foreground)



Dilation

- Gradually enlarges the boundaries of regions of foreground pixels (i.e., white pixels, typically)
 - Areas of foreground pixels grow in size while holes within those regions become smaller



Result of dilation with a square 3 x 3 structuring element



A More interesting Example

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



0	1	0
1	1	1
0	1	0

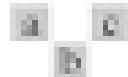
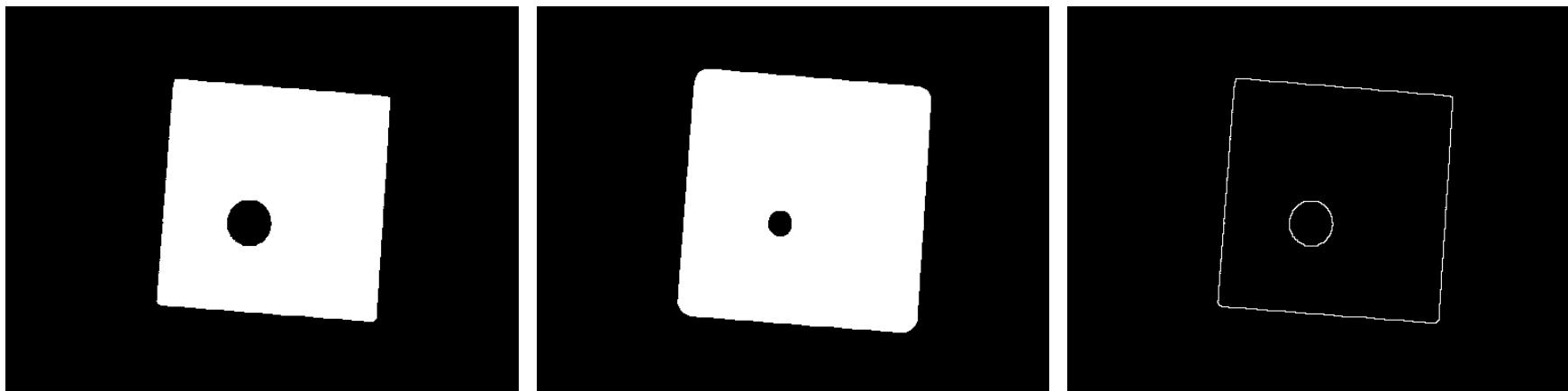


FIGURE 9.5
(a) Sample test of poor resolution with broken characters (magnified view).
(b) Structuring element.
(c) Dilation of (a) by (b). Broken segments were joined.



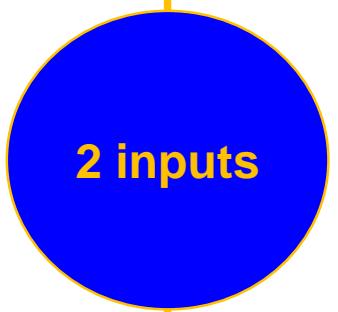
Edge Detection by Dilation

- Dilation input image (*note rounding of corners*)
- Subtract from original
- Edges remain (*pixels on the outside of the boundary*)

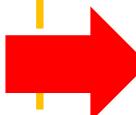




Erosion



- Binary image to erode
- Set of points to be considered; a *structuring element*
- Origin is the central element



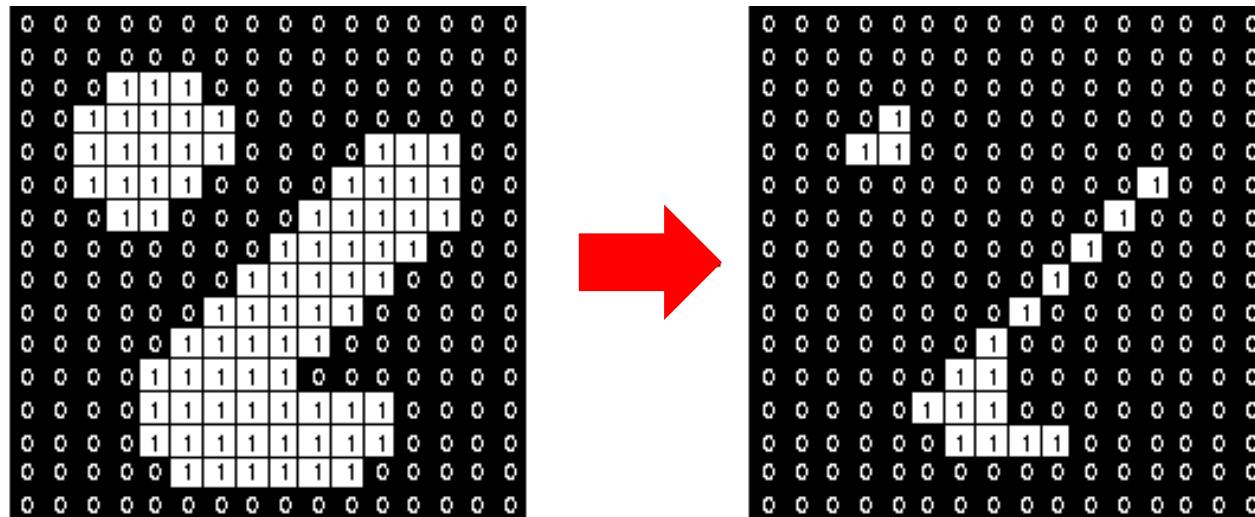
1	1	1
1	1	1
1	1	1

- The structuring element is superimposed on each of the **foreground** pixels such that origin of the structuring element coincides with the input pixel position
- If any of the '1' pixels in the structuring element overlap (intersect) the **background** then the **foreground** pixel is also set to **background**

Algorithm
(erode foreground)



Erosion



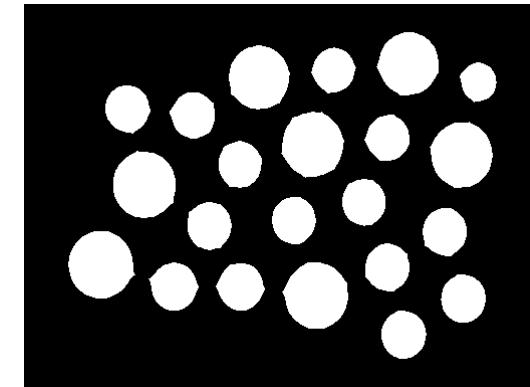
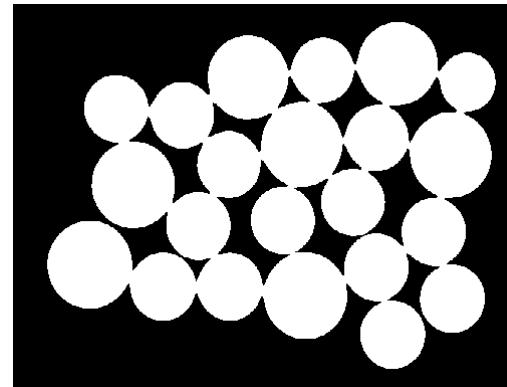
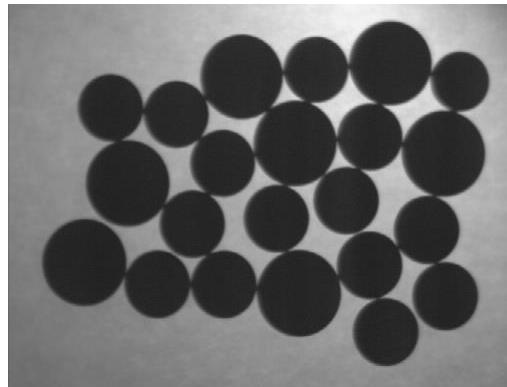
Result of erosion with a square 3 x 3 structuring element

- Erosion is the *dual* of dilation, i.e., eroding foreground pixels is equivalent to dilating the background pixels with the same structuring element



Erosion

- Counting objects (cells, coins) can be difficult if they touch
- Erosion can separate them



- Erosion can be used for edge detection too – giving pixels on the inside of the boundary

REMEMBER

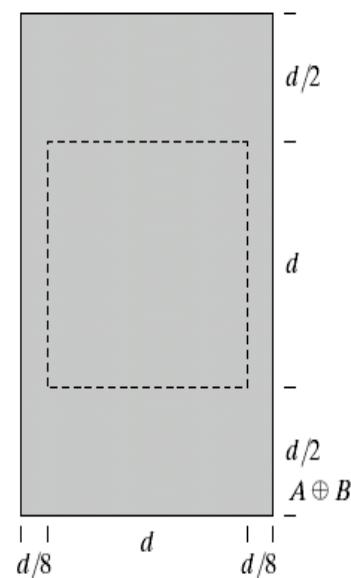
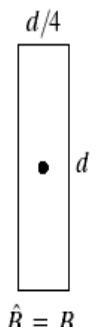
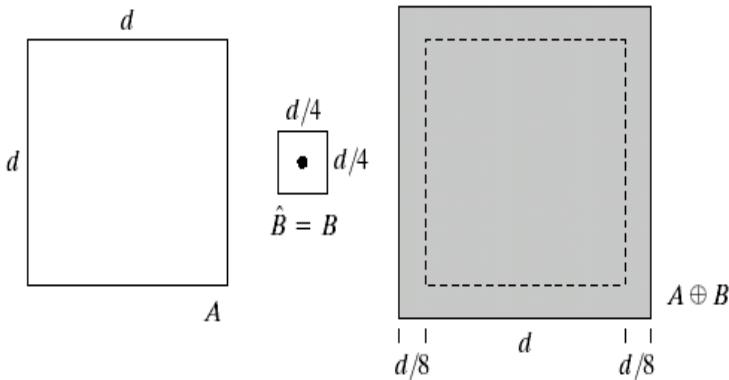


Asymmetric Structuring Elements

a	b	c
d		e

FIGURE 9.4

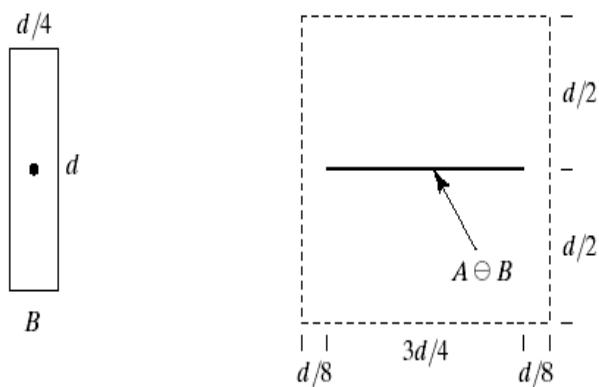
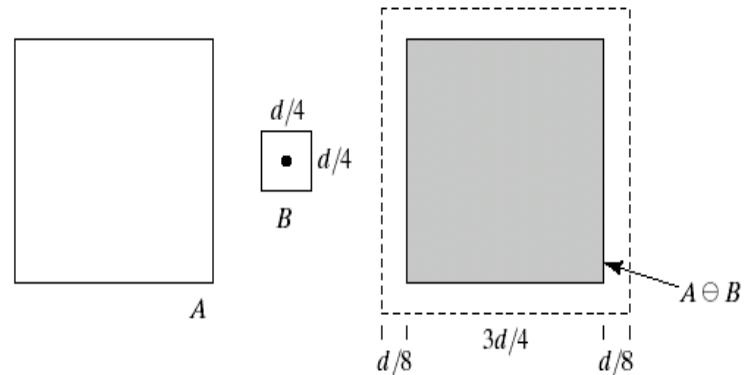
- (a) Set A .
- (b) Square structuring element (dot is the center).
- (c) Dilation of A by B , shown shaded.
- (d) Elongated structuring element.
- (e) Dilation of A using this element.



Dilation



Asymmetric Structuring Elements



a	b	c
d	e	

FIGURE 9.6 (a) Set A. (b) Square structuring element. (c) Erosion of A by B, shown shaded. (d) Elongated structuring element. (e) Erosion of A using this element.

Erosion



Break





Opening & Closing



Combining Dilation and Erosion

Its rare to need only erosion and dilation, and they are much more useful when combined



a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.



Opening and Closing

- First erode A with B, then dilate A with B
- Smoothes contours, eliminates protrusions

Opening

Closing

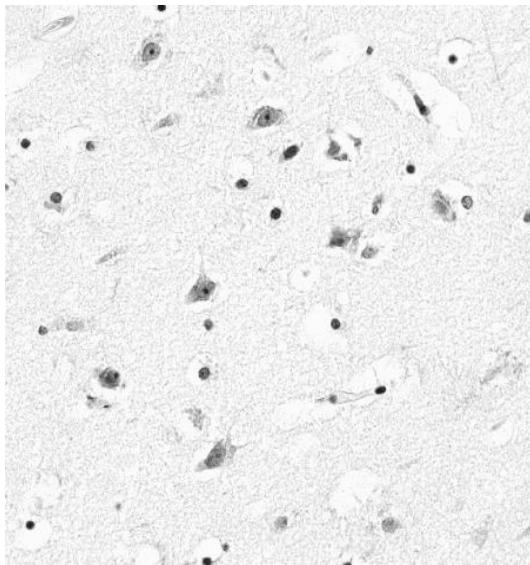
- First dilate A with B, then erode A with B
- Smoothes sections of contours, fuses narrow breaks and long thin gulfs, eliminates small holes and fill gaps in contours

- These operations are dual to each other
- These operations can be applied multiple times, but have an effect only once (*the first time*)

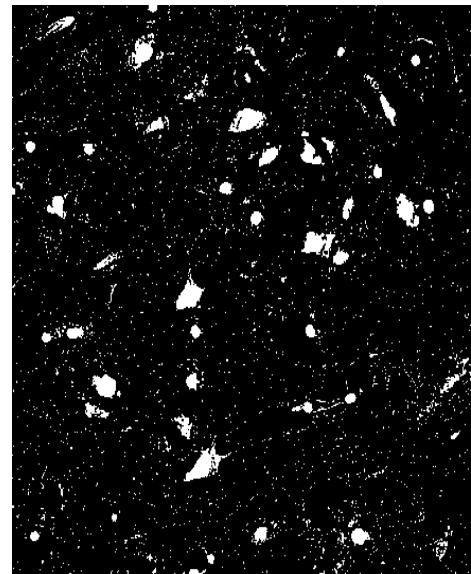


Opening

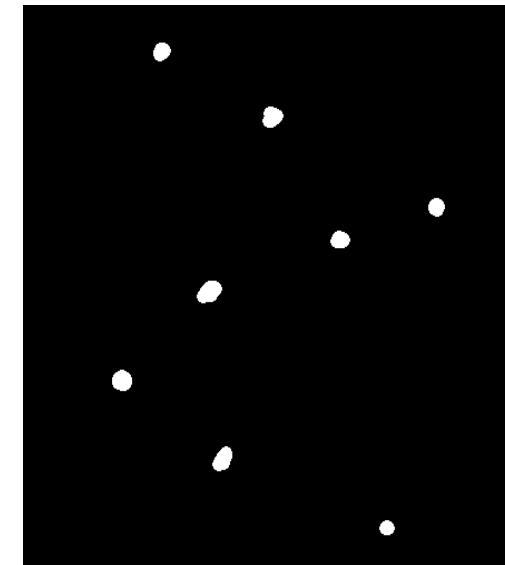
(a) Contains two kinds of cells: small, black ones and larger, grey ones. Thresholding with a value of 210 yields (b). Opening (b) using an 11 pixel circular structuring elements yields (c)



(a)



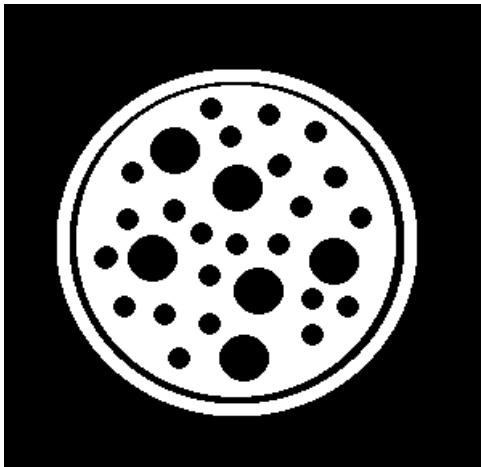
(b)



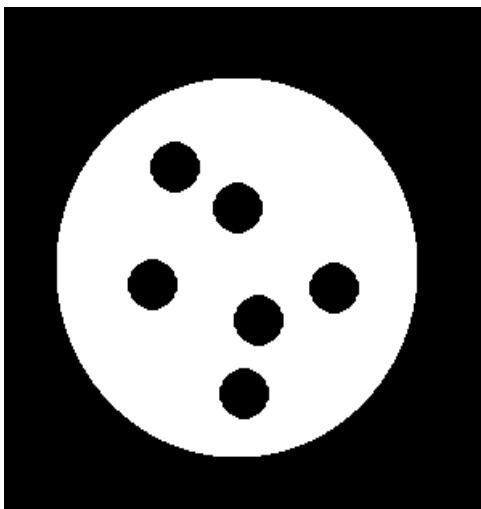
(c)



Closing



To retain only the large holes, we can simply perform a closing with a disk-shaped structuring element with a diameter larger than the smaller holes, but smaller than the large holes



The result of a closing with a 22 pixel diameter disk



ROIs & Masks

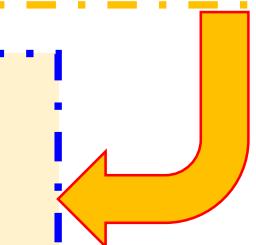


The Whole Image

- All our examples so far have applied processes to the whole image – or section of image – input
- The tools we have become more powerful if we can apply them to selected pixels, and more powerful still if the selection can be automatic

OpenCV supports *Region of Interest Processing* through NumPy array slicing

- ROI Object Creation and Modification
- Mask creation
- ROI filtering





Python OpenCV

Supports filtering and inpainting of:

- Manually defined rectangular ROIs
- Interactively drawn bounding box ROIs
- ROIs defined by any appropriately sized binary image
- ROIs defined by colour

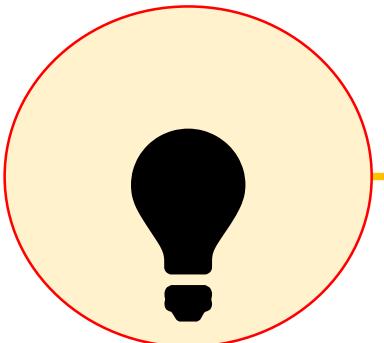


The concept of ROIs is very powerful; code can be written to allow the output of one process to determine where another is applied

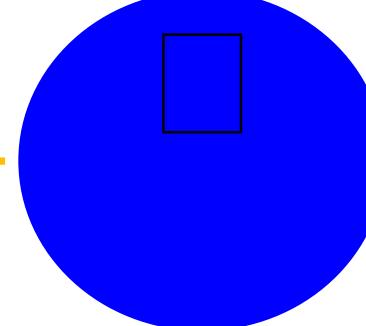
REMEMBER



Summary



1. Connected Components
2. What is Mathematical Morphology
3. Erosion and Dilation
4. Opening and Closing
5. ROIs and Masks





University of
Nottingham

UK | CHINA | MALAYSIA

Questions



University of
Nottingham

UK | CHINA | MALAYSIA

A photograph of Earth from space, showing the curvature of the planet and city lights at night. A white rectangular frame is positioned over the upper portion of the image, containing the text "NEXT:" and "Derivates and Edges".

NEXT:

Derivates and Edges

COMP2005

Derivative Filters

In 1D

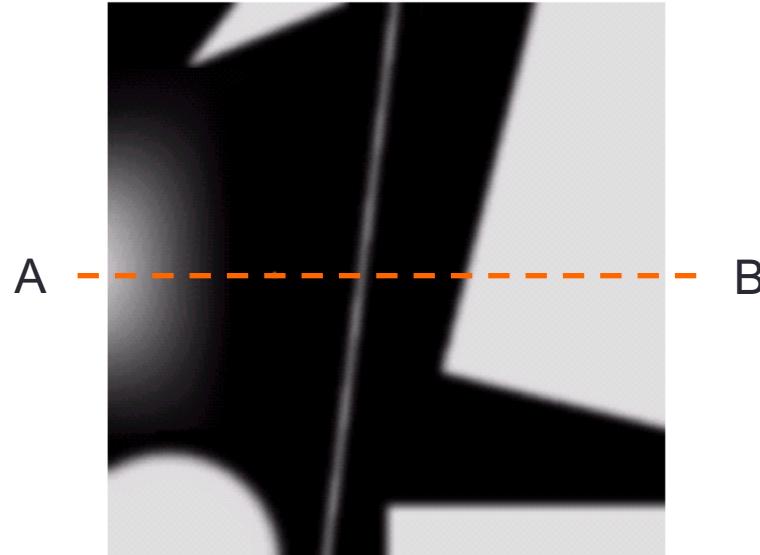
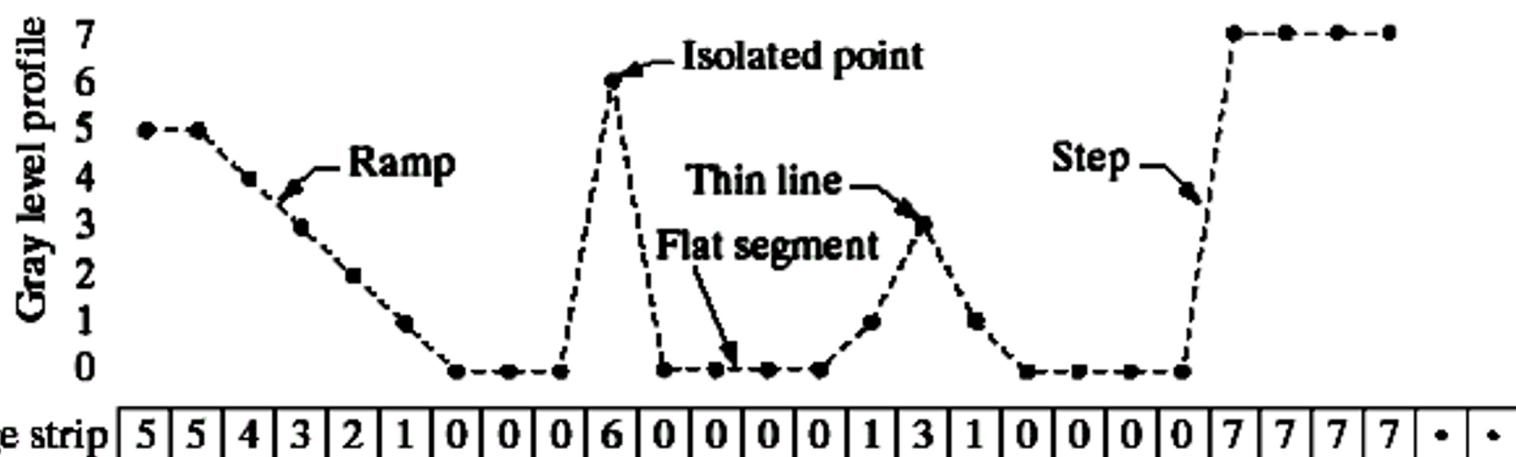


Image features are often characterised by changes in intensity

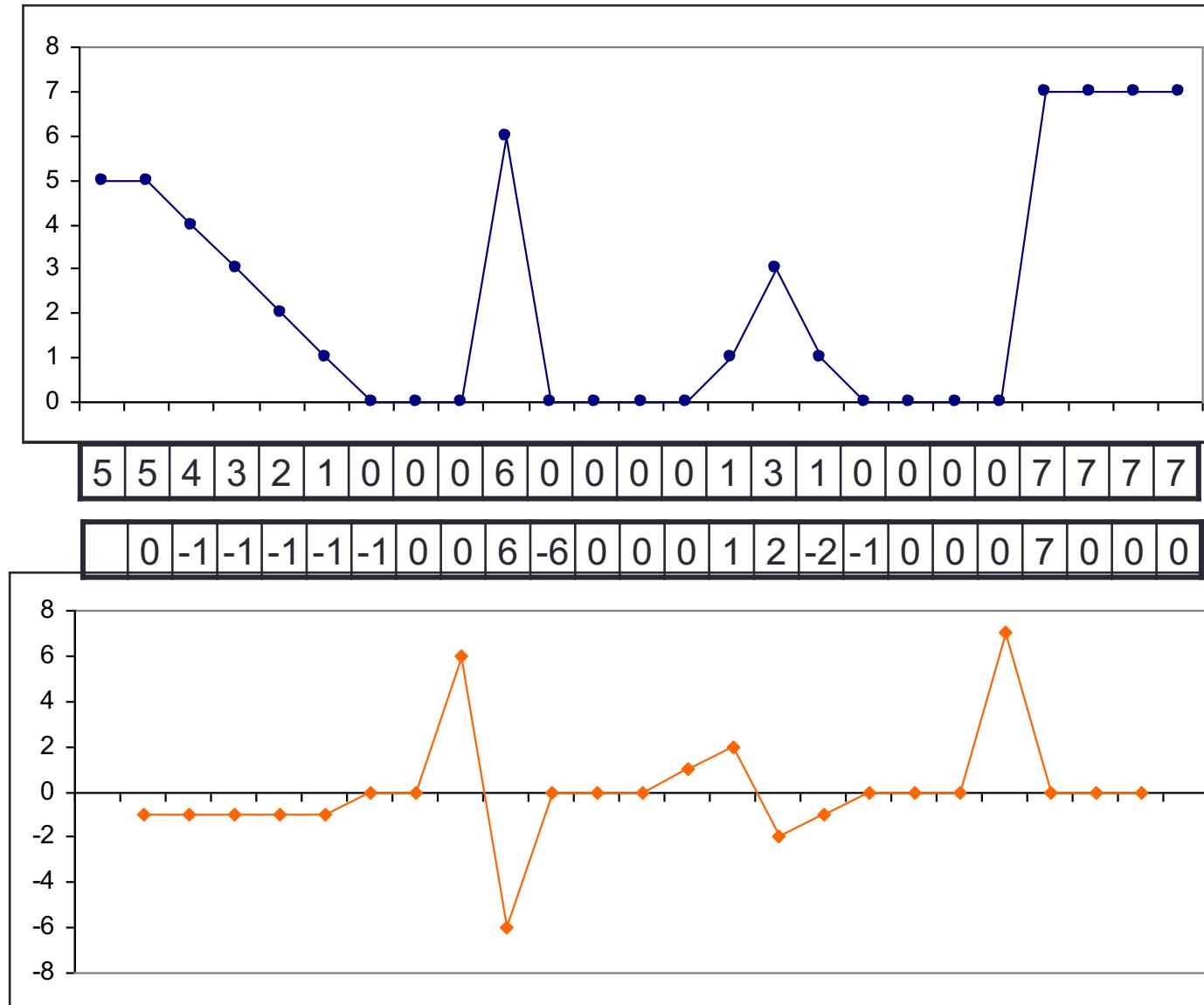


1st Derivative

- The 1st derivative of a function can be approximated by:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- The difference between neighbouring values and measures the rate of change of the function
- Can also be approximated by $(f(x+1) - f(x-1))/2$, etc.



Raw data

1st derivative

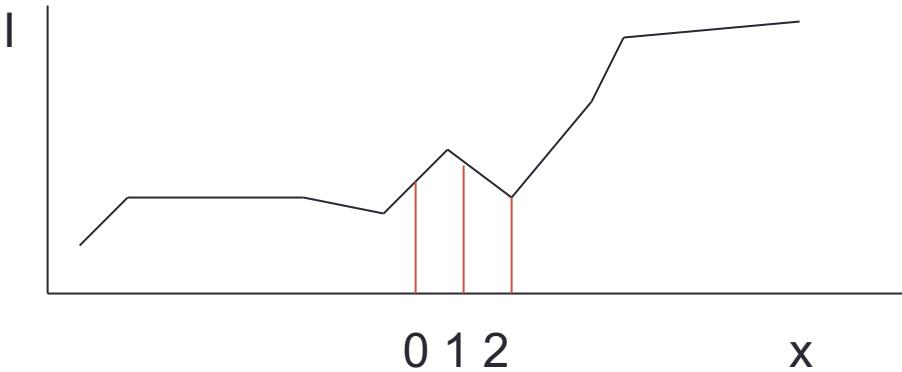
2nd Derivative

- The formula for the 2nd derivative of a function is:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

- Simply takes into account the values both before and after the current value
- Derived by estimating the 1st derivative at $x + 0.5$ and $x - 0.5$ and computing the derivative of the resulting data

2nd Derivative

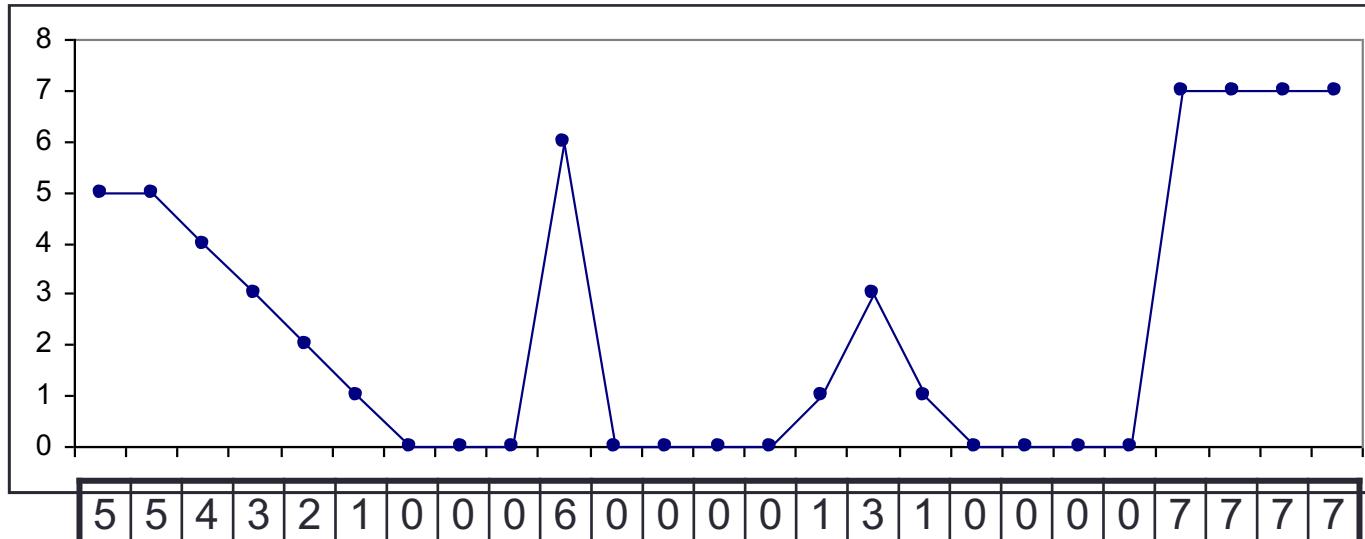


$$I''(1) = (I'(1.5) - I'(0.5))/1$$

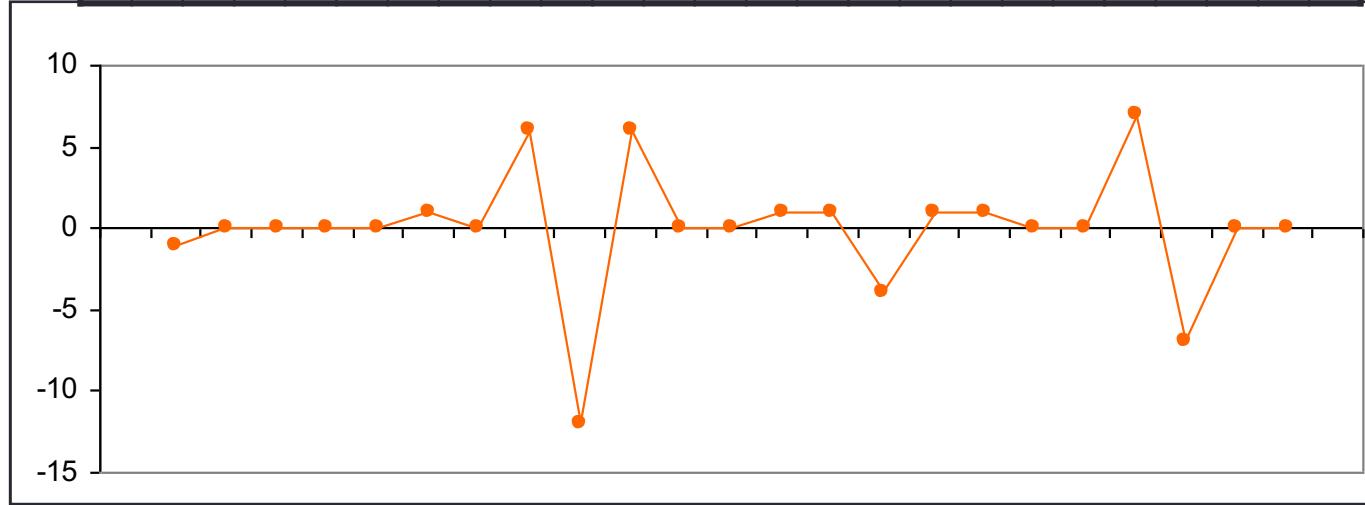
$$I'(0.5) = (I(1) - I(0))/1 \quad \text{and} \quad I'(1.5) = (I(2) - I(1))/1$$

$$\therefore I''(1) = 1.I(0) - 2.I(1) + 1.I(2)$$

1	-2	1
---	----	---



Raw data



2nd
derivative

Derivatives in 2D

- 2nd derivatives generalize to 2D quite easily, implementing a 1st derivative in 2D is a little more complex
- For a function $f(x, y)$ the gradient of f at coordinates (x, y) is given as the column vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- Computation of the 1st derivative can't be done by convolution alone

1st Derivative Filtering

- The magnitude of the 1st derivative vector is

$$\begin{aligned}\nabla f &= \text{mag}(\nabla f) \\ &= [G_x^2 + G_y^2]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}\end{aligned}$$

which can be simplified to

$$\nabla f \approx |G_x| + |G_y|$$

1st Derivative Filters

- Many 1st derivative filters have been proposed

- Roberts' Cross Operators

1	0
0	-1

0	1
-1	0

- Sobel Operators

-1	0	1
-2	0	2
-1	0	1

G_x

-1	-2	-1
0	0	0
1	2	1

G_y

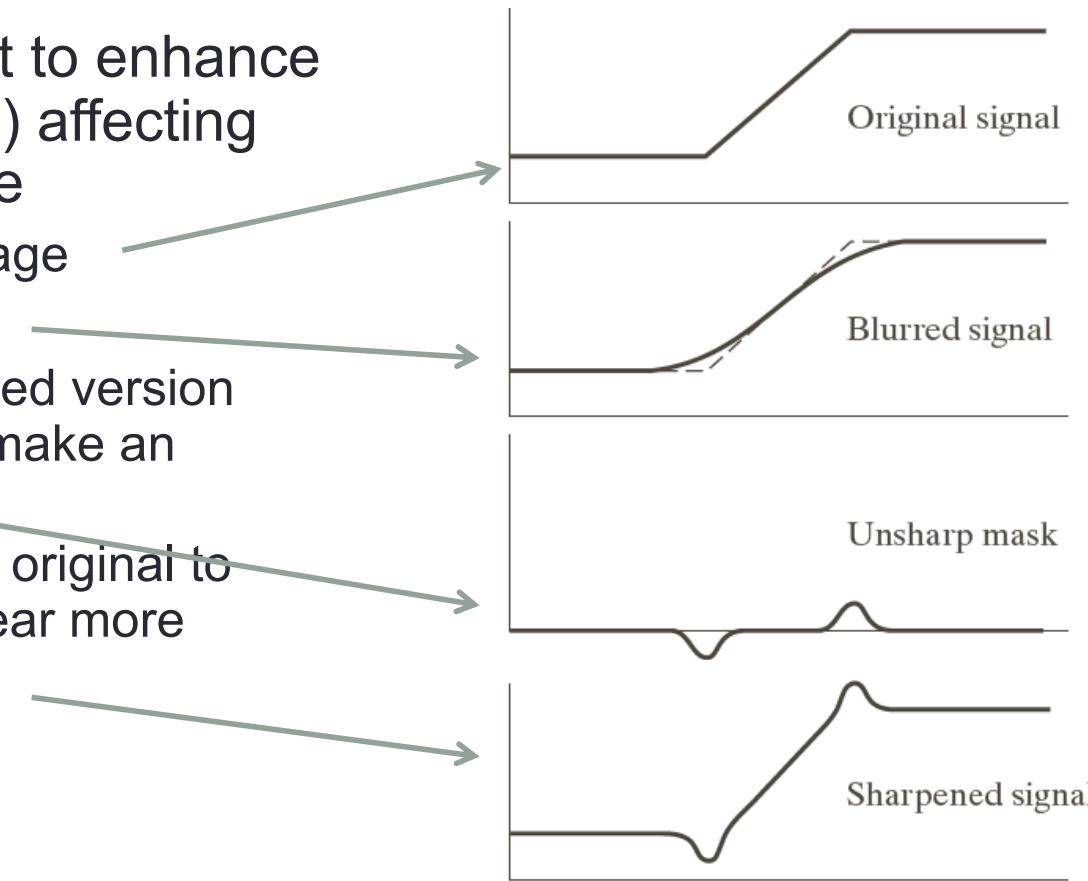
These operators are most commonly associated with edge detection

COMP2005

Image Sharpening

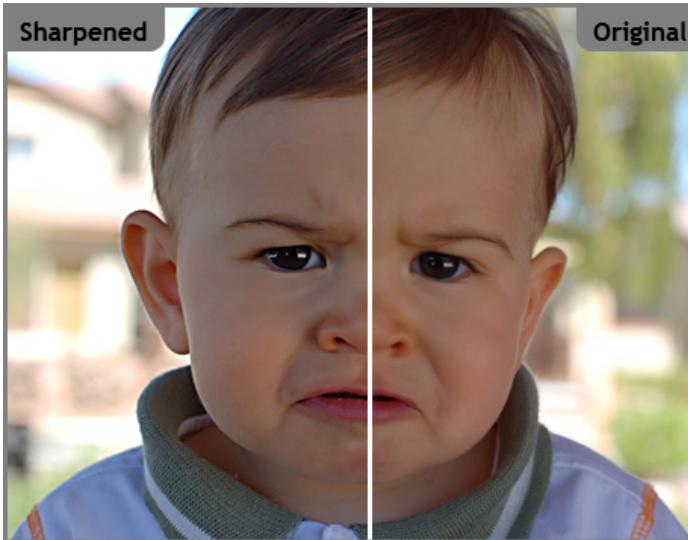
Edge Enhancement: Unsharp Masking

- Edges are important
- Sometimes we want to enhance them without (much) affecting the rest of the image
 - Take the original image
 - Gaussian smooth it
 - Subtract the smoothed version from the original to make an *unsharp mask*
 - Add the mask to the original to make the edge appear more obvious



Unsharp Masking

- Makes edges noticeably sharper
 - Even if they are noise
 - Sometimes too much



Derivative Filters

- Unsharp filtering enhances edges by comparing the original with a smoothed image
 - relies on the smoothing effect of a Gaussian function introducing a difference between original and processed images
 - parameterised by σ
 - simple, but effect is hard to predict, so hard to parameterise
- A more direct way to highlight edges and other features associated with high image gradients is to estimate derivatives.....

Image Sharpening with Derivatives

- The 2nd derivative is more useful for image enhancement than the 1st derivative
 - Stronger response to fine detail
 - Simpler implementation
- The most common sharpening filter is the *Laplacian*
 - Isotropic
 - One of the simplest sharpening filters
 - Straightforward digital implementation via convolution

The Laplacian

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

The Laplacian

$$\begin{aligned}\nabla^2 f = & [f(x+1, y) + f(x-1, y) \\& + f(x, y+1) + f(x, y-1)] \\& - 4f(x, y)\end{aligned}$$

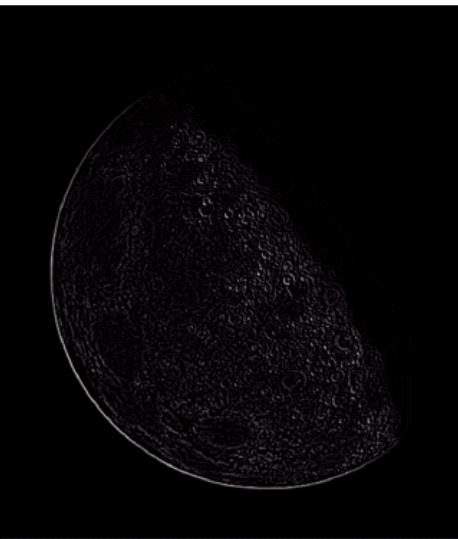
0	1	0
1	-4	1
0	1	0

The Laplacian

- Highlights edges and other discontinuities



Original
Image



Laplacian
Filtered Image



Laplacian
Filtered Image
Scaled for Display

Could do better?

- The result of Laplacian filtering is not an enhanced image
 - subtract the Laplacian result from the original image to generate the final sharpened enhanced image,
c.f. unsharp masking

$$g(x, y) = f(x, y) - \nabla^2 f$$



Original



Laplacian Filtered

=



Sharpened

Laplacian Enhancement

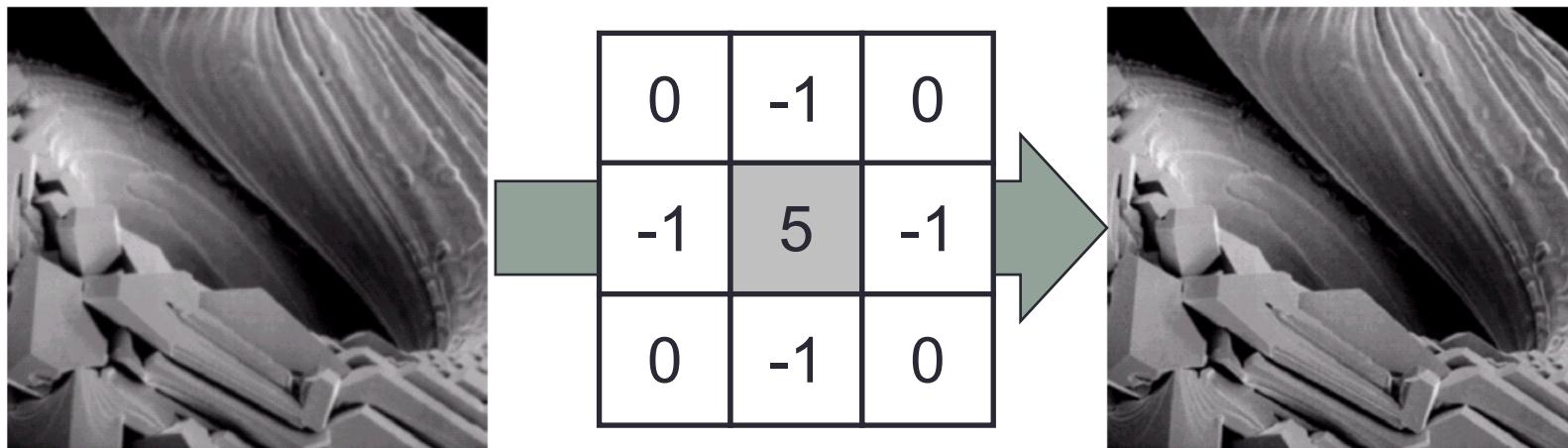


A Single Enhancement Operator

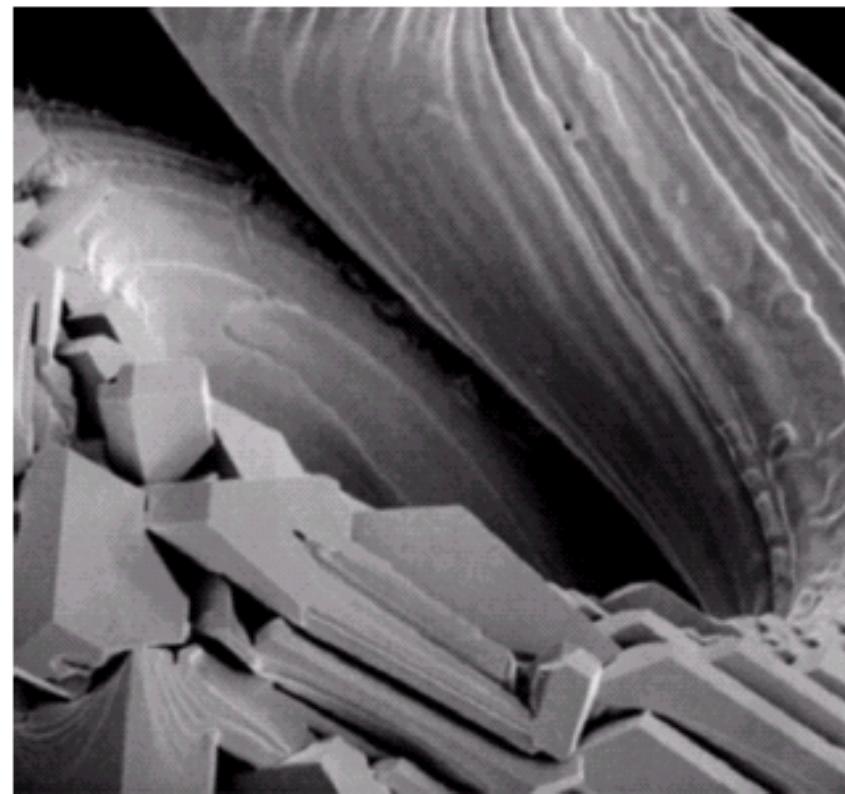
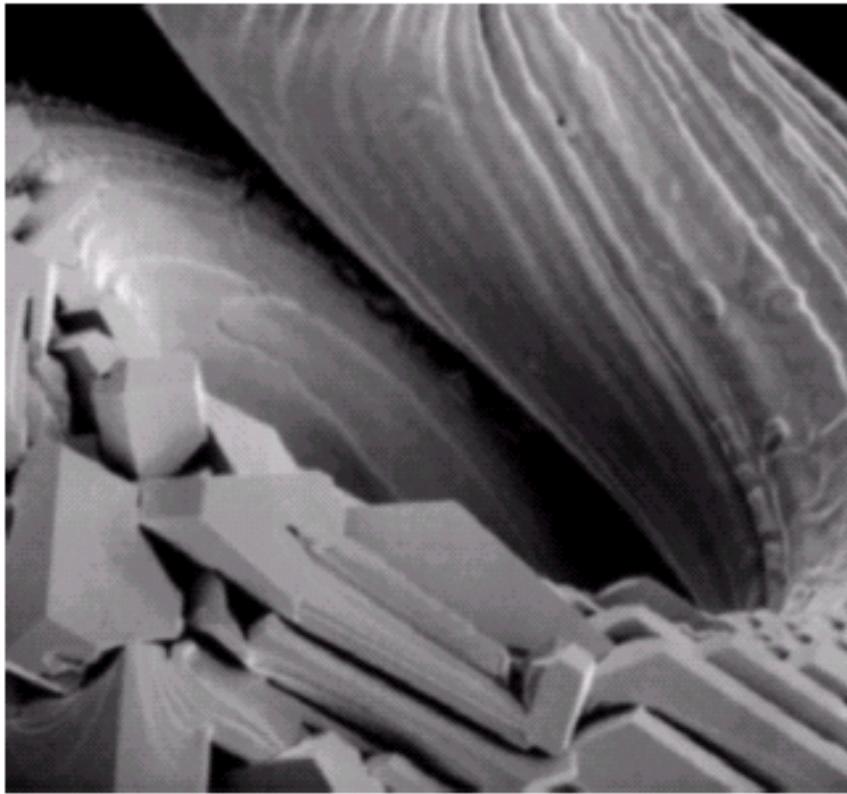
$$\begin{aligned}g(x, y) &= f(x, y) - \nabla^2 f \\&= f(x, y) - [f(x+1, y) + f(x-1, y) \\&\quad + f(x, y+1) + f(x, y-1) \\&\quad - 4f(x, y)] \\&= 5f(x, y) - f(x+1, y) - f(x-1, y) \\&\quad - f(x, y+1) - f(x, y-1)\end{aligned}$$

A Single Operator

- Convolution with this operator performs image sharpening in a single step



A Single Operator



Variations on the Theme

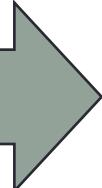
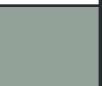
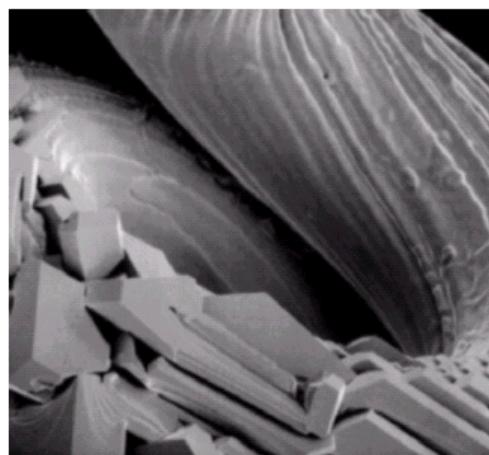
- Some formulations take 2nd derivatives measured across the diagonals into account

0	1	0
1	-4	1
0	1	0

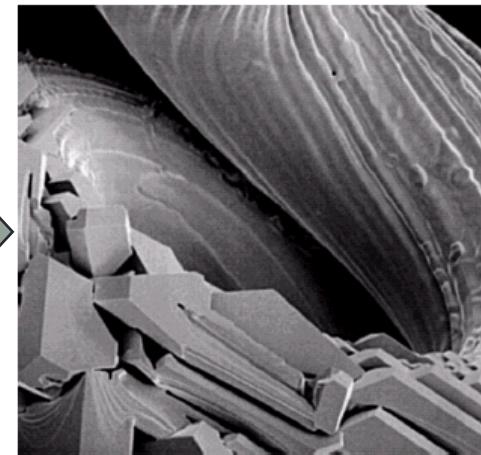
Basic

1	1	1
1	-8	1
1	1	1

Extended



-1	-1	-1
-1	9	-1
-1	-1	-1

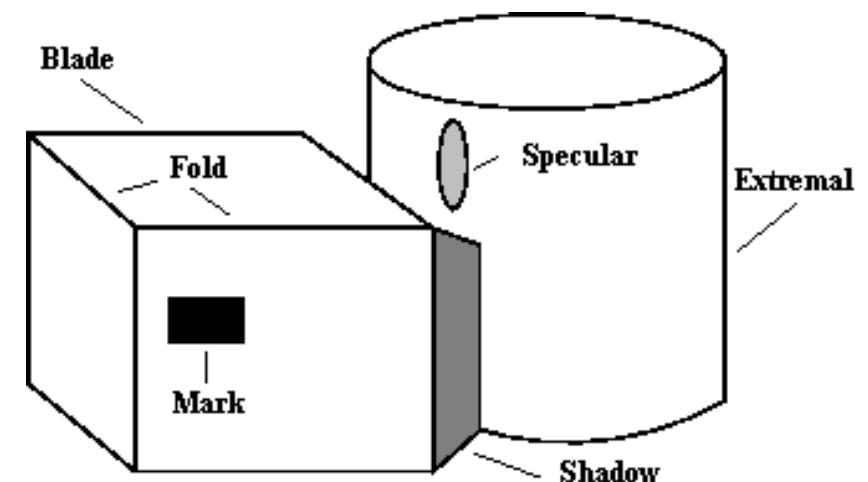


COMP2005

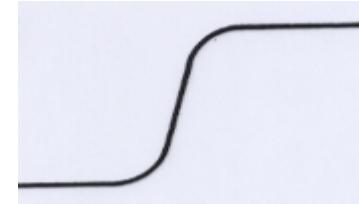
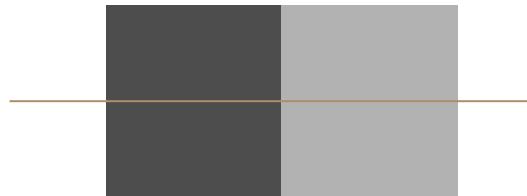
What is Edge Detection?

Edge Detection

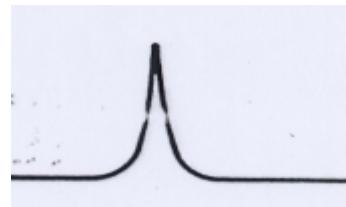
- First step in many image analysis and computer vision processes and applications
- The goal is to mark points at which image intensity changes sharply – edges
- Sharp changes in **image** properties reflect events/changes in the **world**
- This is only an assumption, but it is usually true



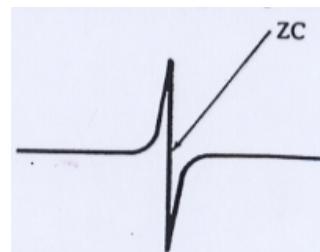
The Theory



- An idealised image of an edge



- 1st derivative: a peak



- 2nd derivative: a zero-crossing

To detect edges find peaks in the 1st derivative of intensity or zero-crossings in the 2nd derivative

The Result

```
>> im = imread('cameraman.tif');  
>> edges = edge(im, 'Canny');  
>> imshowpair(im, edges, 'montage');
```



COMP2005

Edge Detection using 1st Derivative Filters

1st Derivative Filters

- Roberts' Cross Operators

1	0
0	-1

0	1
-1	0

- Sobel Operators

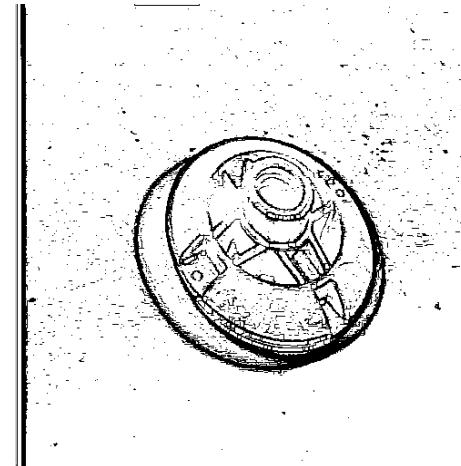
-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

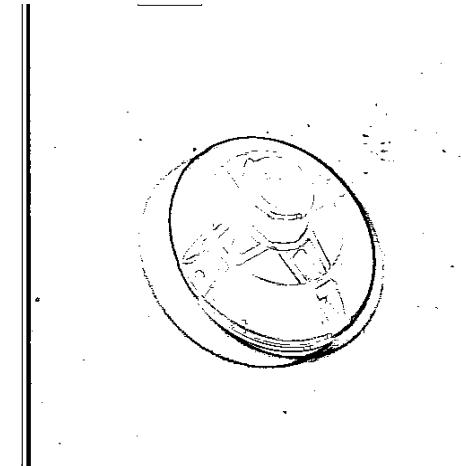
- Applied separately and results combined to estimate magnitude

Detection & Thresholding

- Significant peaks in magnitude of 1st derivative are high
- Apply a threshold, all peaks higher than the threshold value are significant, all others are ignored



■ Too low



■ Too high

Edge Magnitude & Direction

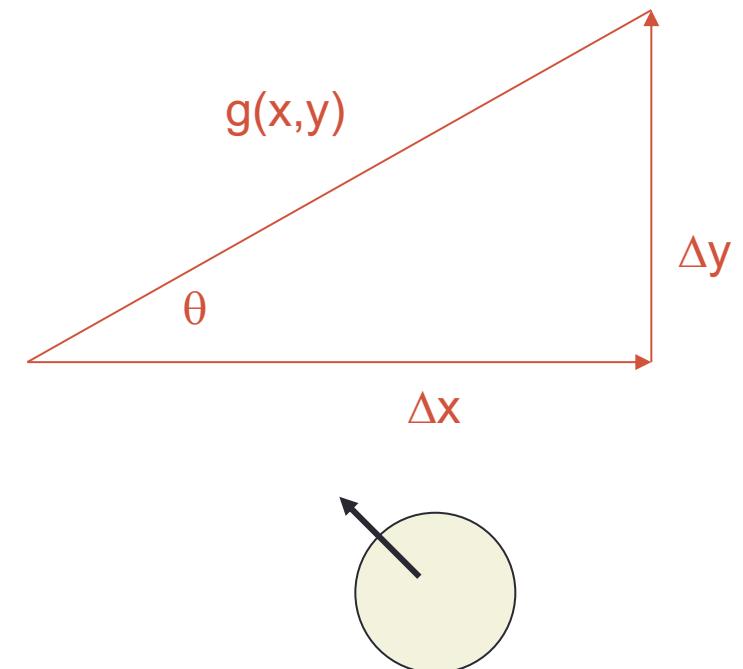
- For an image function, $I(x,y)$, the gradient magnitude, $g(x,y)$ is given by

$$g(x,y) \cong (\Delta x^2 + \Delta y^2)^{1/2}$$

- The gradient direction, $\theta(x,y)$, gives the direction of steepest image gradient

$$\theta(x,y) \cong \text{atan}(\Delta y / \Delta x)$$

- This gives the direction of a line perpendicular to the edge



Roberts' Cross Operator

- Very quick to compute - 4 pixels, only subtractions and additions, but is very sensitive to noise and only gives a strong response to very sharp edges



Original



Cross Operator



Thresholded

Sobel vs Roberts

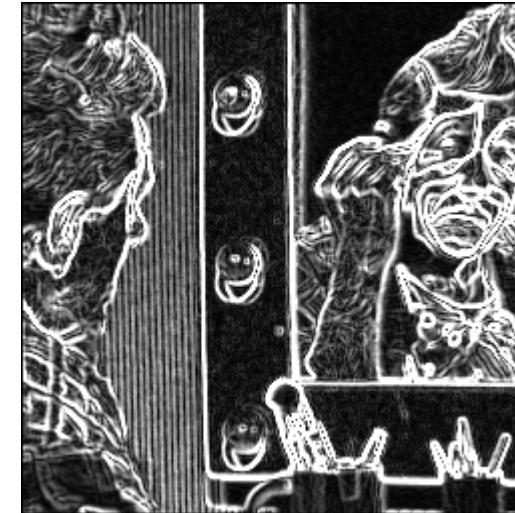
- Both use a user-supplied threshold. Sobel is still in use, Roberts is less common.
- Larger Sobel operators **are more stable in noise**



Original



Roberts

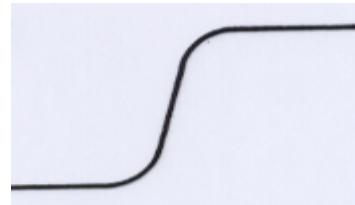
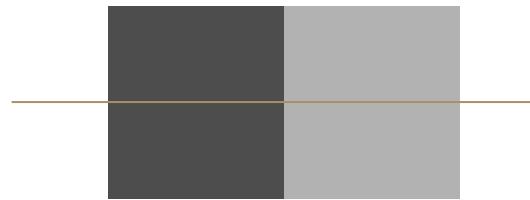


Sobel

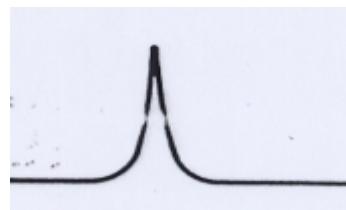
COMP2005

Edge Detection using 2nd Derivatives

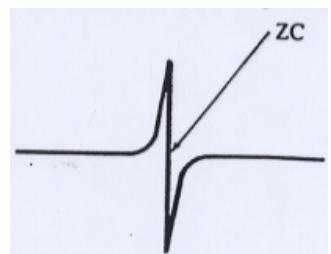
The Theory



- An idealised image of an edge



- 1st derivative: a peak



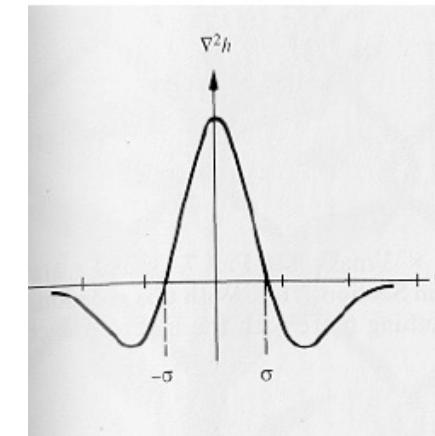
- 2nd derivative: a zero-crossing

To detect edges find peaks in the 1st derivative of intensity or zero-crossings in the 2nd derivative

2nd Derivatives: Marr-Hildreth

- Biologically inspired
 - Gaussian smooth, compute Laplacian
- OR
- Convolve with the **Laplacian of a Gaussian**

$$\nabla^2[f(x, y) * G(x, y)] = \nabla^2G(x, y) * f(x, y)$$

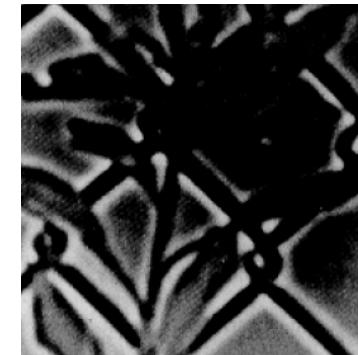


Laplacian of Gaussian (LoG)

5 × 5 Laplacian of Gaussian mask

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

17 × 17 Laplacian of Gaussian mask																
0	0	0	0	0	0	-1	-1	-1	-1	0	0	0	0	0	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-2	-1	-1	0	0	0	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-2	-1	0	0	0	0
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-2	-1	-1	0	0
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1	0
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1	-1
-1	-1	-3	-3	-3	4	12	21	24	21	12	4	-3	-3	-3	-1	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1	-1
0	-1	-2	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1	0	0
0	-1	-1	-2	-3	-3	-2	-3	-2	-3	-3	-2	-1	-1	0	0	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-2	-1	-1	0	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-2	-1	-1	-1	0	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0



filtering



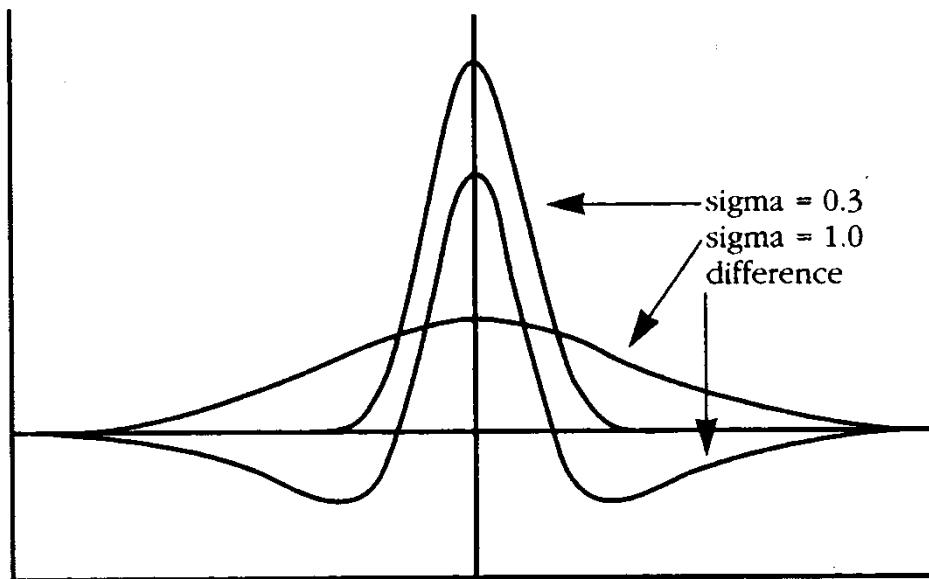
zero-crossings

Difference of Gaussians

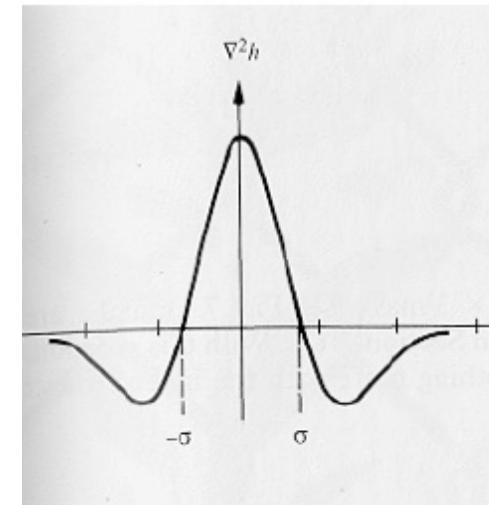
- The Laplacian of Gaussian can be approximated by the difference between two Gaussian functions:

$$\nabla^2 G \approx G(x, y; \sigma_1) - G(x, y; \sigma_2)$$

approximation



actual LoG

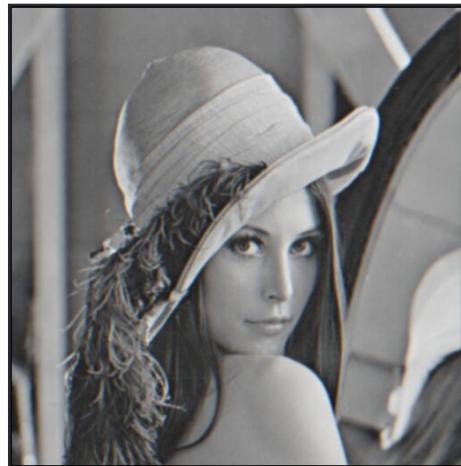


DoG Filtering

$$\nabla^2 G \approx G(x, y; \sigma_1) - G(x, y; \sigma_2)$$



$\sigma = 1$



$\sigma = 2$

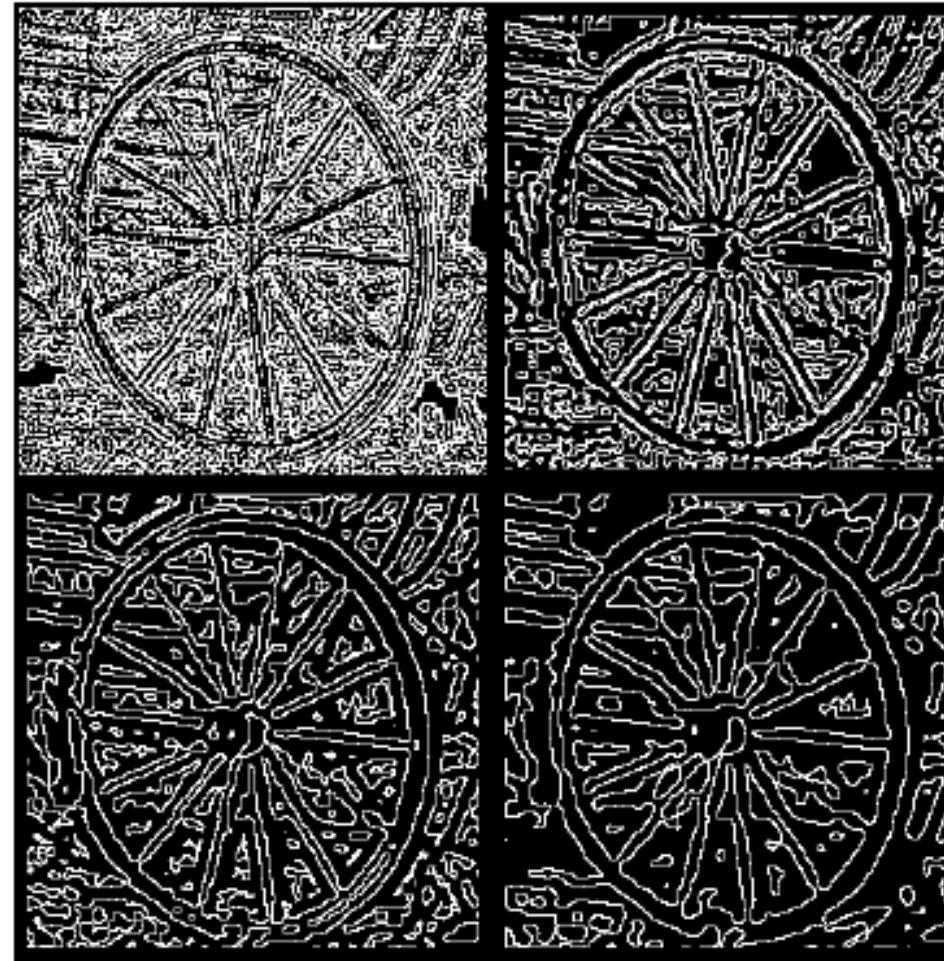


(b)-(a)

Ratio (σ_1/σ_2) for best approximation is about 1.6.
(Some people like $\sqrt{2}$.)

Marr-Hildreth

- Choice of σ gives flexibility



$\sigma = 1$	$\sigma = 2$
$\sigma = 3$	$\sigma = 4$

1st vs 2nd Derivative Methods

- Peaks in 1st derivative
 - strong response at edges, but also respond to noise
 - Peak detection and threshold selection need care
- Zero crossings in 2nd derivative
 - well-defined, easy to detect
 - must form smooth, connected contours
 - tend to round off corners
- 1st derivative methods are much more common in practical applications,
 - in part because of John Canny

COMP2005

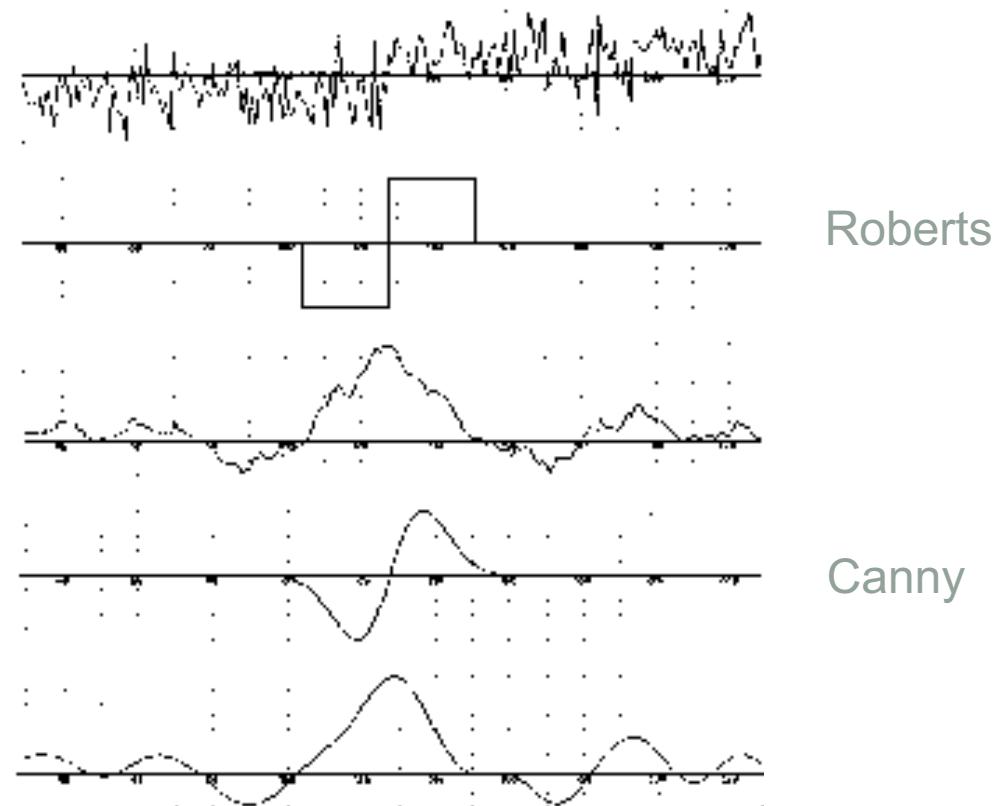
The Canny Operator

What Canny Did

- John Canny tried to find the optimal edge detector, assuming a perfect step edge in Gaussian noise
- Optimal means
 - **Good Detection** - it should mark all the edges and only all the edges
 - **Good Localisation** - the points marked should be as close to the real edge as possible
 - **Minimal Response** - each edge should be reported only once
- Canny used the *Calculus of Variations*: finds the function which best satisfies some functional

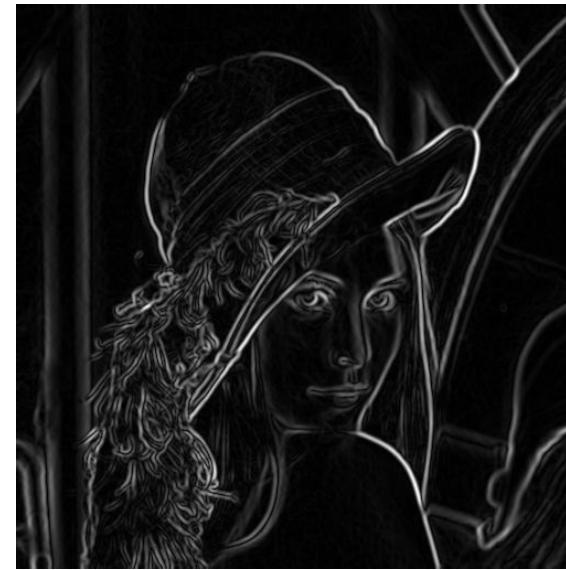
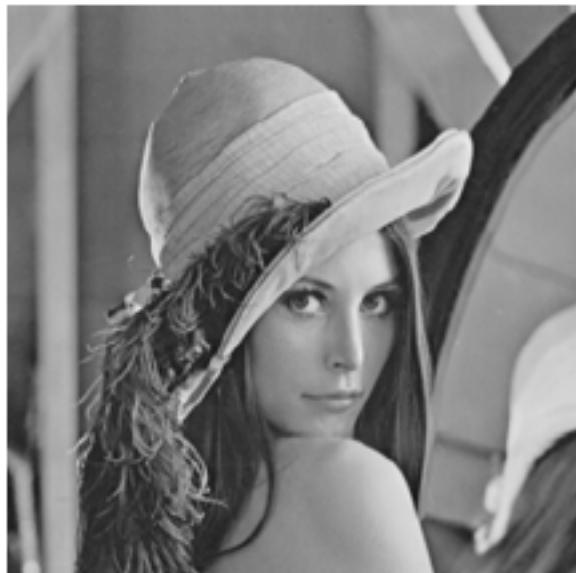
The Canny Operator

- The optimal detector was a sum of 4 exponential terms, but is very closely approximated by the 1st derivative of a Gaussian
 - i.e. 1st derivative of a Gaussian smoothed image
- Gives a cleaner response to a noisy edge than square operators
- Most implementations are 2D Gaussian smoothing + Roberts style derivative



Non-Maximal Suppression

- The Canny operator's response is cleaner than Sobel or Roberts, but it needs an explicit step to enforce Minimal Response



Canny Operator

Non-Maximal Suppression

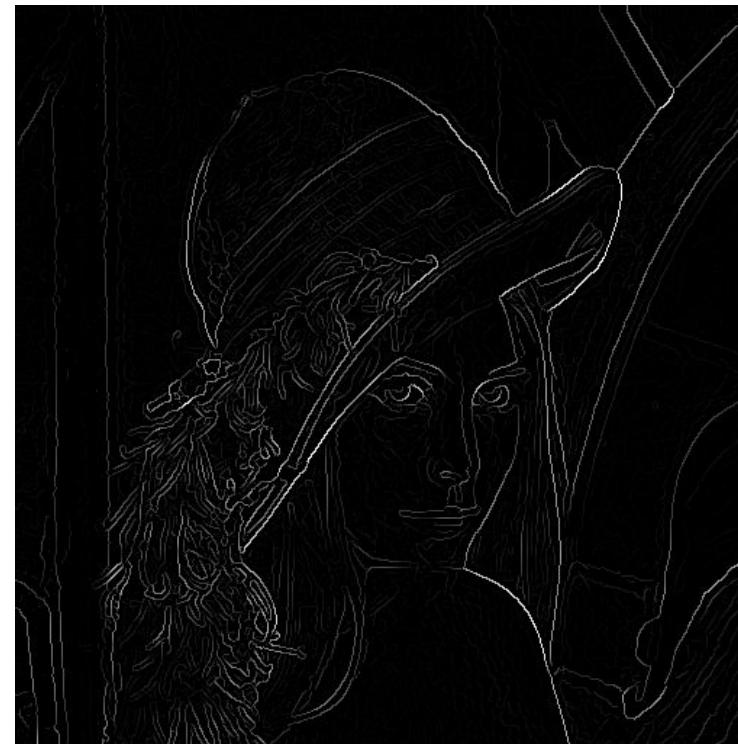
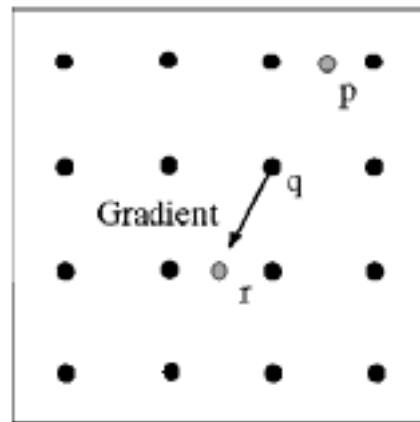
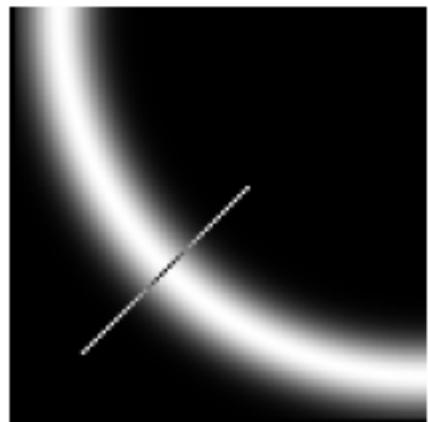
- Thresholding raw operator response would leave thick lines



How to turn
these thick
regions of the
gradient into
curves?

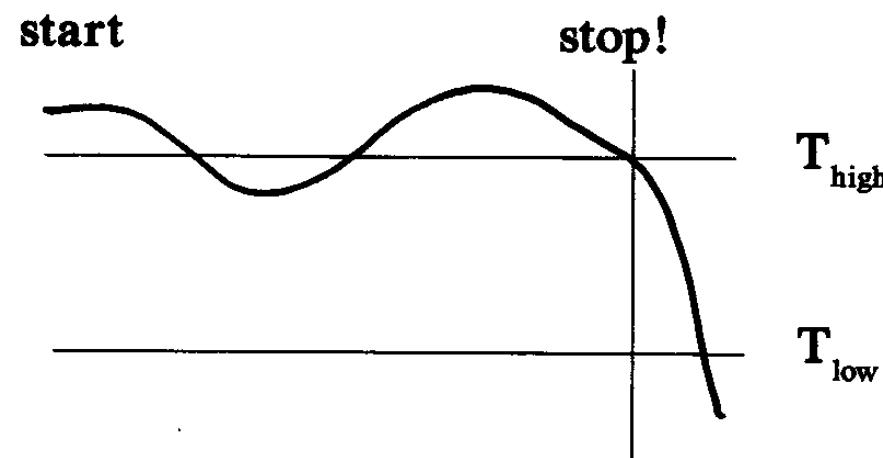
Non-Maximal Suppression

- Check if pixel is a local maximum along the gradient direction
- Select a single maximum across the width of the edge



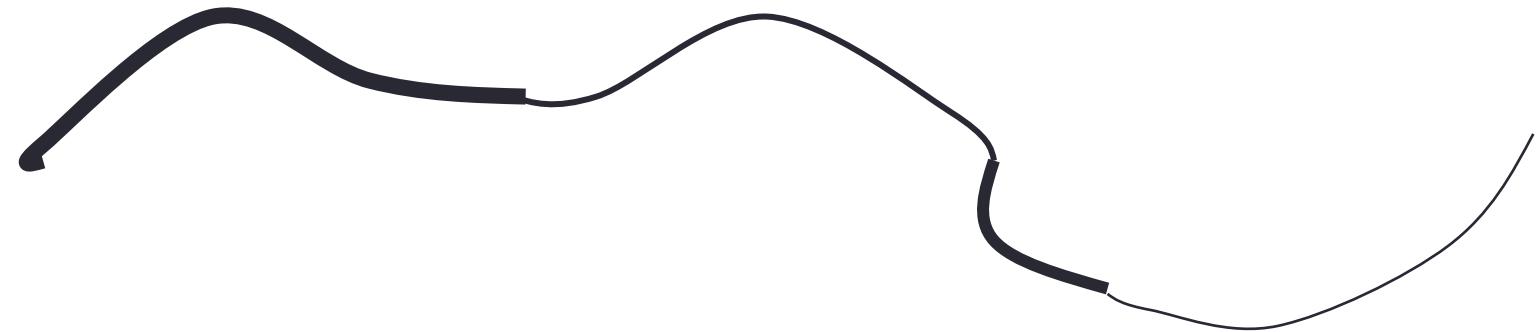
Thresholding with Hysteresis

- Simple thresholding tests each pixel independently: edges aren't really independent, they make up lines
- The industry standard edge thresholding method
 - Allows a band of variation, but assumes continuous edges
 - User still selects parameters, but it's easier, less precise



Thresholding with Hysteresis

- The effect is to keep weak edges if they connect strong edges, as long as
 - the strong edges are really strong
 - the weak edges aren't really weak



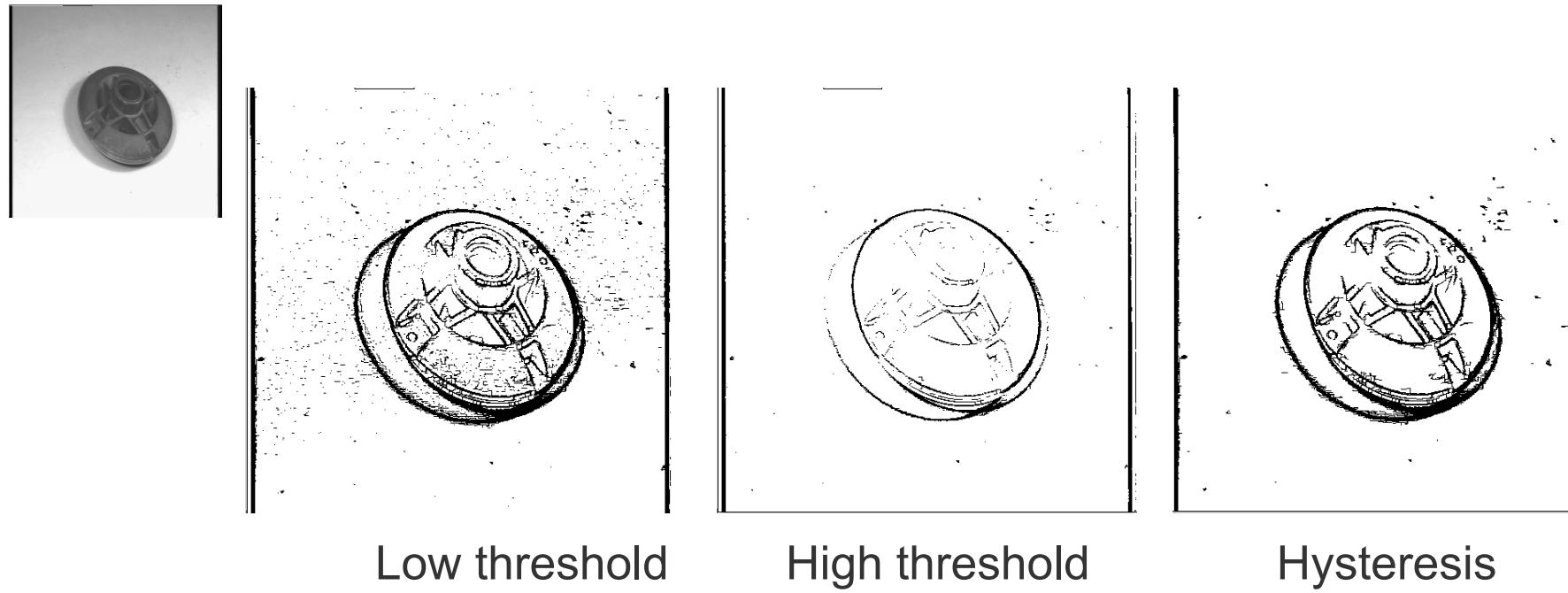
Thresholding with Hysteresis

Hysteresis fills in most of most of the gaps



Problem:
pixels along
this edge
didn't
survive the
thresholding

Thresholding with Hysteresis



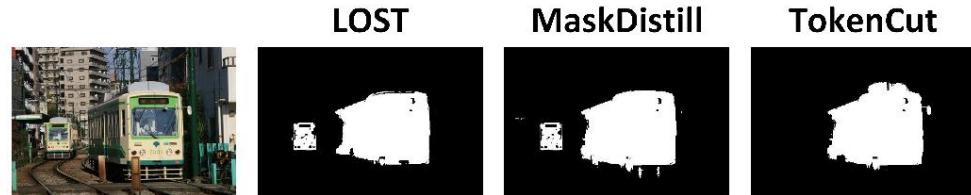
What Canny Did

- Showed that 1st derivative of a Gaussian smoothed image is the optimal way to detect step edges in noise
 - Explained why 1st derivatives are a good idea
- Designed the industry standard thresholding method
 - Non-maximal suppression
 - Thresholding with hysteresis
- Effectively solved the edge detection problem

COMP2005

Metrics for Image Segmentation

Why do we need metrics for image segmentation?



◆ Which algorithm is the best?

Hard to answer by looking at the results ...

◆ What we can do with metrics?

- Automatic evaluation
- Comparison between different approaches
- Evaluation from multiple perspectives

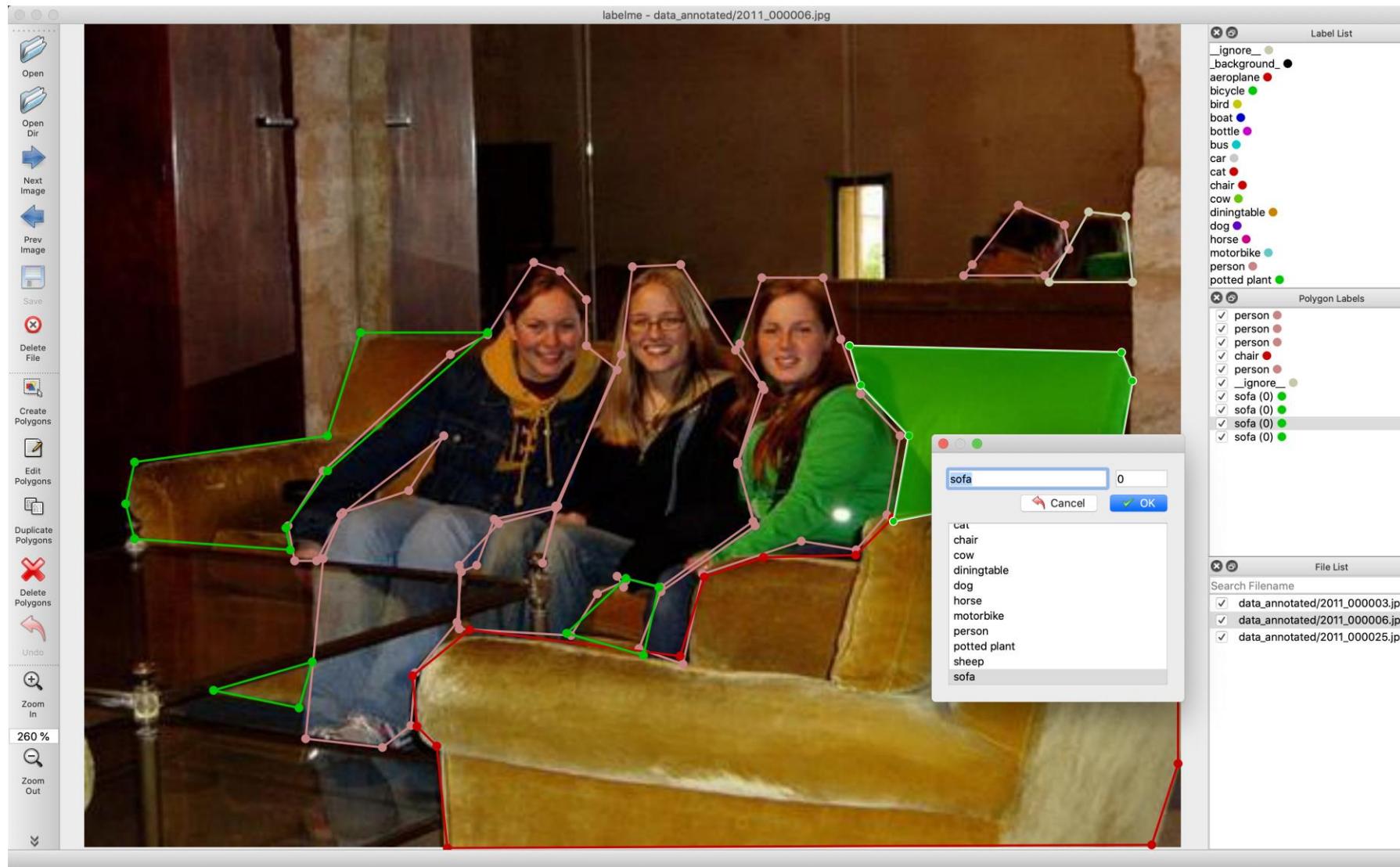
...

Siméoni, Oriane, et al. "Localizing Objects with Self-Supervised Transformers and no Labels." BMVC 2021-32nd British Machine Vision Conference. 2021.

Van Gansbeke, et al. Discovering object masks with transformers for unsupervised semantic segmentation. arXiv preprint arXiv:2206.06363 (2022).

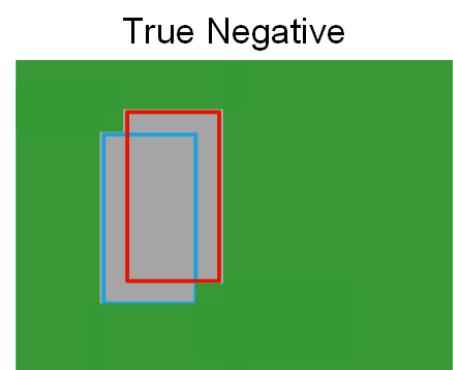
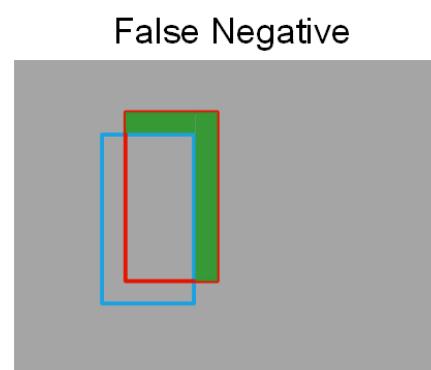
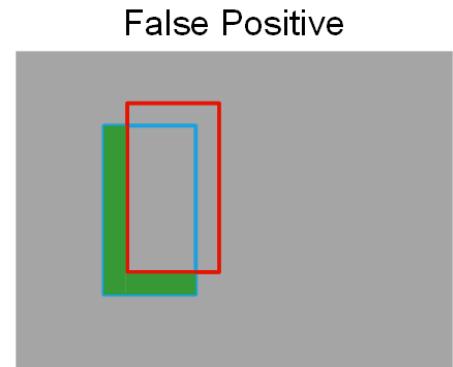
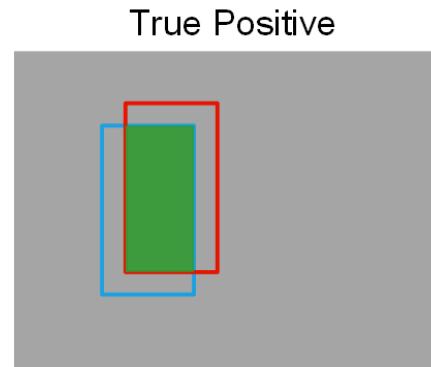
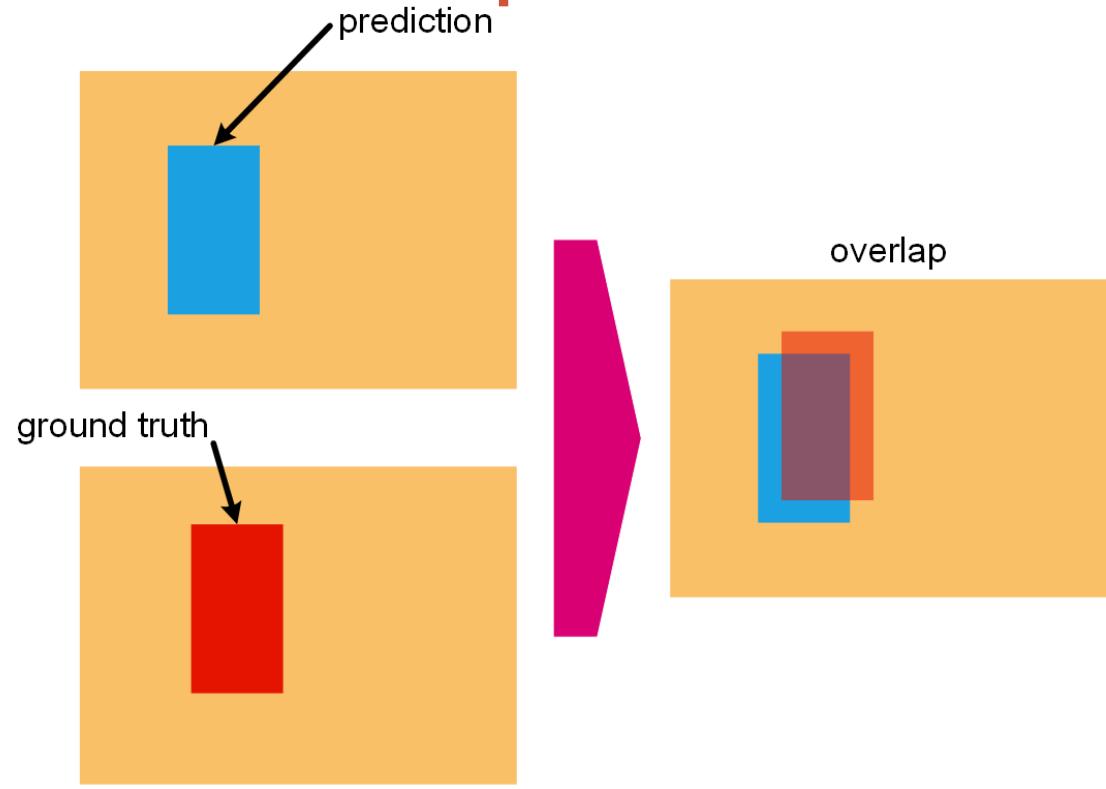
Wang, Yangtao, et al. "Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut." IEEE Transactions on Pattern Analysis and Machine Intelligence (2023).

First thing to do: preparing ground truth



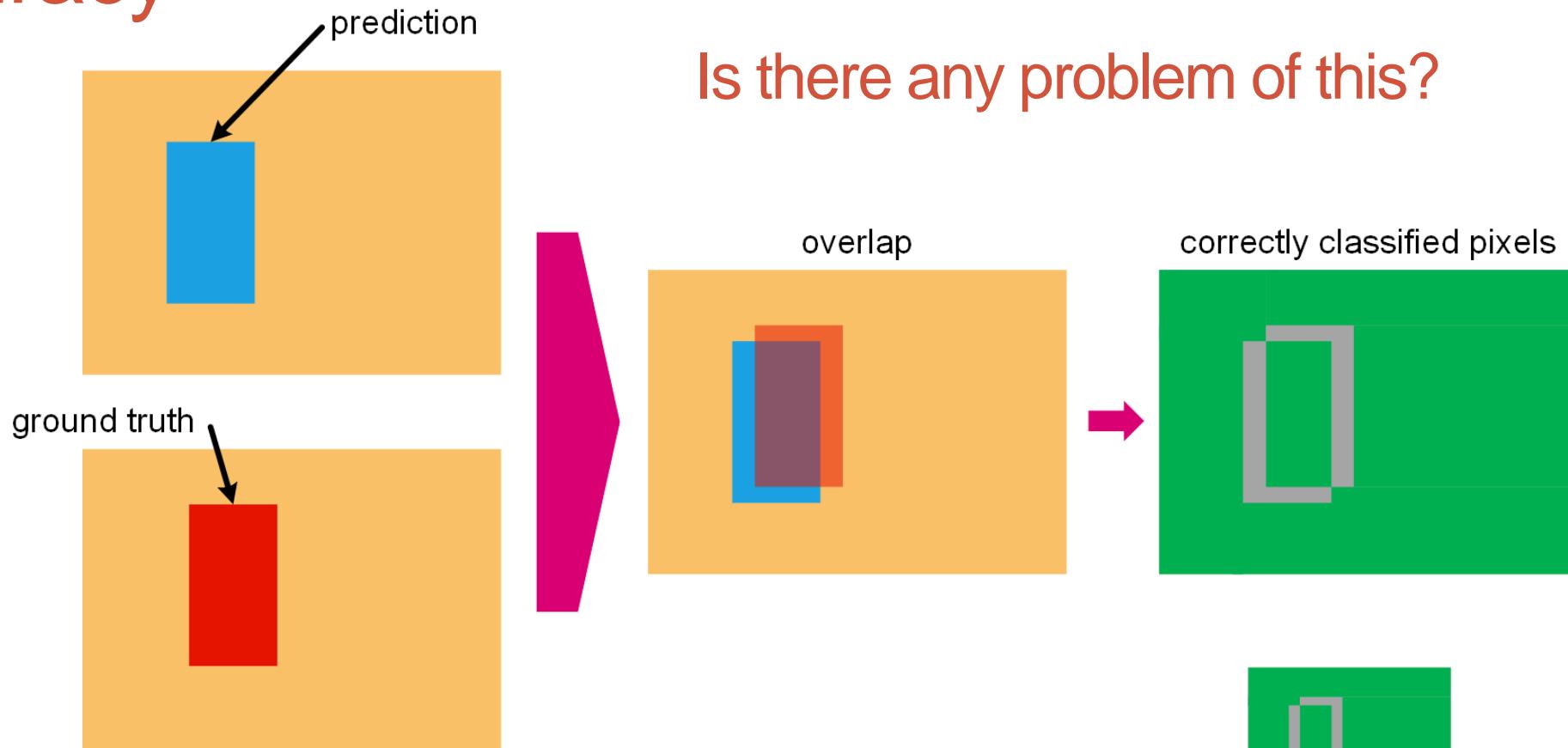
labelme.
<https://github.com/wkentaro/labelme>
Accessed March 13, 2024.

Core concepts



- True Positive (TP): prediction is positive and correct
- False Positive (FP): prediction is positive but incorrect
- False Negative (FN): prediction is negative but incorrect
- True Negative (TN): prediction is negative and correct

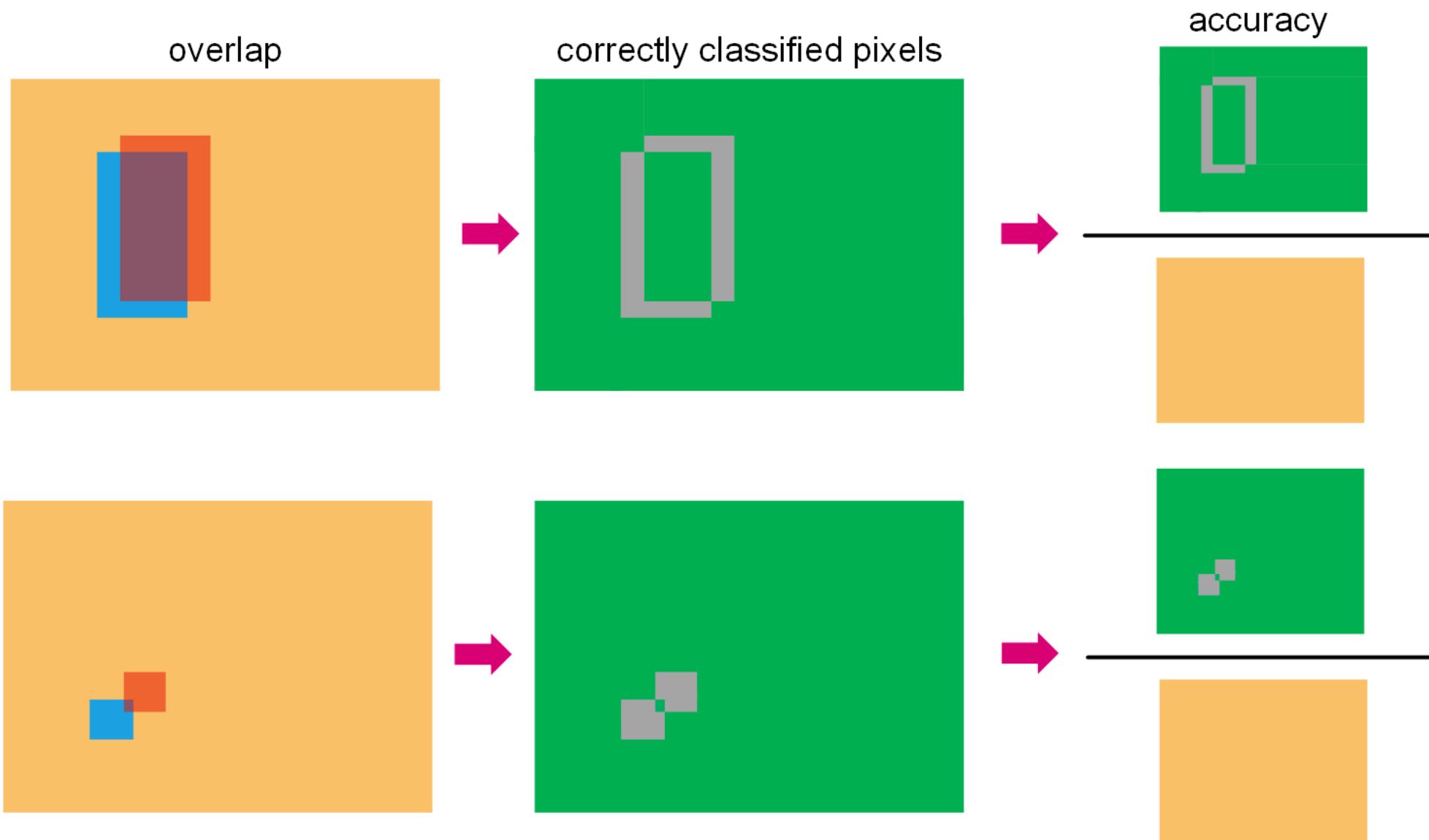
Accuracy



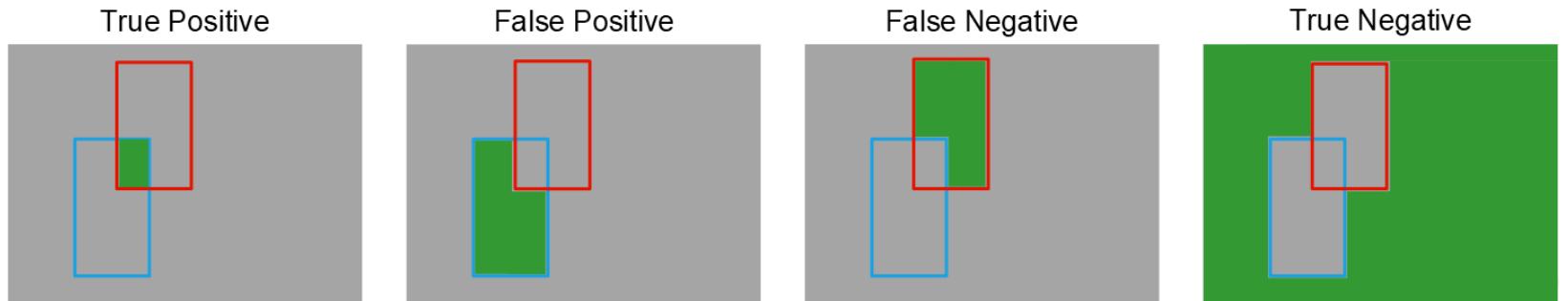
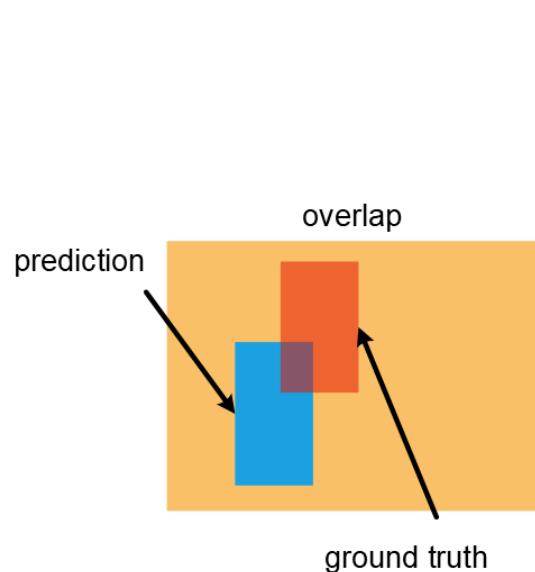
Is there any problem of this?

$$\frac{TP+TN}{TP+TN+FP+FN} = \text{accuracy} = \frac{\text{the number of correctly classified pixels}}{\text{the total number of pixels}} = \frac{\text{number of correctly classified pixels}}{\text{total number of pixels}}$$

Accuracy cannot deal with class imbalance



Precision, Recall, and F1 score

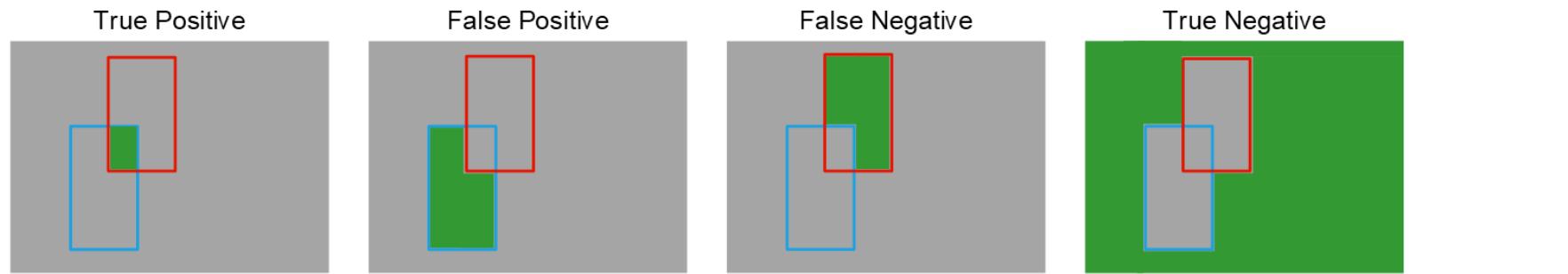
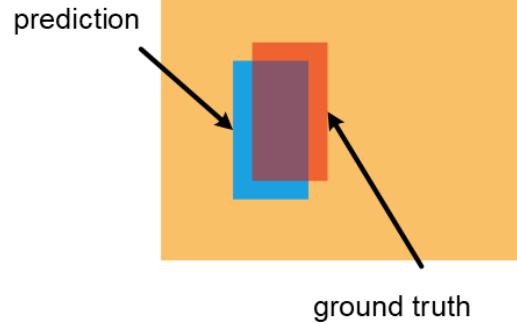


$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FP}} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

$$\begin{aligned}\text{F1 score} &= \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \\ &= \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}\end{aligned}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

Intersection over Union (IoU) and Dice coefficient

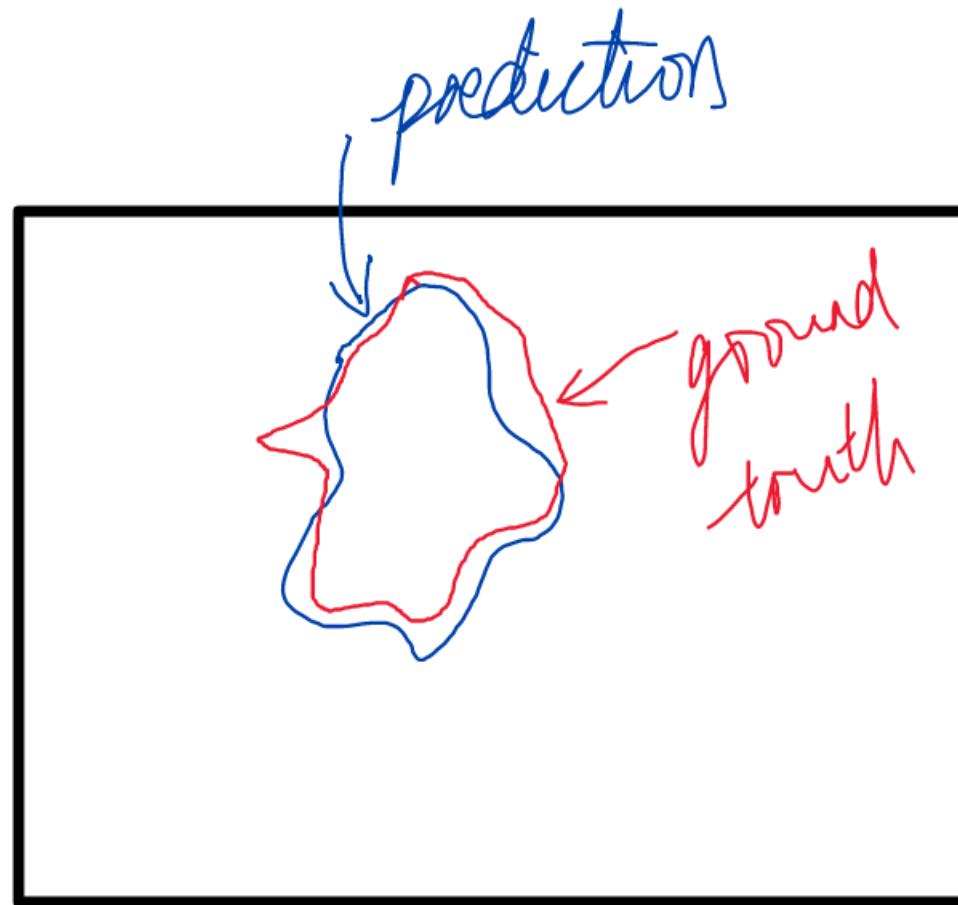


$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} = \frac{\text{TP}}{\text{TP} + \text{TP} + \text{FN}} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{TP}} = \frac{\text{TP}}{\text{TP} + \text{TP} + \text{TP}} = \frac{1}{3}$$

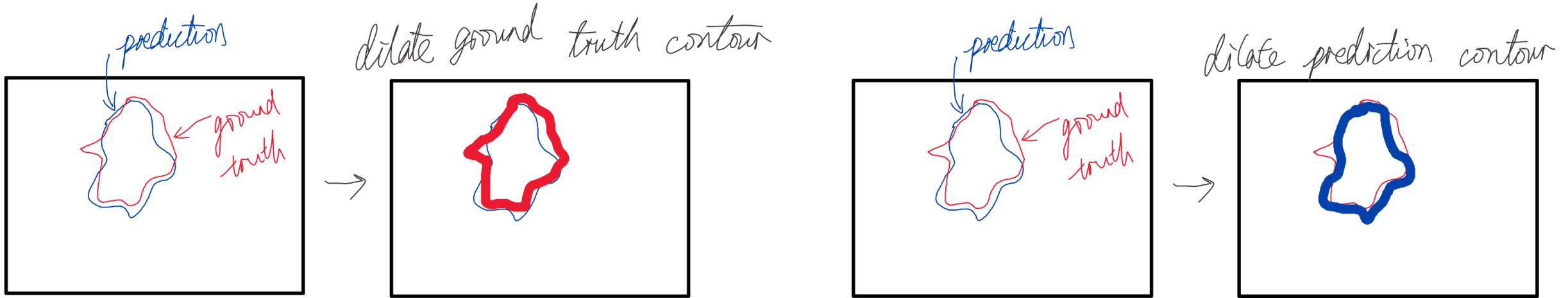
Is there any other potential problems?

$$\text{Dice} = \frac{\text{TP} + \text{TP}}{(\text{TP} + \text{FP}) + (\text{TP} + \text{FN})} = \frac{\text{TP} + \text{TP}}{\text{TP} + \text{TP} + \text{FN} + \text{TP}} = \frac{\text{TP} + \text{TP}}{\text{TP} + \text{TP} + \text{TP} + \text{TP}} = \frac{2}{1 + \text{IoU}} = \frac{2}{1 + \frac{1}{3}} = \frac{3}{2}$$

Contour doesn't match



F-measure



$$\text{contour precision} = \frac{\text{the number of blue contour pixels covered by the dilated red contour}}{\text{the total number of blue contour pixels}}$$

$$\text{contour recall} = \frac{\text{the number of red contour pixels covered by the dilated blue contour}}{\text{the total number of red contour pixels}}$$

$$\text{F measure} = \frac{2 * \text{contour precision} * \text{contour recall}}{\text{contour precision} + \text{contour recall}}$$



University of
Nottingham

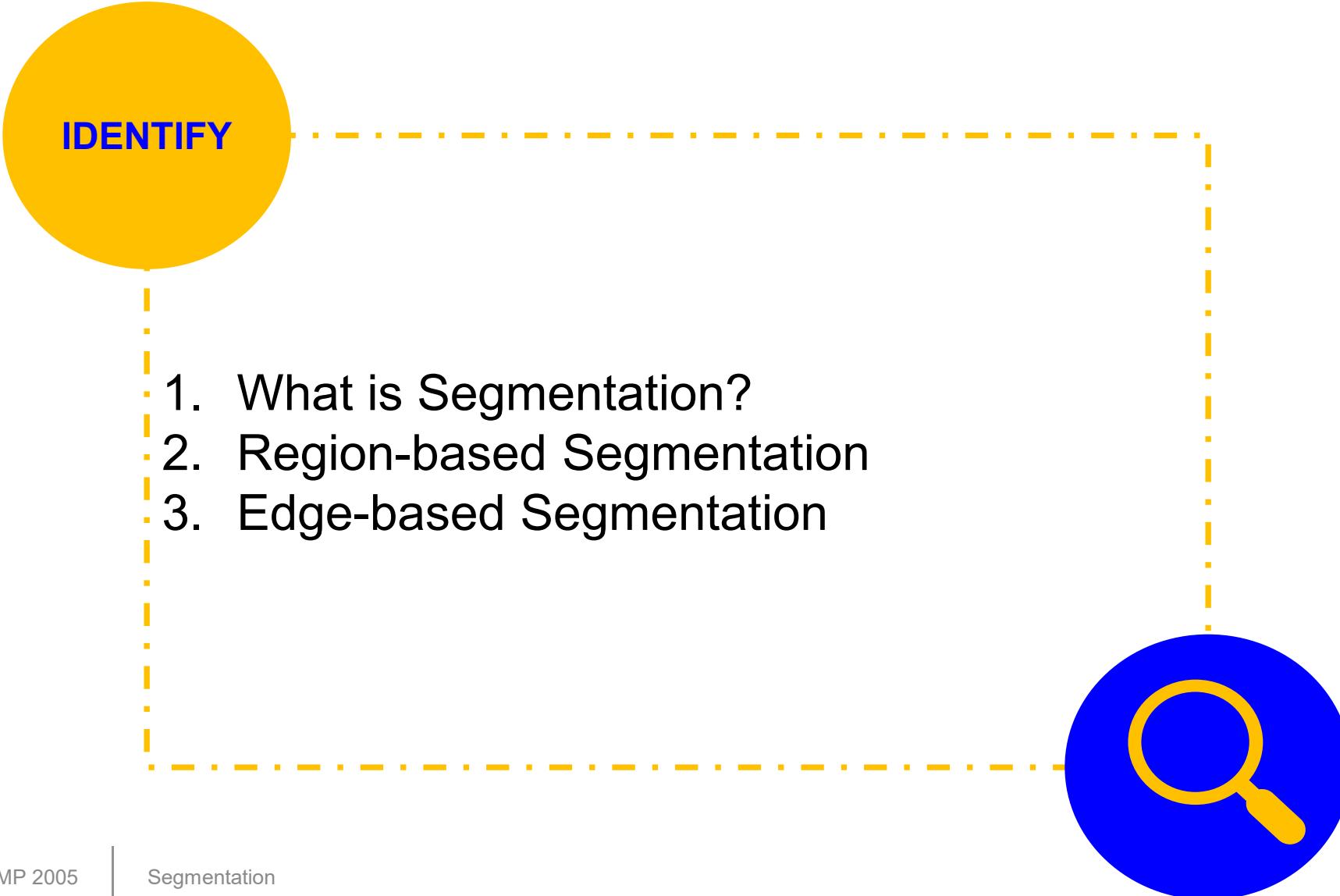
UK | CHINA | MALAYSIA

COMP-2032
**Introduction to
Image Processing**

Lecture 7
Segmentation



Learning Outcomes





What is Segmentation?



Image Segmentation

?

A common task in image analysis & computer vision

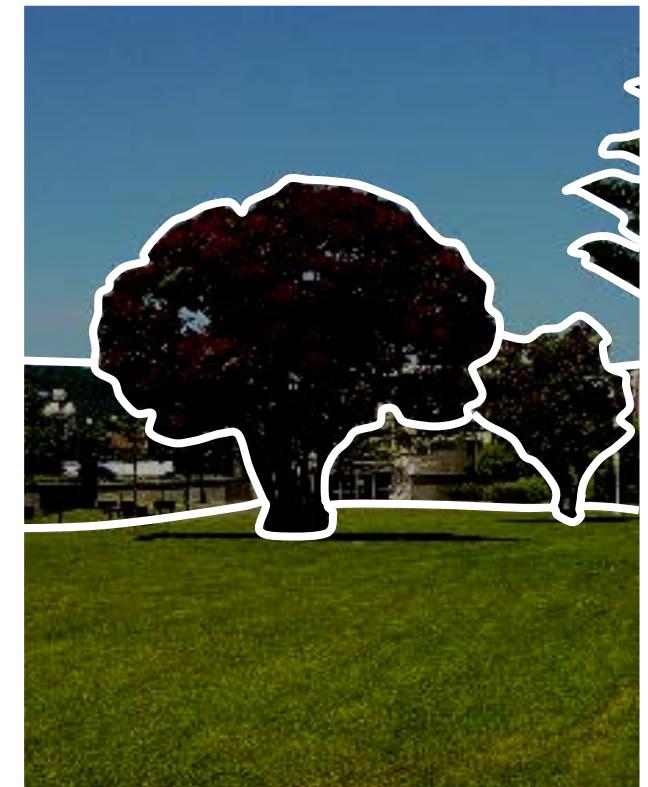
To identify **meaningful** regions

Why do it?

We can partition or group pixels according to local image properties

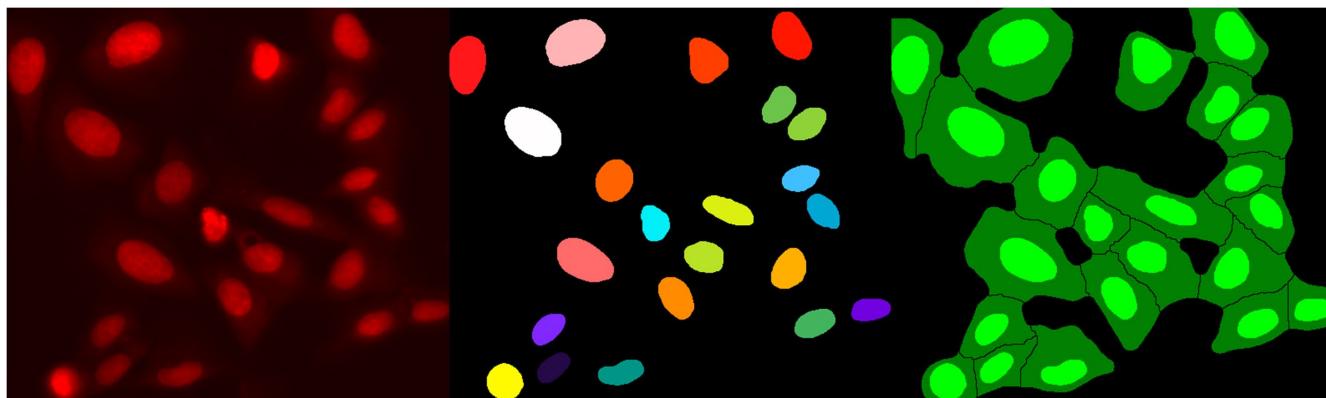
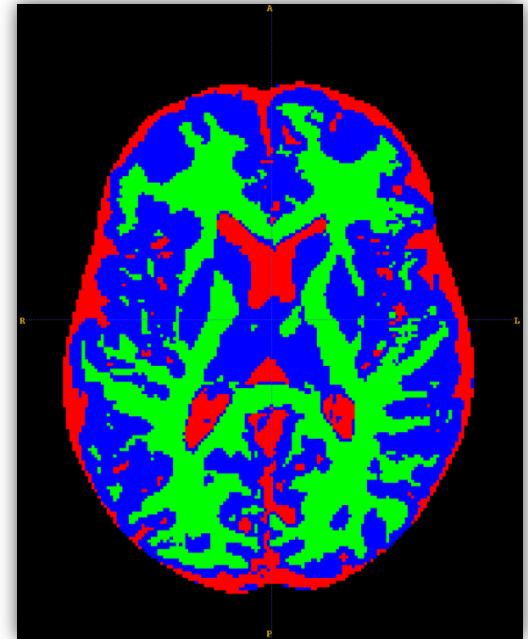
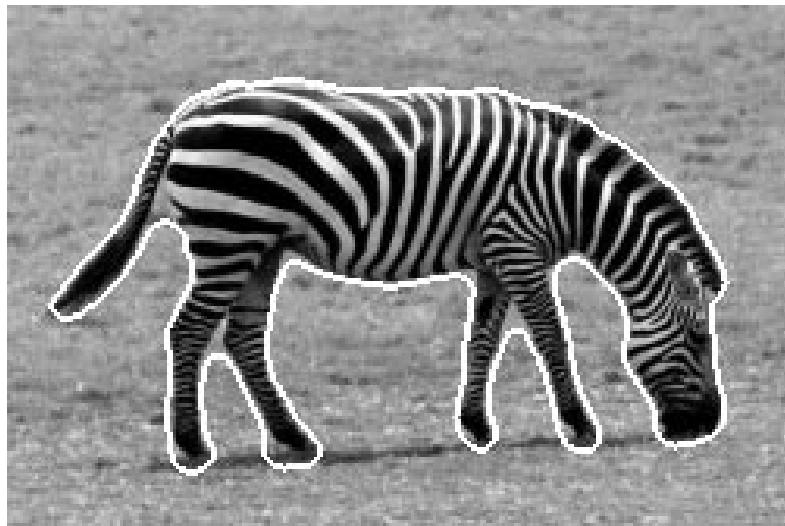
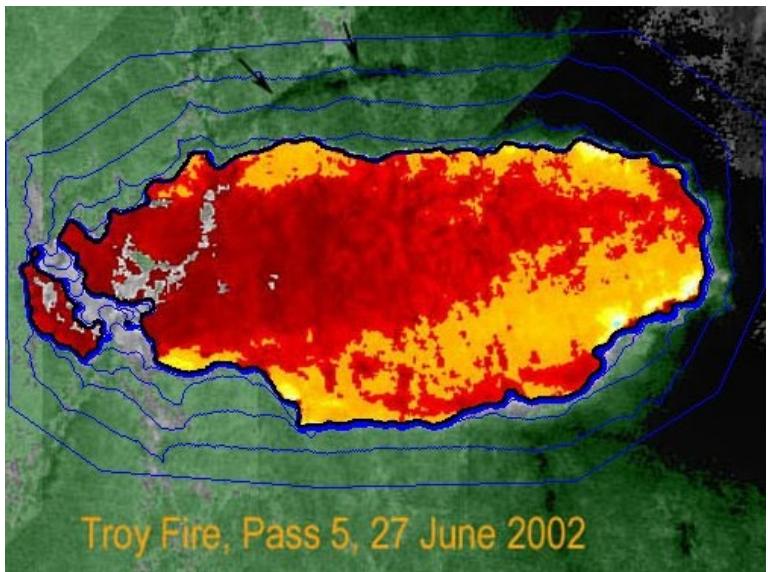
- Intensity or colour from original images, or computed values based on image operators
- Textures or patterns that are unique to each type of region
- Spectral profiles that provide multidimensional image data

Elaborate systems may use a combination of properties





Applications





Approaches

Many different approaches have been taken to the segmentation problem

Seeks groups of similar pixels, with no regard for where they are – views images as uncorrelated data

Clustering

- Focus on finding physically connected sets of pixels
- E.g., region growing, split and merge

Region-based

- Emphasise the boundaries between regions
- E.g., watersheds

Edge-based

Thresholding + connected components is a form of segmentation, but treats grey/colour and spatial information independently

Note



Region-Based Segmentation



Region-based Segmentation

We want smooth regions in the image

- We still want the pixels in each region to be similar, and those in adjacent regions to be different
- One way to do this is to work with regions rather than pixels

Region Growing

Start with a small 'seed' and expand by adding similar pixels

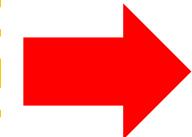
Split & Merge

- Splitting divides regions that are inconsistent
- Merging combines adjacent regions that are consistent



Region Growing

Region growing starts with a small patch of seed pixels



- Compute statistics about the region
- Check neighbours to see if they can be added
- Recompute the statistics

Algorithm

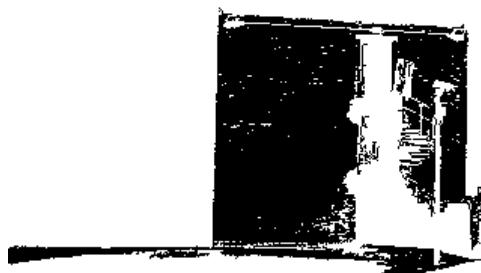
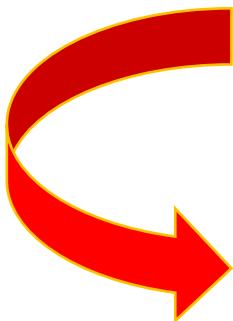
This procedure **repeats** until the region stops growing

- Simple example: we *compute the mean grey level of pixels in the region*
- Neighbours are added if their grey level is near the average

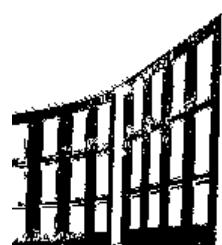




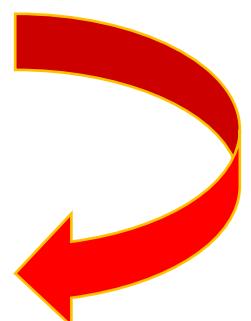
Region Growing Example



Output 1



Output 2



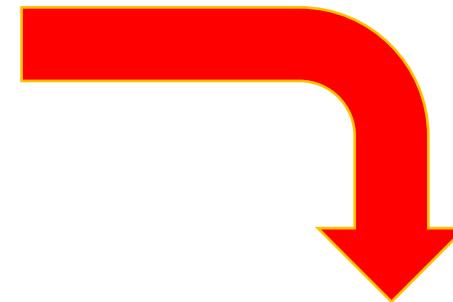
Output 3



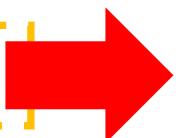
Split and Merge - Split

We start by taking the whole image to be one region

- We compute some measure of internal similarity
- If this indicates there is too much variety, we divide the region
- Repeat until no more splits, or we reach a minimum region size



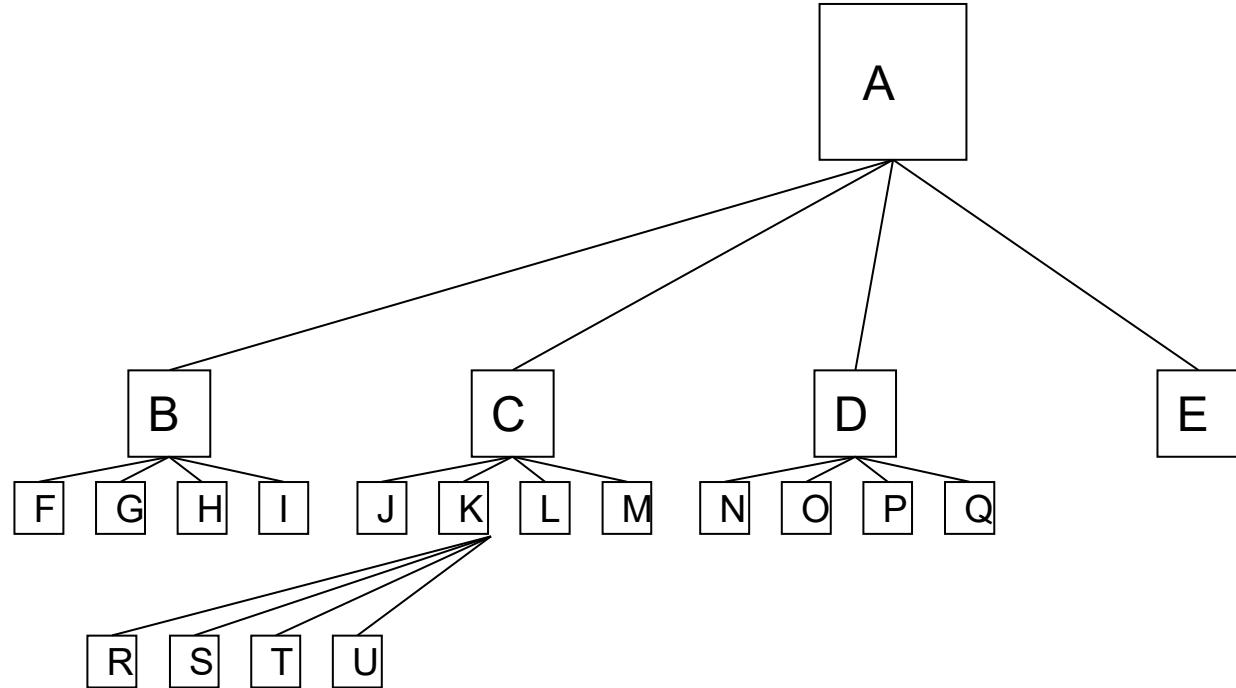
Some details are needed



- How do we measure similarity? – standard deviation are commonly used
- How do we determine whether to split or not? – thresholding is easy
- How do we split regions? – quadtrees are a common method



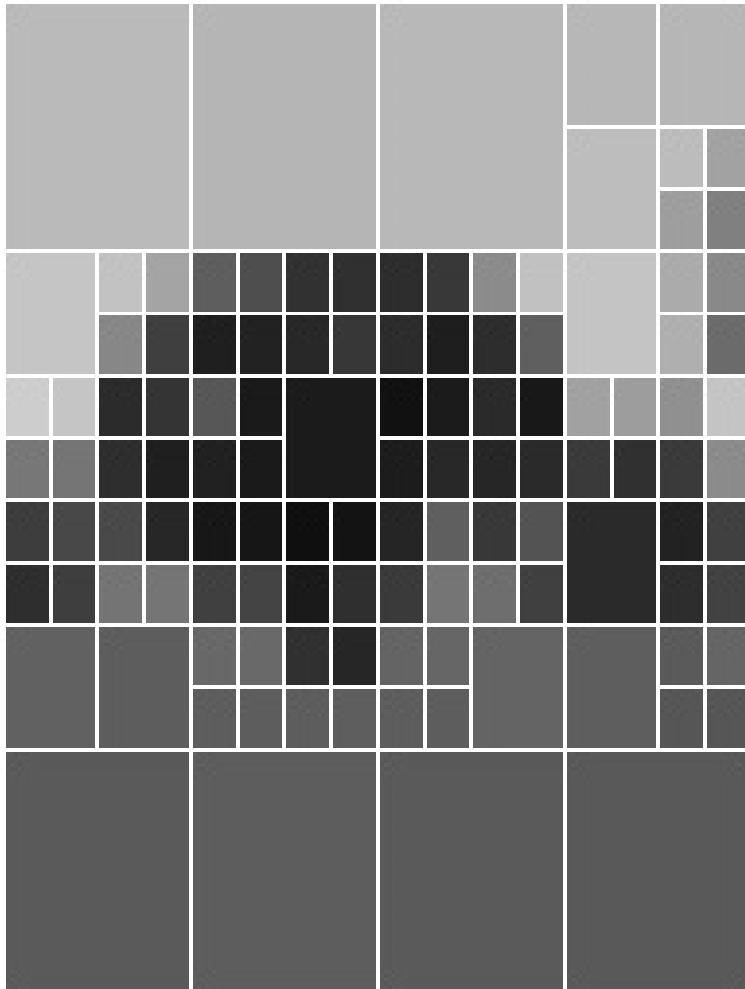
Quadtrees



F	G	J	R	S
H	I	L	T	U
N	O			
P	Q	E		

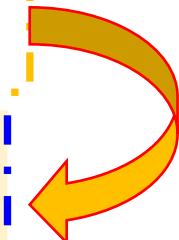


Example - Splitting



We'll use the tree image again

- Splitting based on intensity (*could use something else*)
- Splitting based on standard deviation, with a threshold of 25
- Split using quadtree with a maximum of 5 level

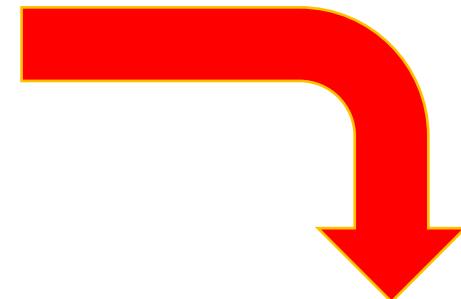




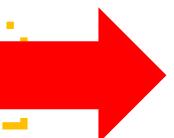
Split and Merge - Merge

Splitting give us...

- Regions that are small, consistent, or both
- Rather too many regions, as adjacent ones may be very similar
- We can now combine adjacent regions to make bigger ones



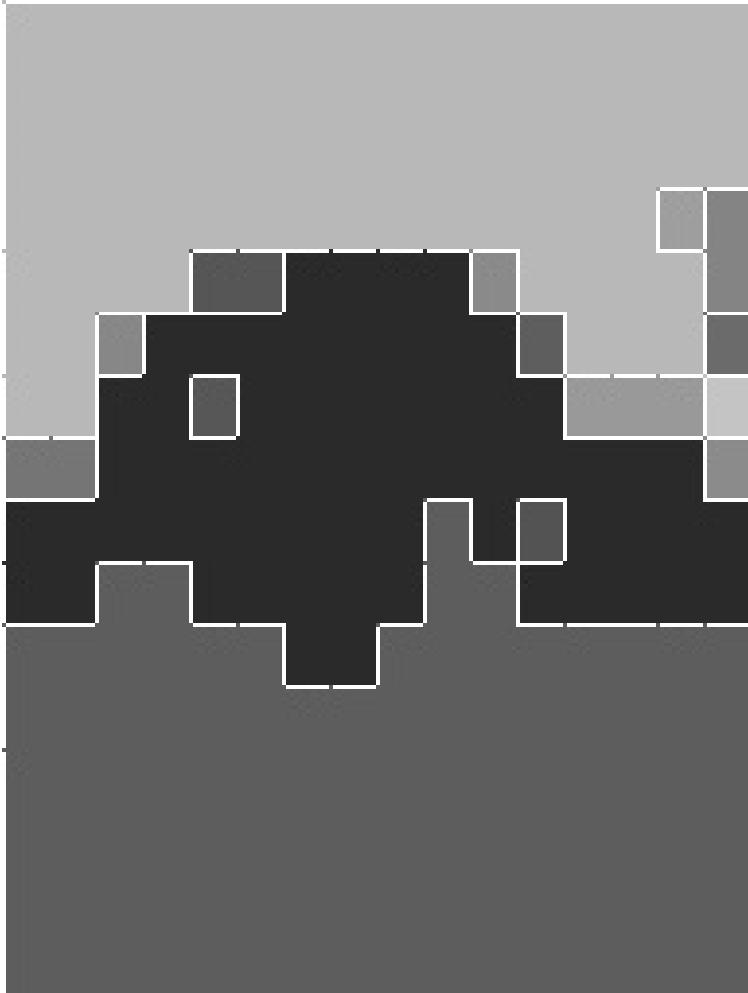
Merging



- We merge two regions if they are adjacent and similar
- Need a measure of similarity – **can compare their mean grey level, or use statistical tests**
- Repeat the merging until you can do no more

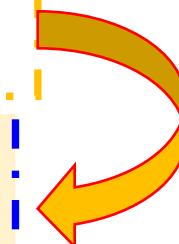


Example - Merging



We consider merging adjacent regions

- Two regions are merged if their mean grey levels differ by less than 25
- This leads to less regularly shaped regions, but they are larger and still consistent





Break



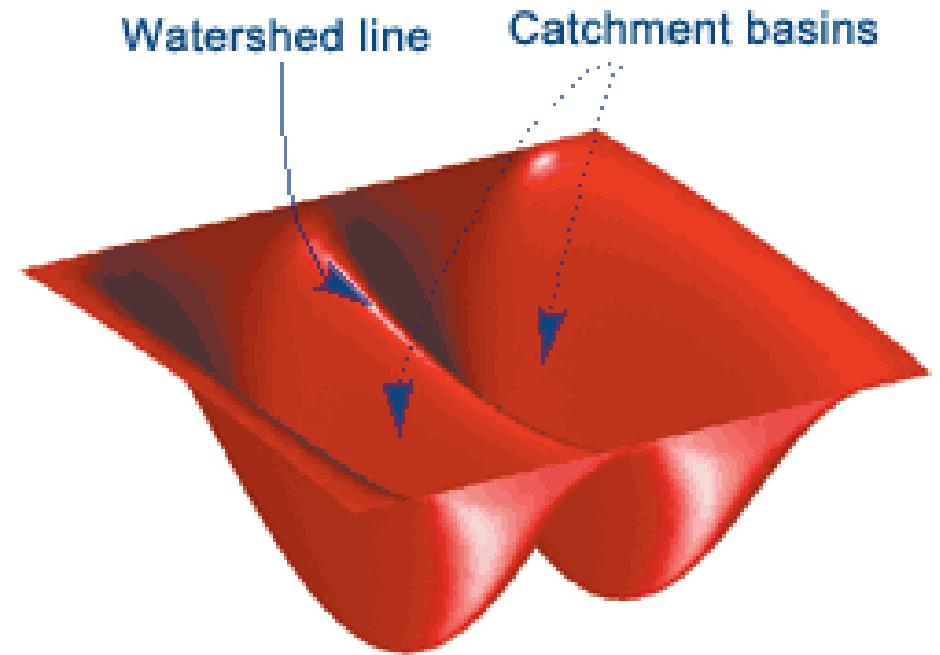


Edge-based Segmentation (*Watersheds*)



Edge-based Segmentation

- Do region-based methods focus too much on regions?
- Edge represent discontinuities in image intensity
- Regions should then be areas without edges, and should be bounded by edges
- One class of edge-based segmentation uses **watersheds**



In geography, a watershed is a ridge which divides rainfall into basins on either side



Catchments in images

We can view the **gradient** image as a terrain

- Areas of high gradient are high points on the terrain
- Catchment basins are regions in the image
- Watersheds are the lines dividing them



We don't have to use the usual intensity gradient

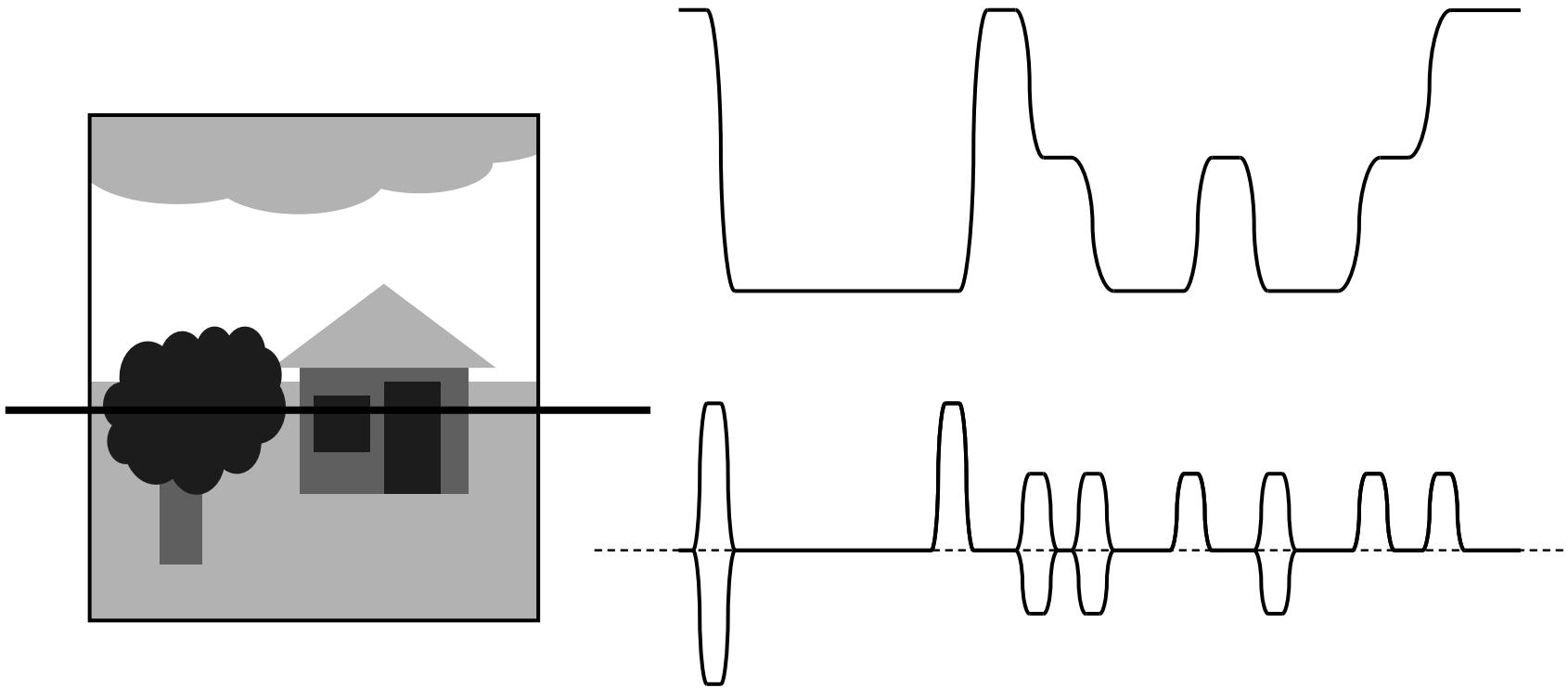
- Gradients can be computed from **hue** etc., if we want
- Any value that is low within a region and high at boundaries could be used



Using gradients is common, though...

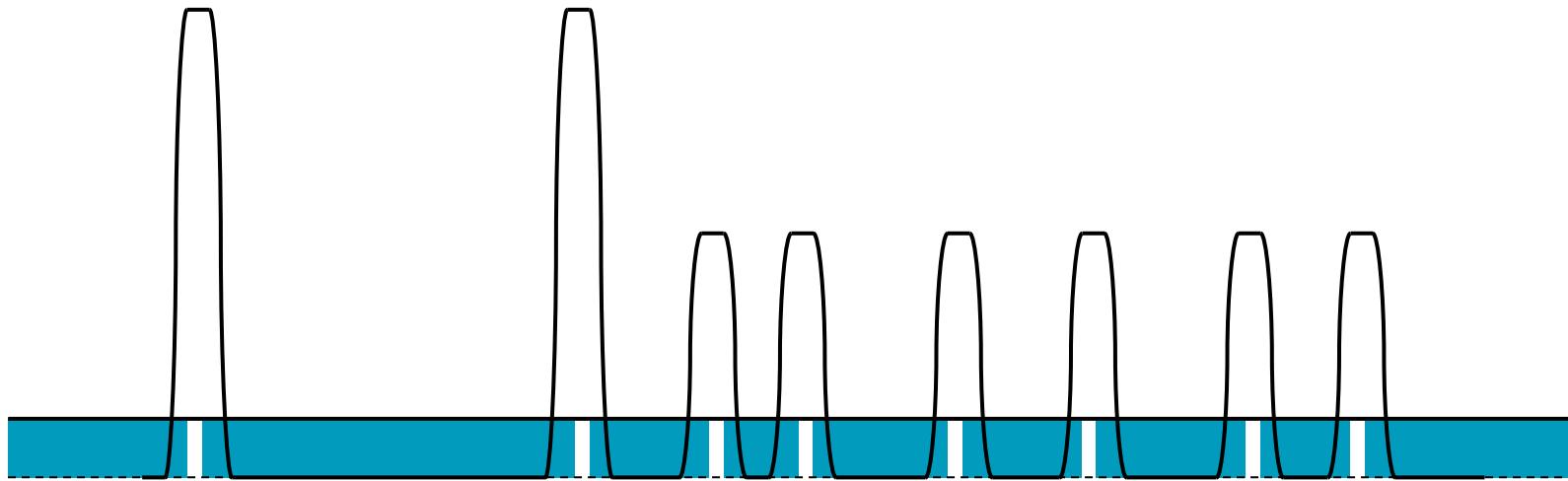


Gradients in Highlight Edges



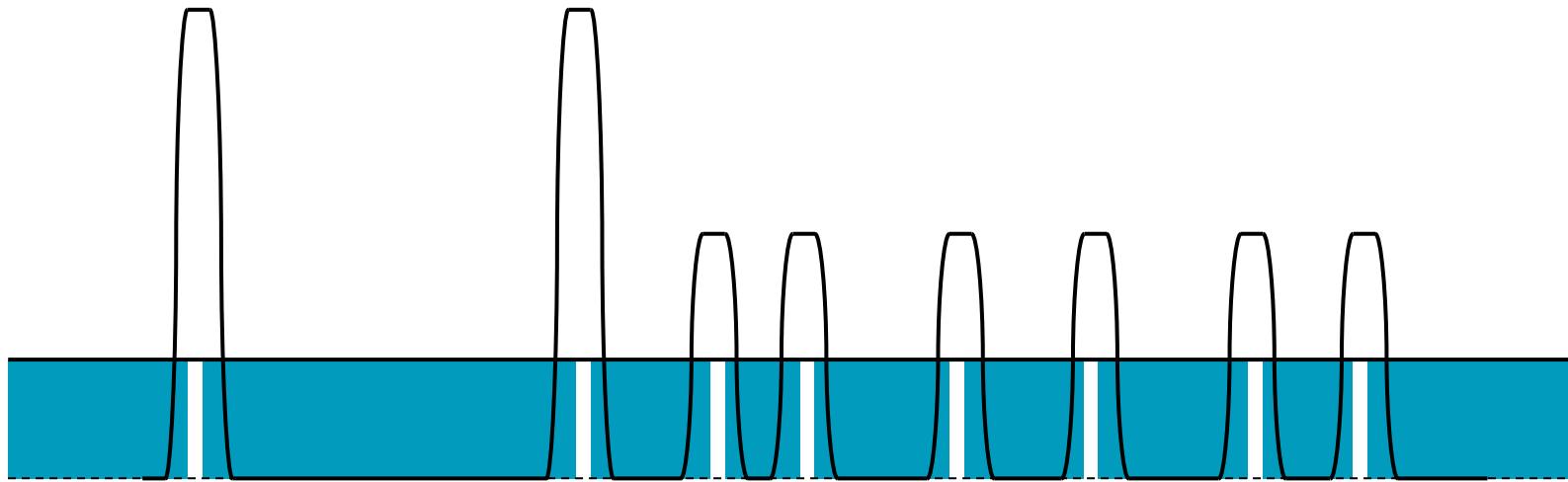


Immersion to Find Regions



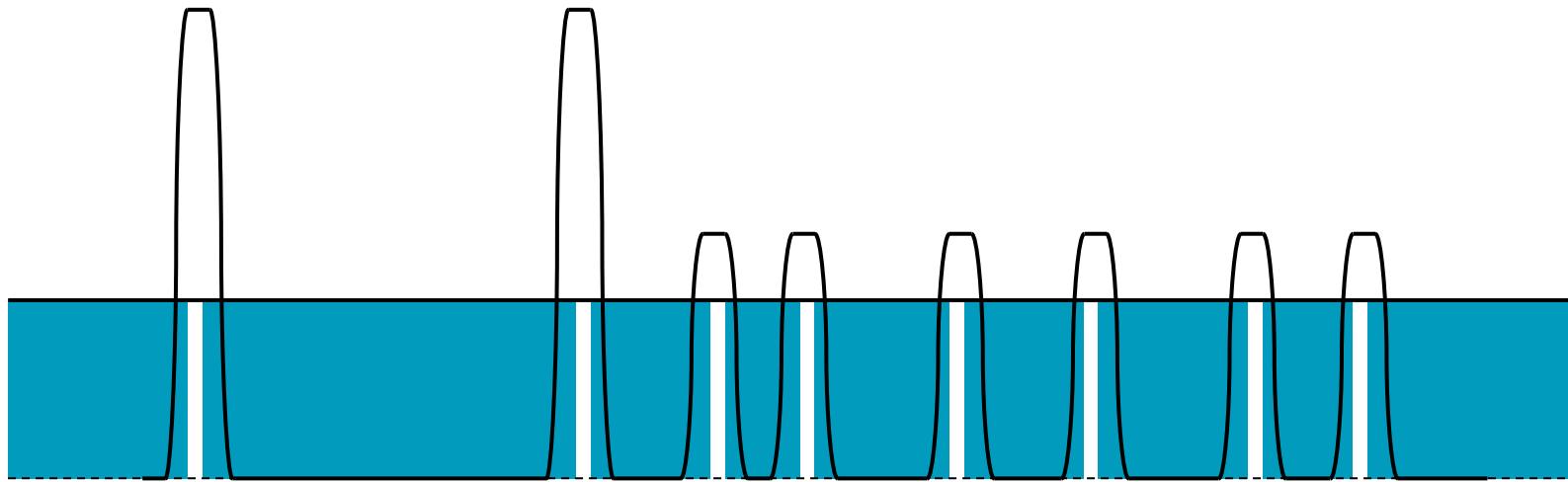


Immersion to Find Regions



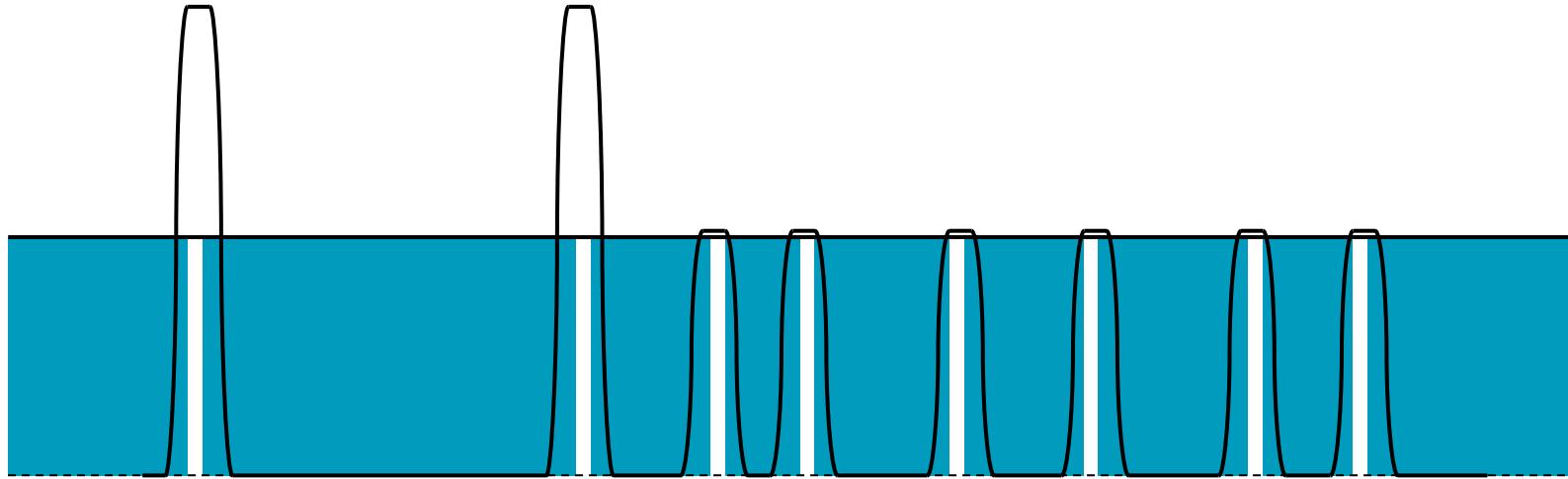


Immersion to Find Regions



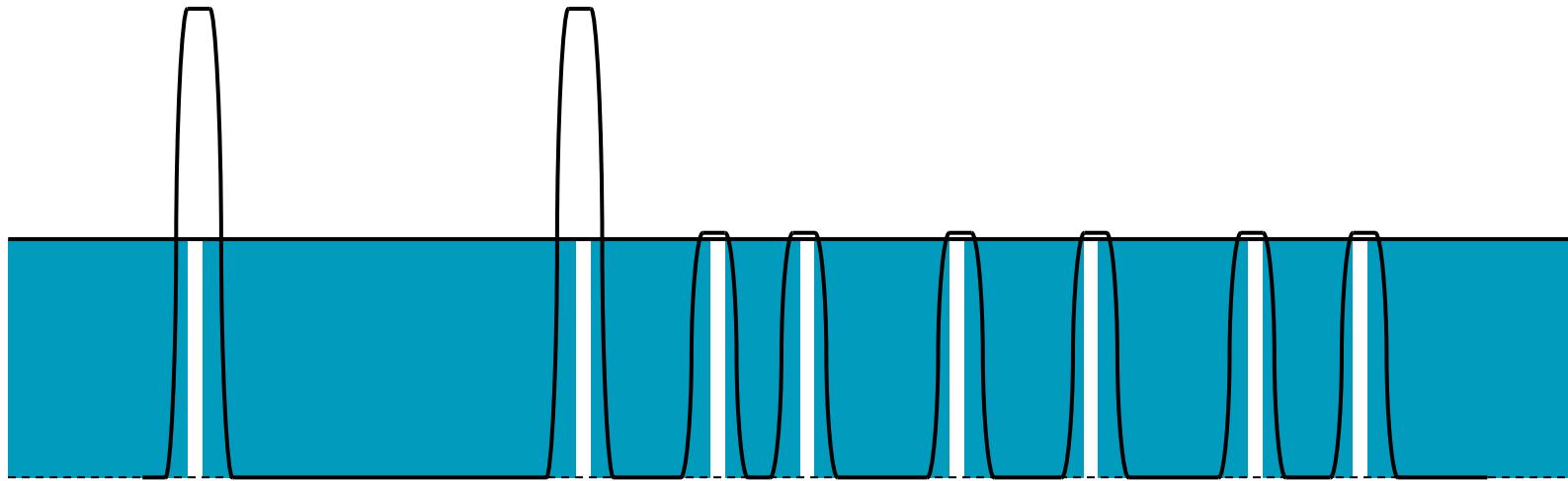


Immersion to Find Regions



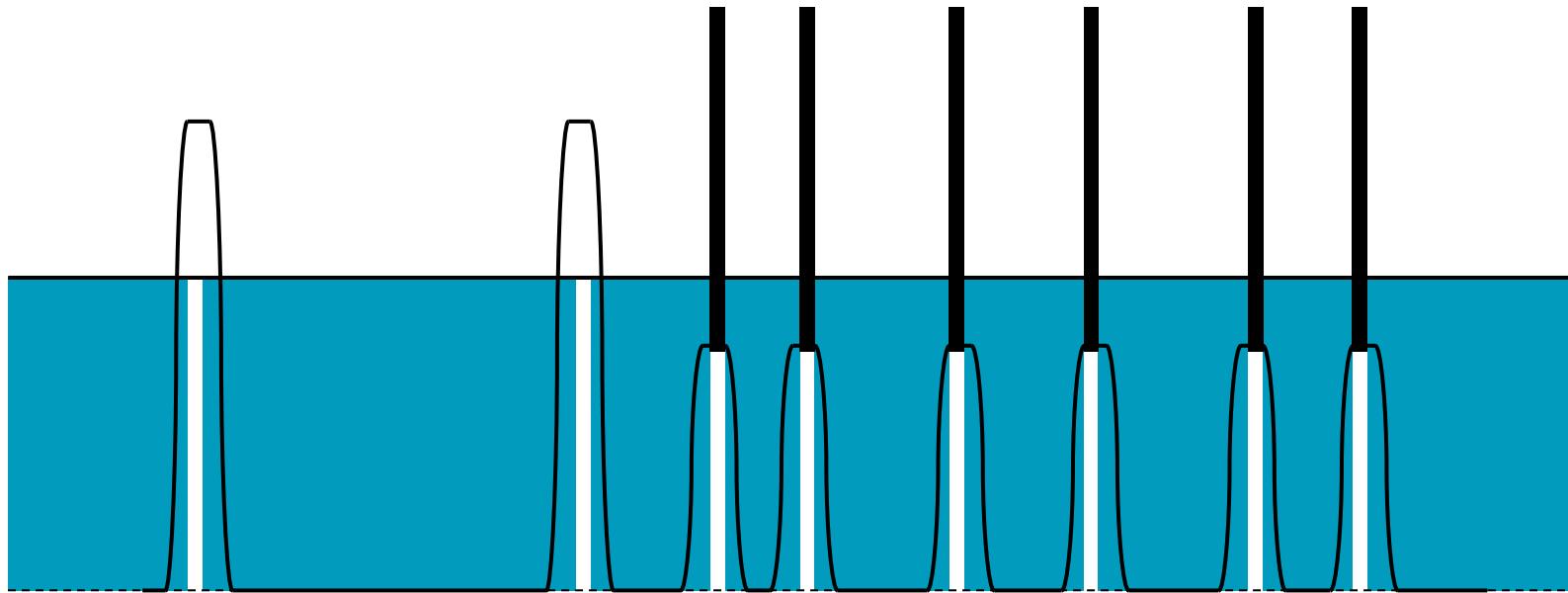


Immersion to Find Regions



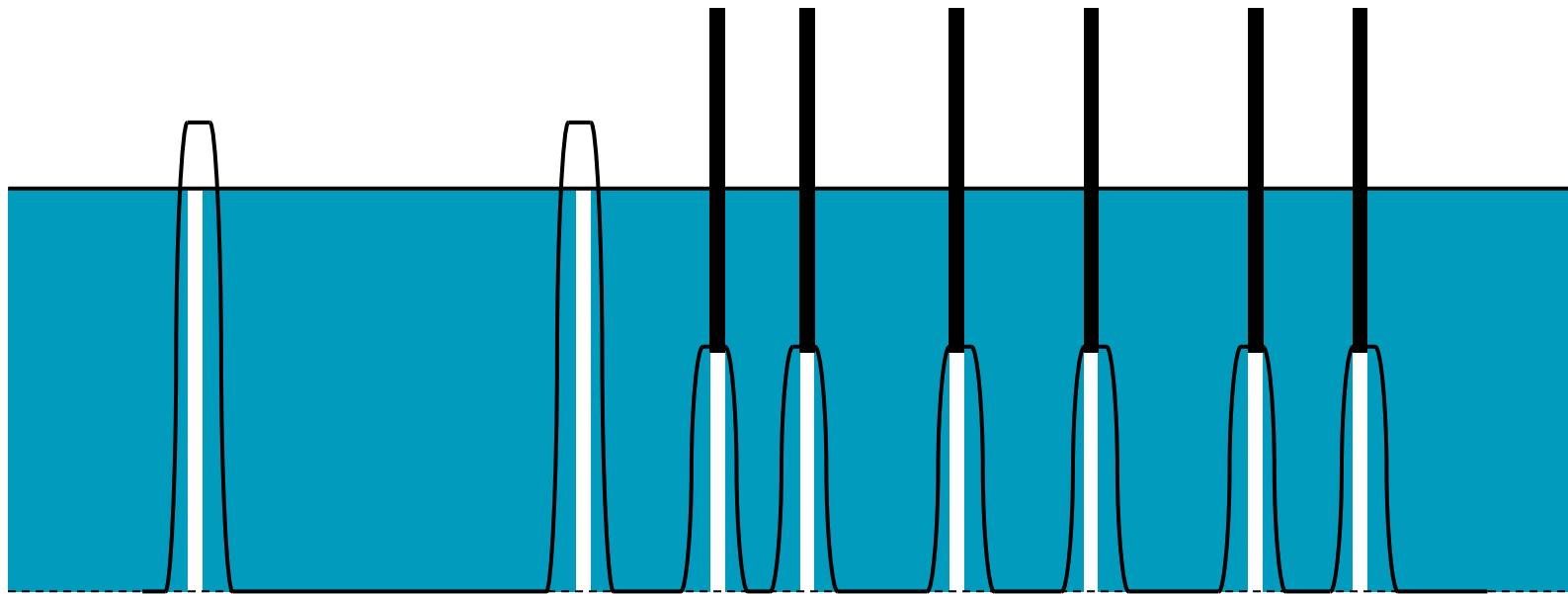


Immersion to Find Regions



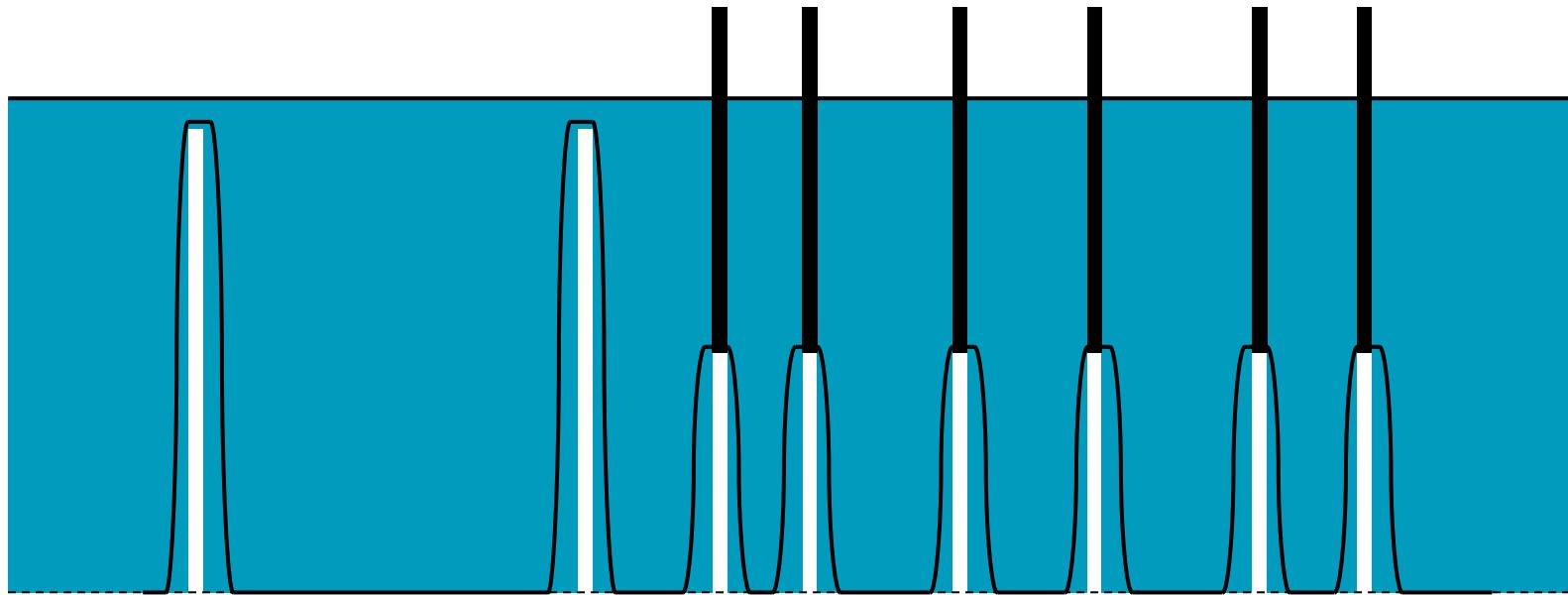


Immersion to Find Regions



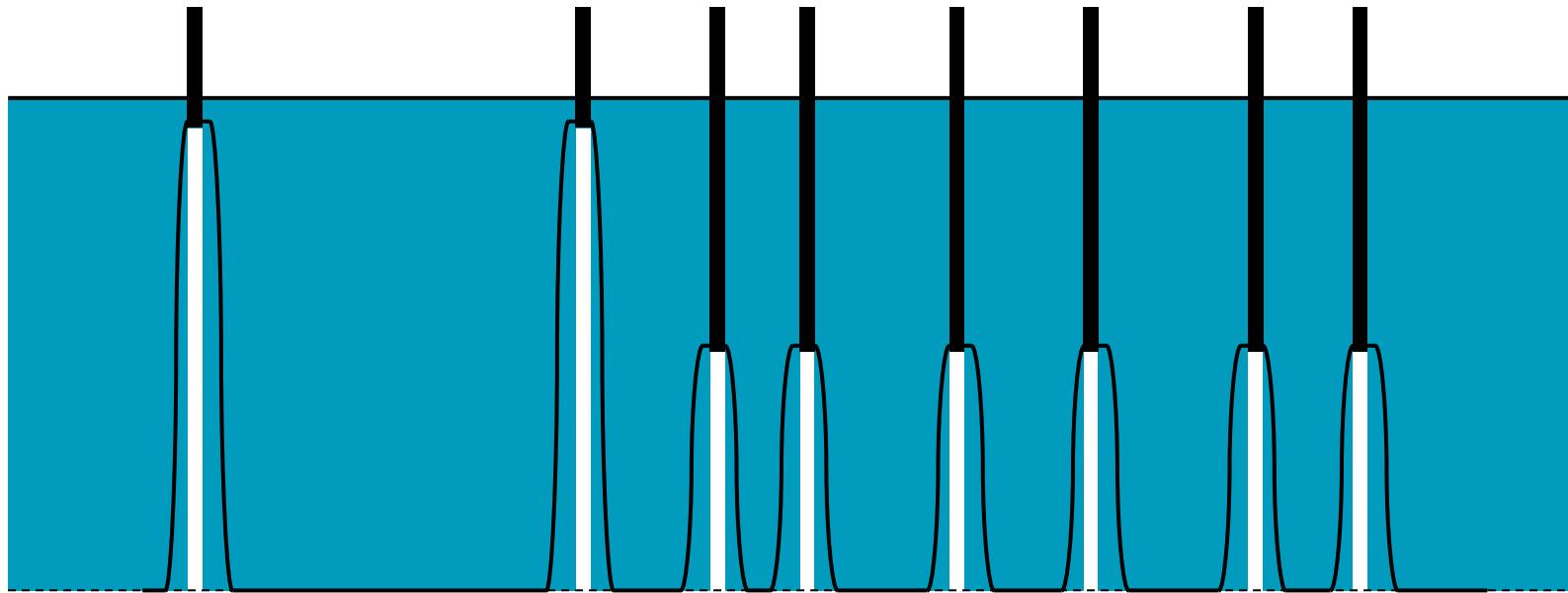


Immersion to Find Regions



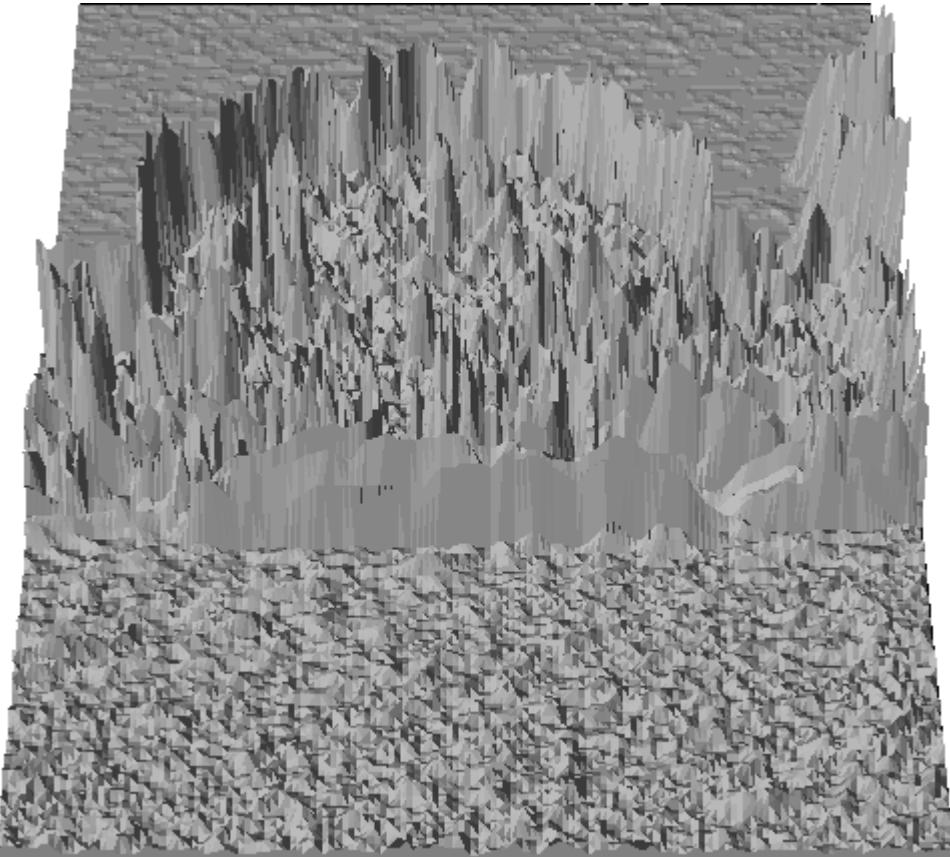


Immersion to Find Regions





Watersheds in Images



We start by finding images gradients

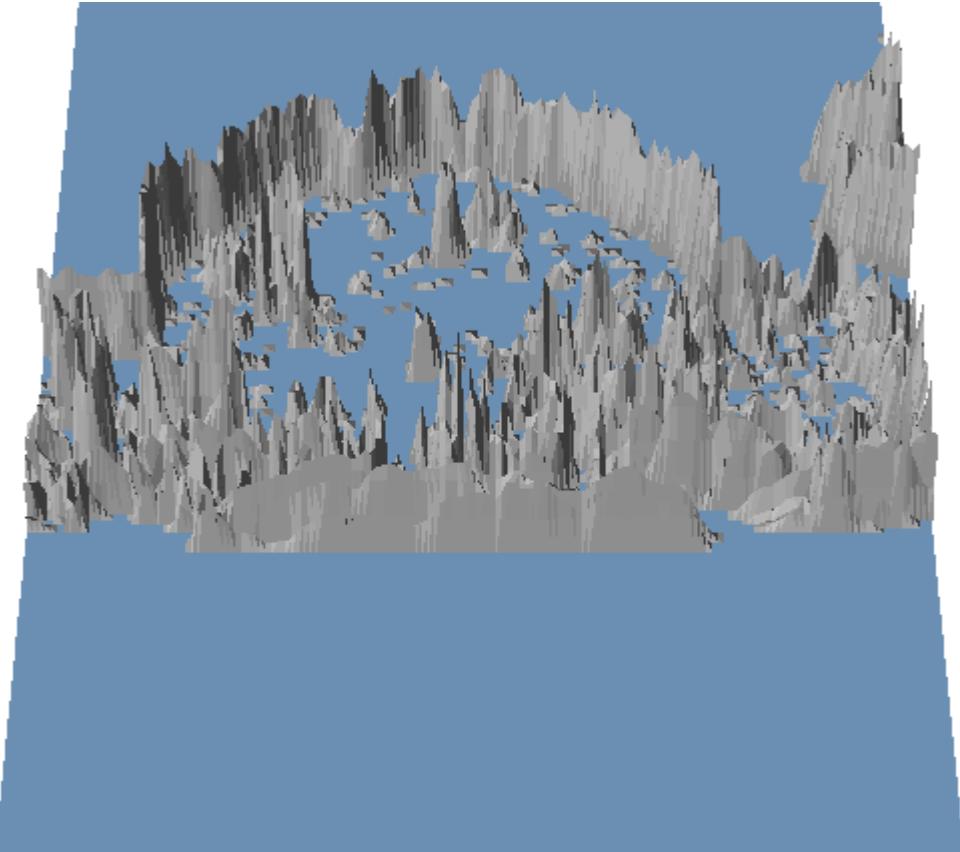


- Using methods like Sobel operators, we get a value for the gradient magnitude
- This can be viewed as a 3D ‘terrain’

$$\sqrt{I_x^2 + I_y^2}$$



Watersheds in Images



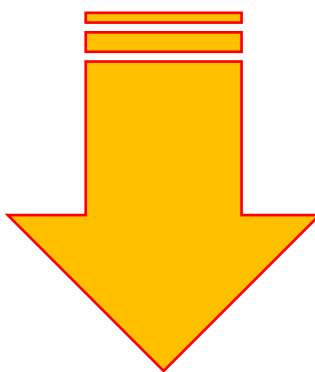
We then slowly flood the terrain

- Flat areas of the image become areas of low gradient, so are valleys in the terrain
- Edges in the image have high gradient and so are ridges in the terrain



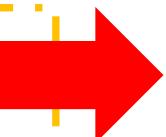
Watershed Algorithm

1. Sort the pixels: **low to high**
2. For each pixels



- If its neighbours are all unlabelled, give it a new label
- If it has neighbours with a single label, it gets that label
- If it has neighbours with two or more labels, it is a watershed

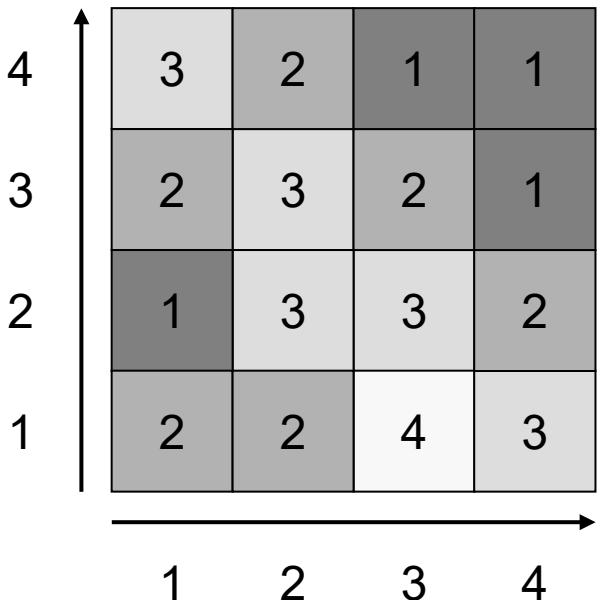
This is a very basic version



- It has certain problems in that it can give 'thick' watersheds rather than fine lines
- It is sensitive to noise and so can generate lots of small regions
- It does show the basic plan, though



An Example



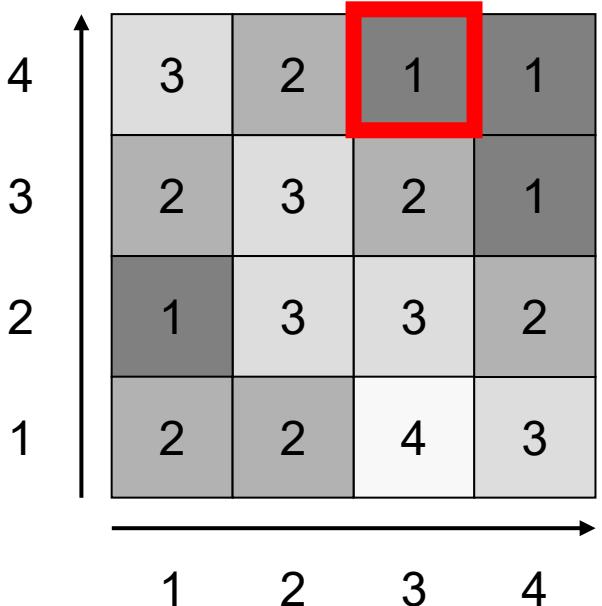
Sorted list:

- $(3,4) = 1$
- $(4,4) = 1$
- $(4,3) = 1$
- $(1,2) = 1$
- $(2,4) = 2$
- $(1,3) = 2$
- $(3,3) = 2$
- $(4,2) = 2$
- $(1,1) = 2$
- $(2,1) = 2$
- $(1,4) = 3$
- $(2,3) = 3$
- $(2,2) = 3$
- $(3,2) = 3$
- $(4,1) = 3$
- $(3,1) = 4$

Labels



An Example



Sorted list:

$$(3,4) = 1$$

$$(4,4) = 1$$

$$(4,3) = 1$$

$$(1,2) = 1$$

$$(2,4) = 2$$

$$(1,3) = 2$$

$$(3,3) = 2$$

$$(4,2) = 2$$

$$(1,1) = 2$$

$$(2,1) = 2$$

$$(1,4) = 3$$

$$(2,3) = 3$$

$$(2,2) = 3$$

$$(3,2) = 3$$

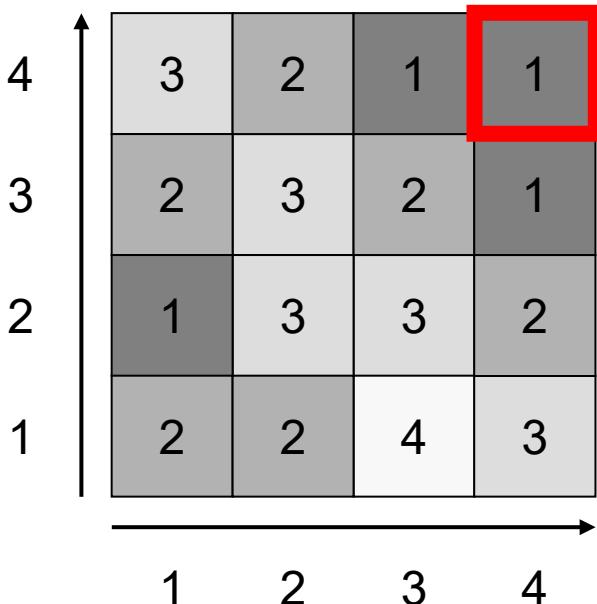
$$(4,1) = 3$$

$$(3,1) = 4$$

Labels



An Example



Sorted list:

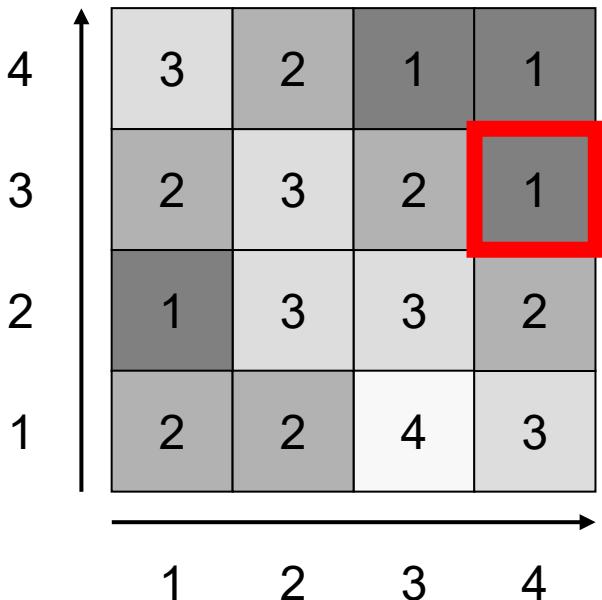
(3,4) = 1
(4,4) = 1
(4,3) = 1
(1,2) = 1
(2,4) = 2
(1,3) = 2
(3,3) = 2
(4,2) = 2
(1,1) = 2
(2,1) = 2
(1,4) = 3
(2,3) = 3
(2,2) = 3
(3,2) = 3
(4,1) = 3
(3,1) = 4

		A	

Labels



An Example



Sorted list:

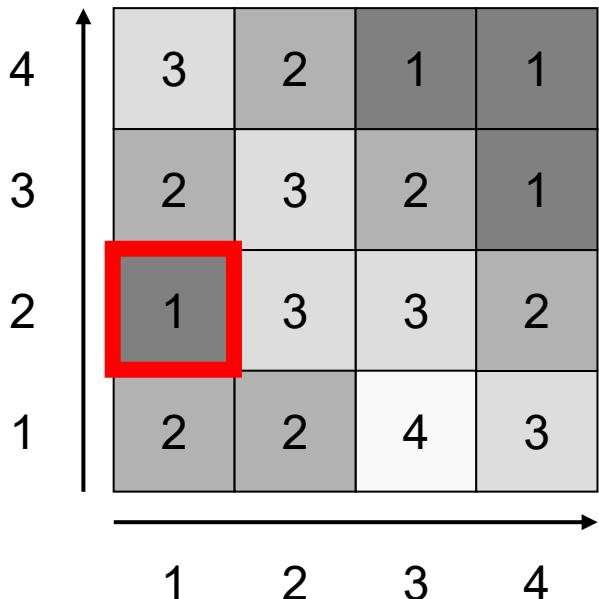
(3,4) = 1
(4,4) = 1
(4,3) = 1
(1,2) = 1
(2,4) = 2
(1,3) = 2
(3,3) = 2
(4,2) = 2
(1,1) = 2
(2,1) = 2
(1,4) = 3
(2,3) = 3
(2,2) = 3
(3,2) = 3
(4,1) = 3
(3,1) = 4

		A	A

Labels



An Example



Sorted list:

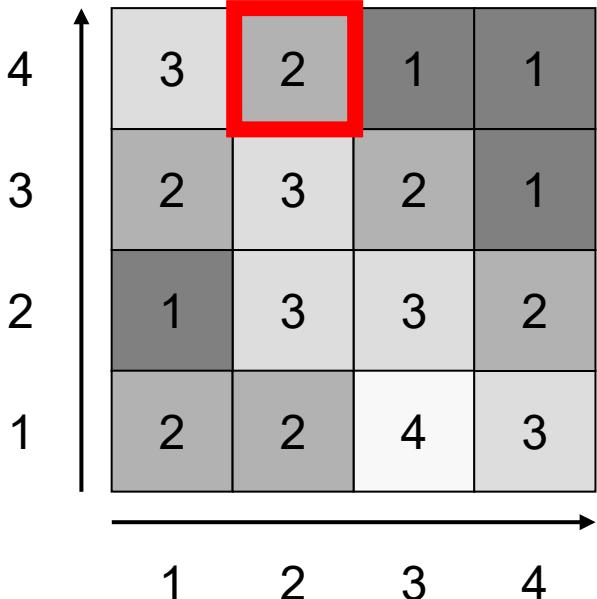
(3,4) = 1
(4,4) = 1
(4,3) = 1
(1,2) = 1
(2,4) = 2
(1,3) = 2
(3,3) = 2
(4,2) = 2
(1,1) = 2
(2,1) = 2
(1,4) = 3
(2,3) = 3
(2,2) = 3
(3,2) = 3
(4,1) = 3
(3,1) = 4

		A	A
			A

Labels



An Example



Sorted list:

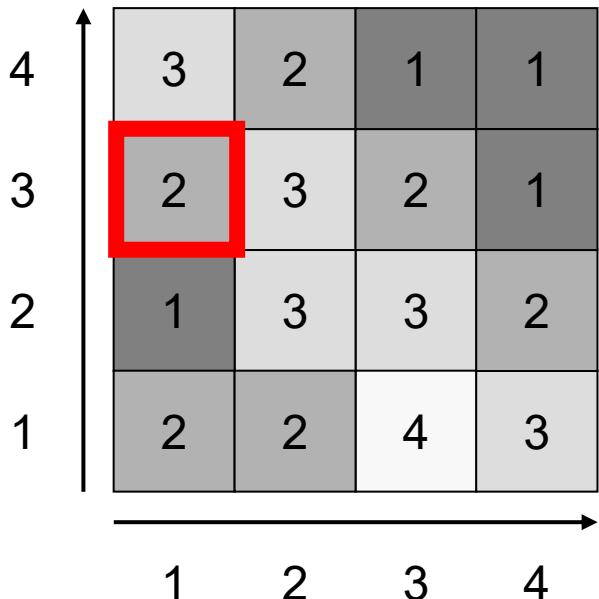
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

		A	A
			A
B			

Labels



An Example



Sorted list:

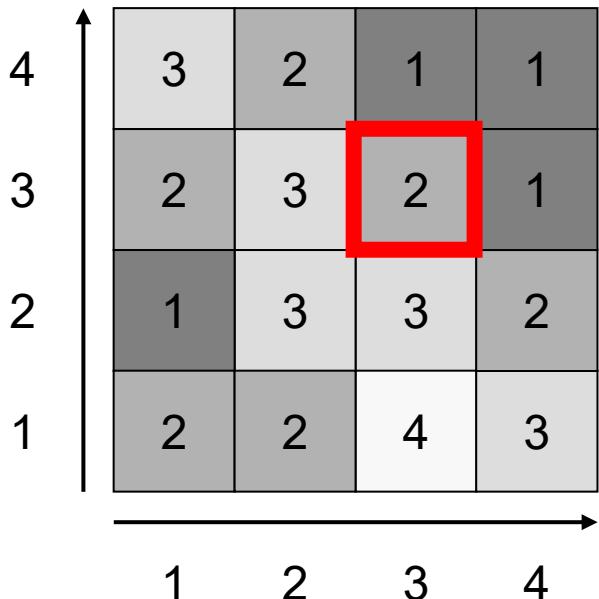
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2**
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

	A	A	A
			A
B			

Labels



An Example



Sorted list:

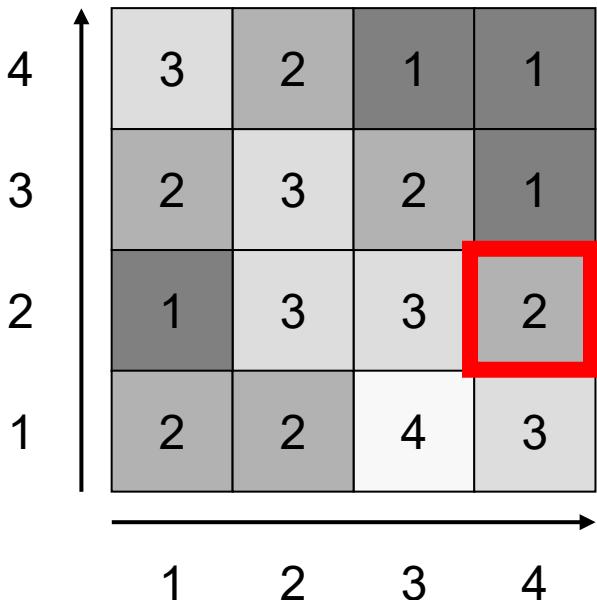
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2**
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

	A	A	A
			A
B			

Labels



An Example



Sorted list:

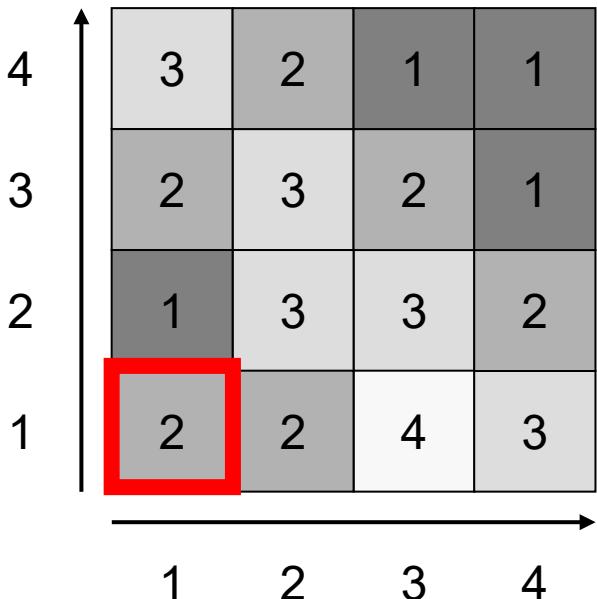
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2**
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

	A	A	A
		A	A
B			

Labels



An Example



Sorted list:

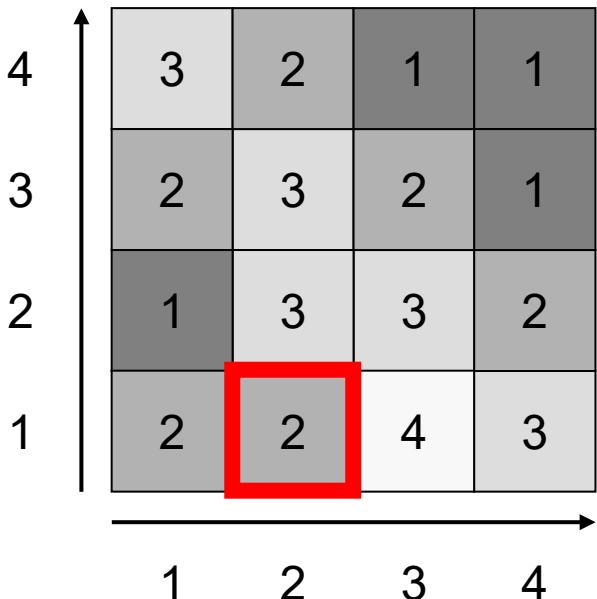
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2**
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

	A	A	A
	A	A	A
B			A

Labels



An Example



Sorted list:

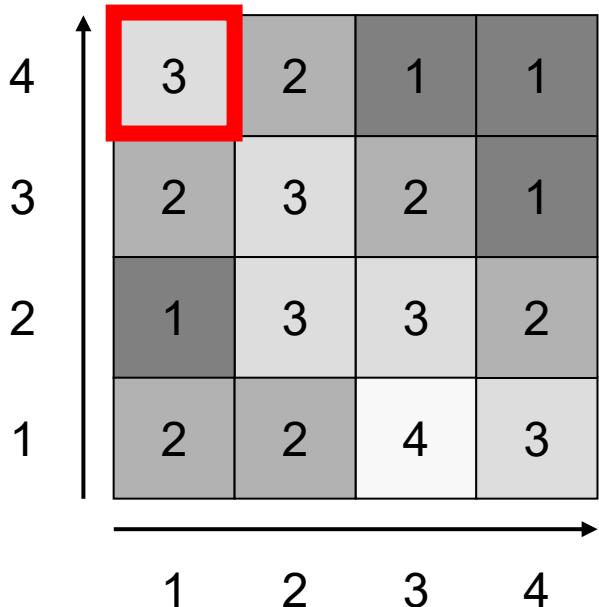
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2**
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

	A	A	A
	A	A	A
B			A
B			

Labels



An Example



Sorted list:

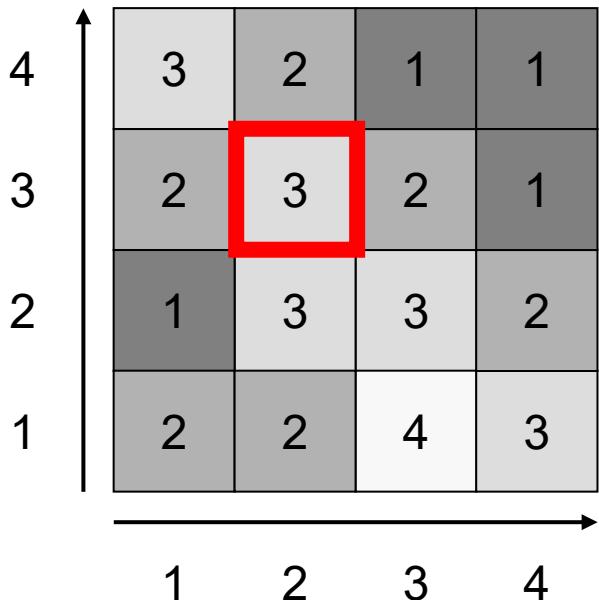
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3**
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

	A	A	A
	A	A	A
B			A
B	B		

Labels



An Example



Sorted list:

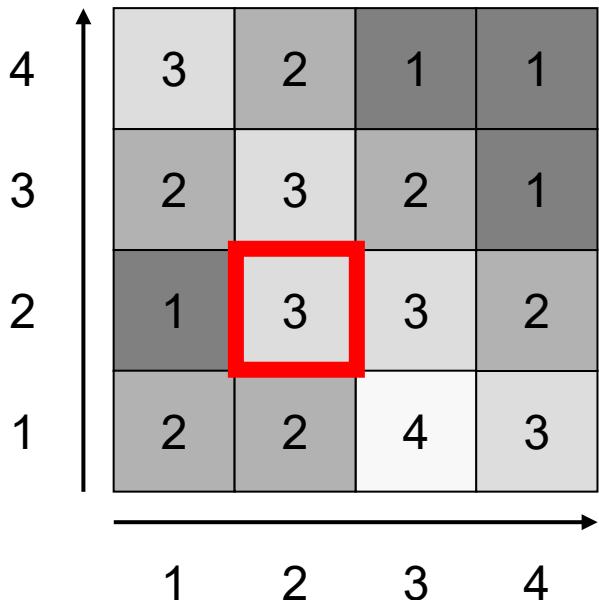
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3**
- (2,2) = 3
- (3,2) = 3
- (4,1) = 3
- (3,1) = 4

A	A	A	A
		A	A
B			A
B	B		

Labels



An Example



Sorted list:

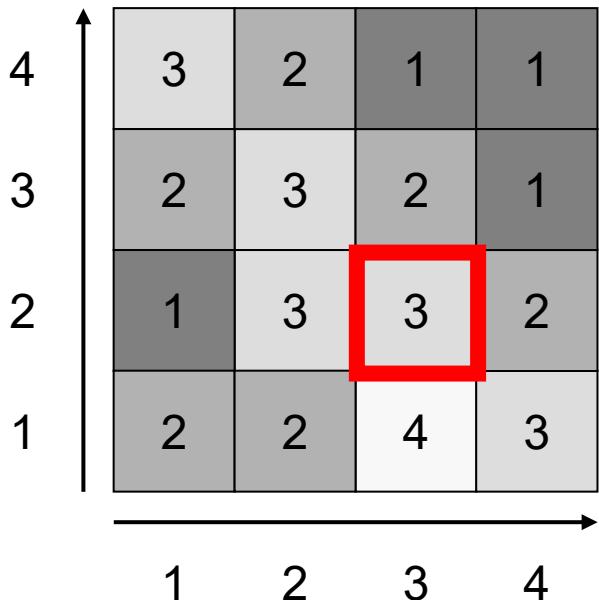
(3,4) = 1
(4,4) = 1
(4,3) = 1
(1,2) = 1
(2,4) = 2
(1,3) = 2
(3,3) = 2
(4,2) = 2
(1,1) = 2
(2,1) = 2
(1,4) = 3
(2,3) = 3
(2,2) = 3
(3,2) = 3
(4,1) = 3
(3,1) = 4

A	A	A	A
		A	A
B			A
B	B		

Labels



An Example



Sorted list:

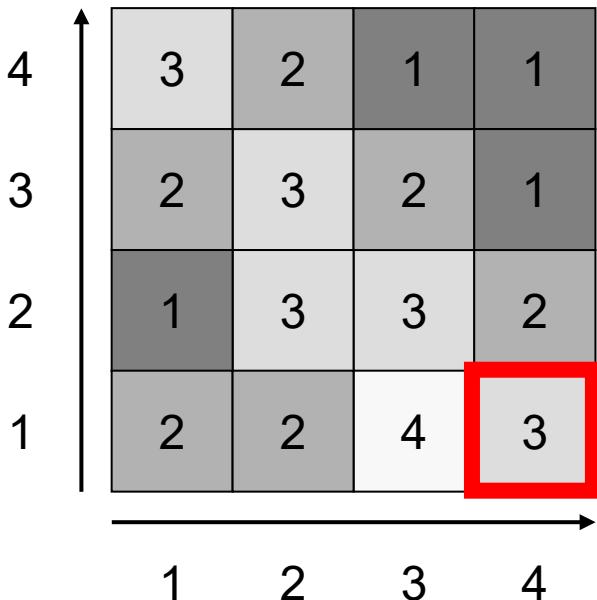
- (3,4) = 1
- (4,4) = 1
- (4,3) = 1
- (1,2) = 1
- (2,4) = 2
- (1,3) = 2
- (3,3) = 2
- (4,2) = 2
- (1,1) = 2
- (2,1) = 2
- (1,4) = 3
- (2,3) = 3
- (2,2) = 3
- (3,2) = 3**
- (4,1) = 3
- (3,1) = 4

A	A	A	A
		A	A
B			A
B	B		

Labels



An Example



Sorted list:

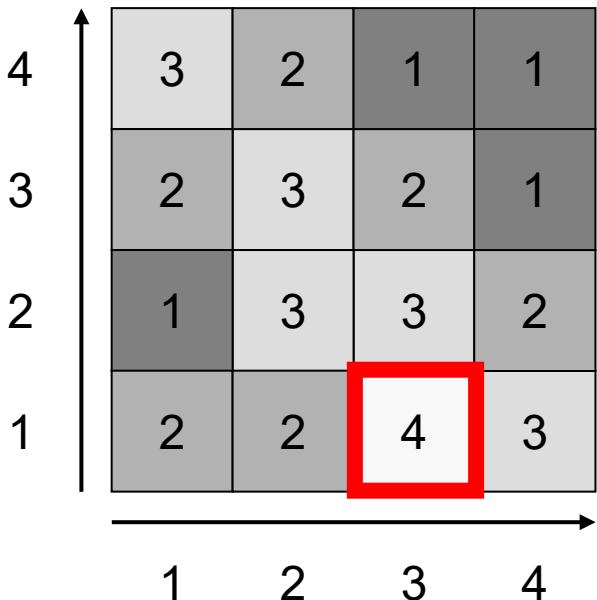
(3,4) = 1
(4,4) = 1
(4,3) = 1
(1,2) = 1
(2,4) = 2
(1,3) = 2
(3,3) = 2
(4,2) = 2
(1,1) = 2
(2,1) = 2
(1,4) = 3
(2,3) = 3
(2,2) = 3
(3,2) = 3
(4,1) = 3
(3,1) = 4

A	A	A	A
		A	A
B			A
B	B		

Labels



An Example



Sorted list:

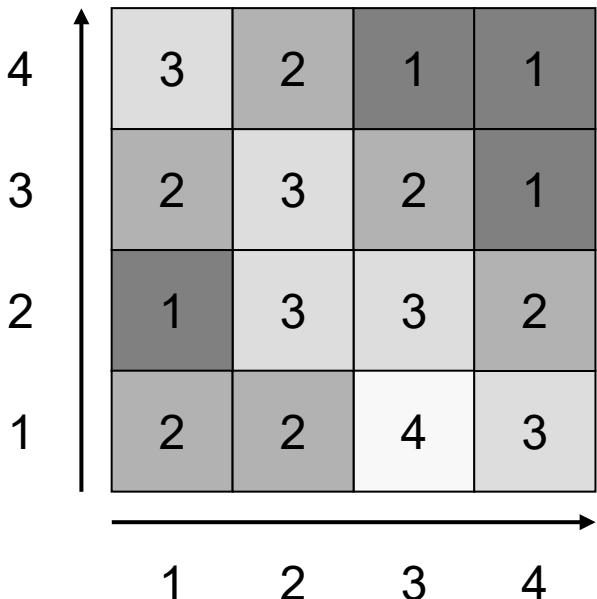
(3,4) = 1
(4,4) = 1
(4,3) = 1
(1,2) = 1
(2,4) = 2
(1,3) = 2
(3,3) = 2
(4,2) = 2
(1,1) = 2
(2,1) = 2
(1,4) = 3
(2,3) = 3
(2,2) = 3
(3,2) = 3
(4,1) = 3
(3,1) = 4

A	A	A	A
		A	A
B			A
B	B		A

Labels



An Example



Sorted list:

- $(3,4) = 1$
- $(4,4) = 1$
- $(4,3) = 1$
- $(1,2) = 1$
- $(2,4) = 2$
- $(1,3) = 2$
- $(3,3) = 2$
- $(4,2) = 2$
- $(1,1) = 2$
- $(2,1) = 2$
- $(1,4) = 3$
- $(2,3) = 3$
- $(2,2) = 3$
- $(3,2) = 3$
- $(4,1) = 3$
- $(3,1) = 4$

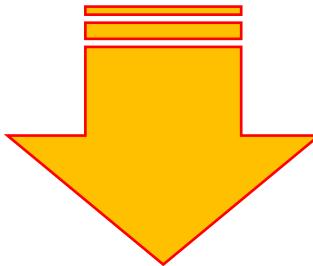
A	A	A	A
		A	A
B			A
B	B		A

Labels



Computing Watersheds

Watershed based segmentations can be very efficient



- It is possible to implement it in $O(n)$ time, where n is the number of pixels
- Since it takes $O(n)$ time to read or write an image, this is as good as it can get in most situations

To implement watersheds we need to sort the pixels

- They need to be sorted from highest to lowest gradient
- Sorting is $O(n \log(n))$, so how do we get an $O(n)$ algorithm?



Sorting in Linear Time

In any situation where we have

- A large number of values, and
- Those values are drawn from a smaller set of possibilities

We can sort in linear time with a bin sort!

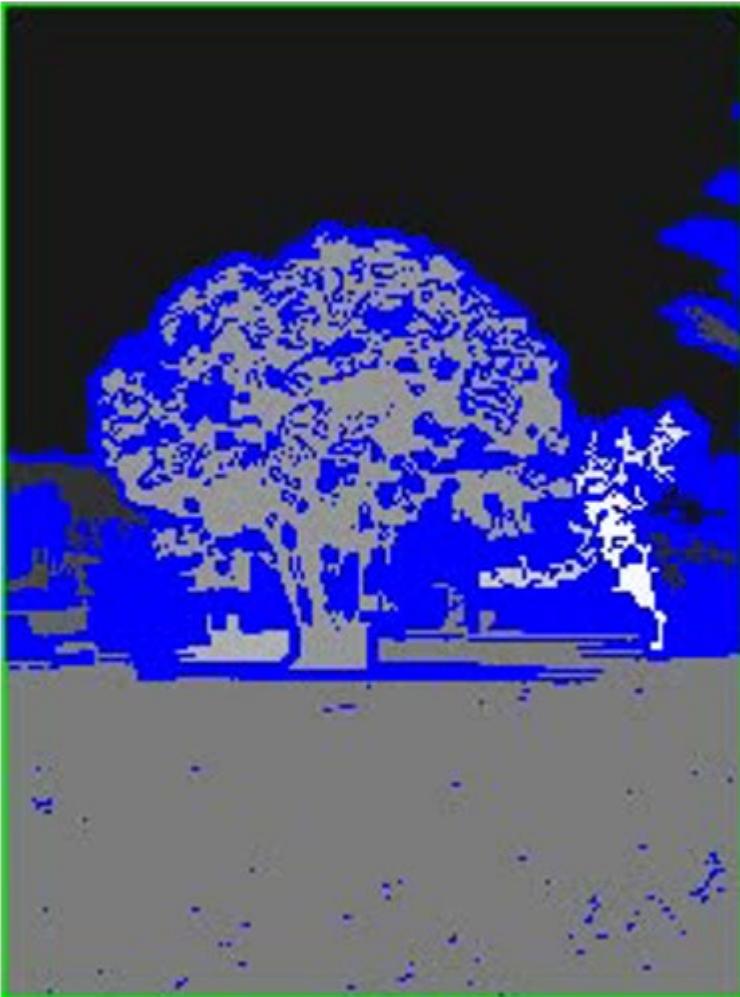
1. Make a bin (*a list, queue or stack*) for each possible value
2. For each item: put it in the appropriate bin

Bin Sort

The items are now SORTED!



Example - Watershed



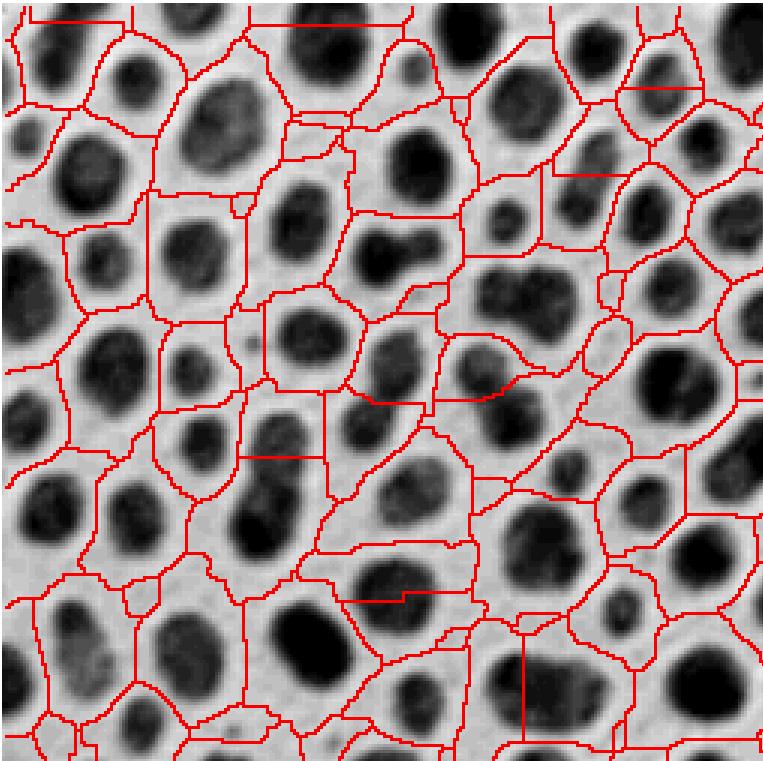
Segmenting the tree image

- The basic algorithm has been modified to avoid the effects of noise
- The gradient has been quantised to remove small variations
- Above a threshold water level, no more new segments are introduced as the water rises





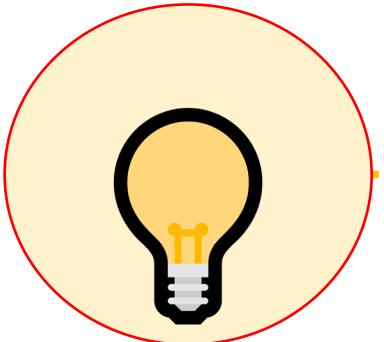
Example - Watershed



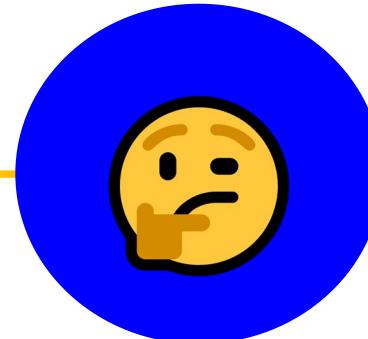
- Watersheds can also be applied to some greyscale images directly
- For example, in many medical and biological images the regions are dark or light regions against a light or dark background



Summary



1. What is Segmentation?
2. Region-based Segmentation
3. Edge-based Segmentation





University of
Nottingham

UK | CHINA | MALAYSIA

Questions



University of
Nottingham

UK | CHINA | MALAYSIA

NEXT:

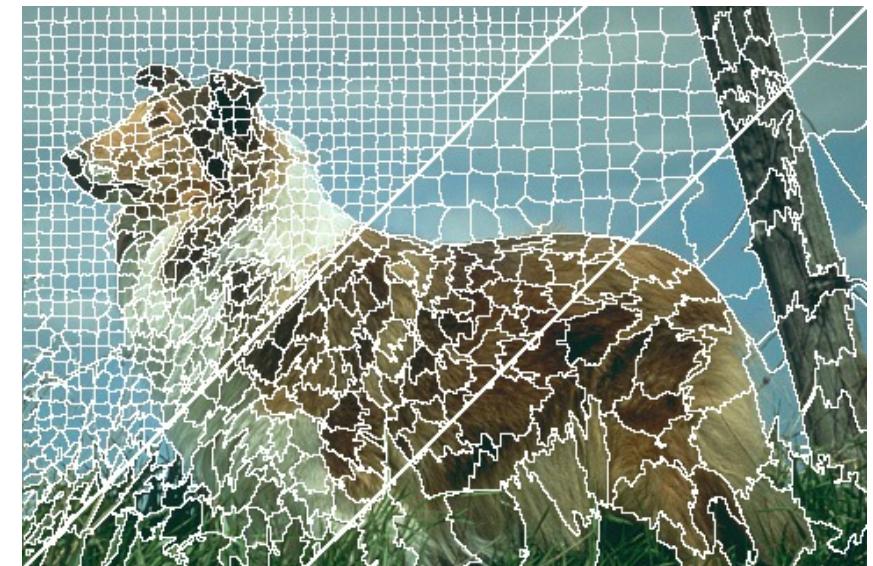
**Interactive
Segmentation**

COMP2005

Segmentation and Superpixels

An Alternative? Superpixels

- Segmentation has motivated development of some useful techniques, but:
 - ‘segmentation’ is poorly defined
 - trying to achieve meaningful, semantically correct results without knowledge of the application domain is optimistic at best
 - segmentation methods really just divide the image into similar regions
- So let’s accept that and forget the semantics.....



Simple Linear Iterative Clustering

- High-quality, compact, nearly uniform superpixels
- Simple, efficient algorithm based on K-means
- Only parameter is number of superpixels required (K)

1. Initialize cluster centers on pixel grid in steps S

- Image has N pixels, you want K superpixels
- Each superpixel is a roughly square area of roughly N/K pixels
- Each superpixel is roughly $\sqrt{N/K}$ by $\sqrt{N/K}$
- $S = \sqrt{N/K}$

2. Move centres to the position in a 3×3 window with the smallest intensity (or colour) gradient

- Move centres away from edges, onto flattest area available
- Only a small move, these are still initial positions

SLIC

3. Compare each pixel to all cluster centres within $2S$ pixels and assign it to the best matching centre

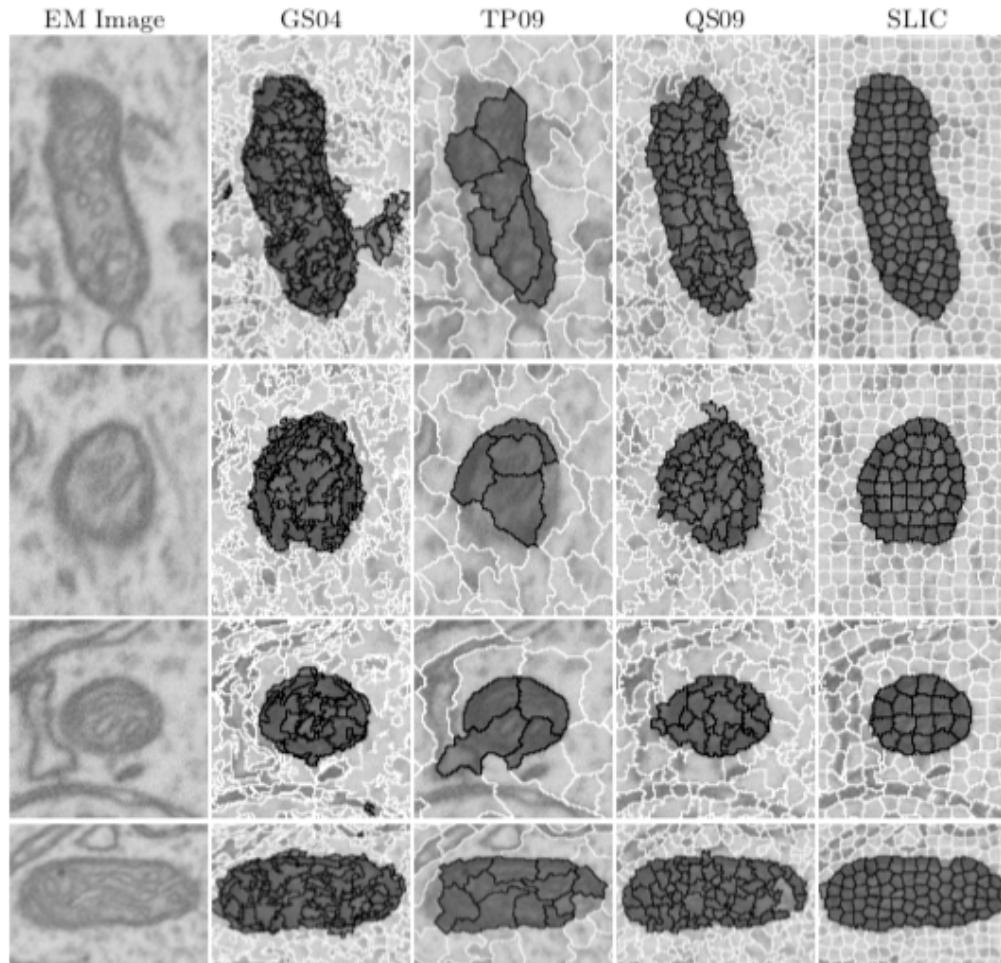
- Best matching = nearby and similar in colour
- Distance measure is sum of colour distance and image plane distance

[See the paper on Moodle for details](#)

4. Recompute cluster centres as mean colour and position of the pixels belonging to each cluster

5. Repeat 3 and 4 until total change made to position and colour of centres is below a threshold, or for a fixed number of iterations

SLIC



- Evaluated on
 - similarity of pixels in superpixels vs variation of values between adjacent superpixels
 - proportion of object boundaries marked by a superpixel boundary
- E.g. EM images of brain mitochondria (linked to degenerative diseases)
- Segmentation meets edge detection?

SLIC in Matlab

```
A = imread('kobi.png');  
[L,N] = superpixels(A,500);  
// L is a label image  
// N number of superpixels actually produced  
  
figure  
BW = boundarymask(L);  
// marks transitions from one label to another  
  
imshow(imoverlay(A,BW,'cyan'),  
      'InitialMagnification',67);  
//overlays boundary mask on original image  
in cyan
```

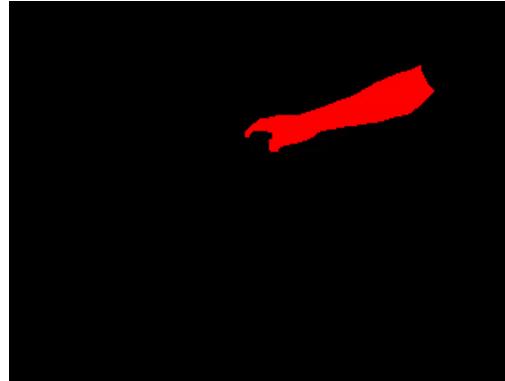
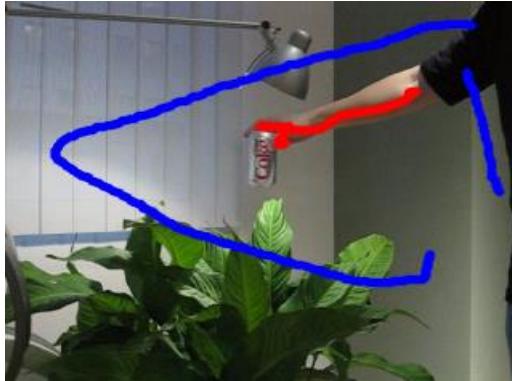


<https://www.mathworks.com/help/images/ref/superpixels.html>

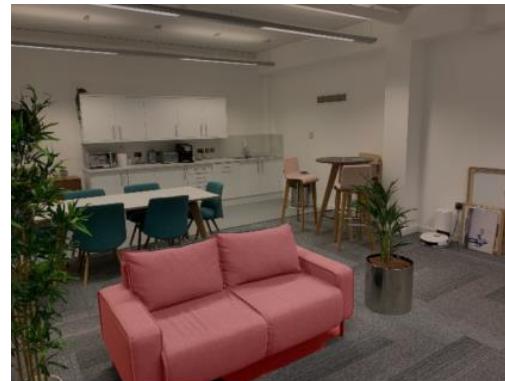
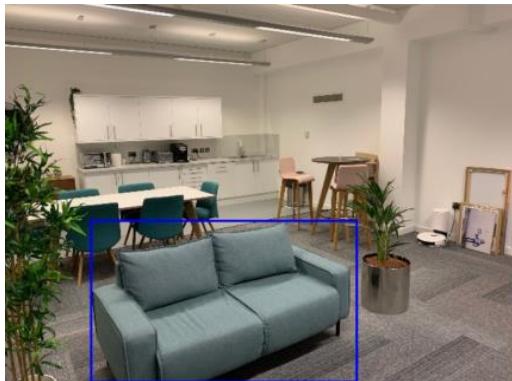
COMP2005

Interactive Segmentation: Graph Cut

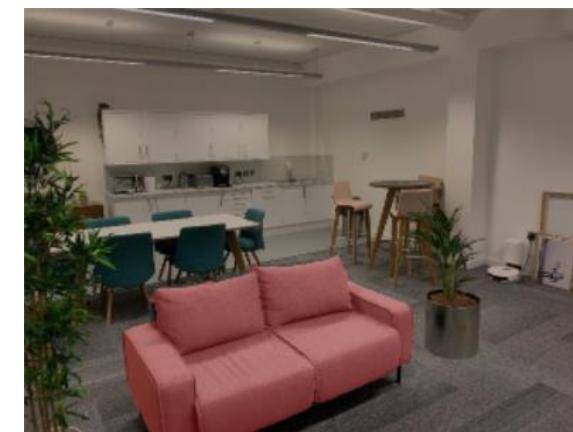
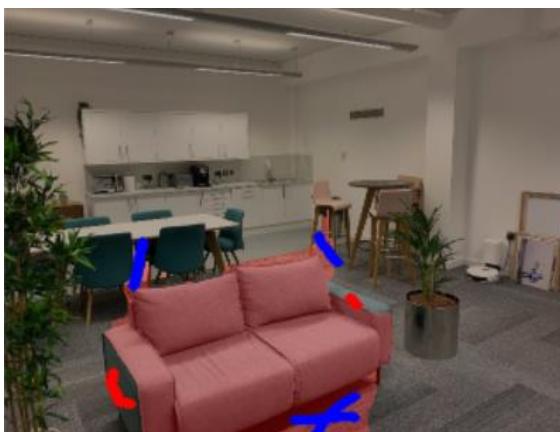
What is Interactive Segmentation?



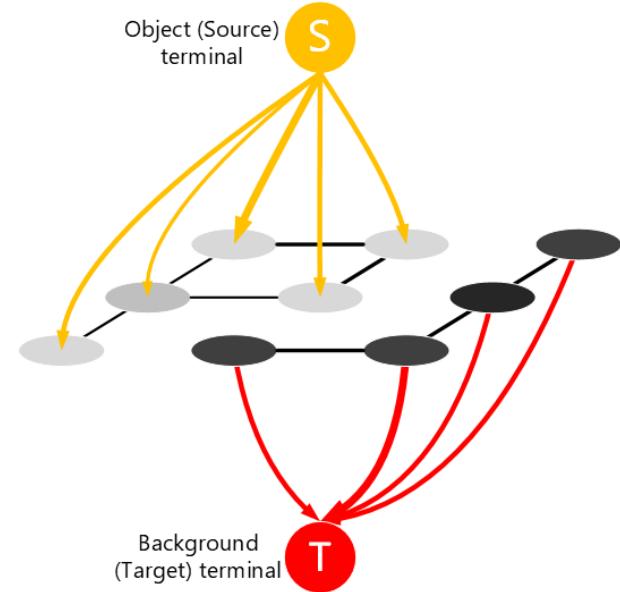
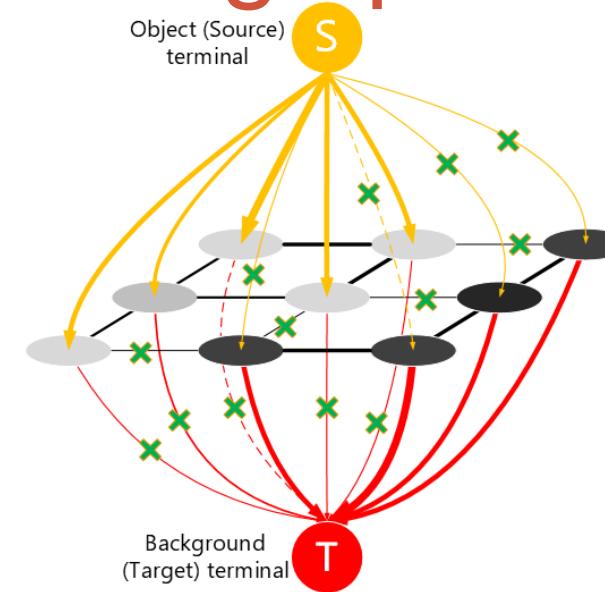
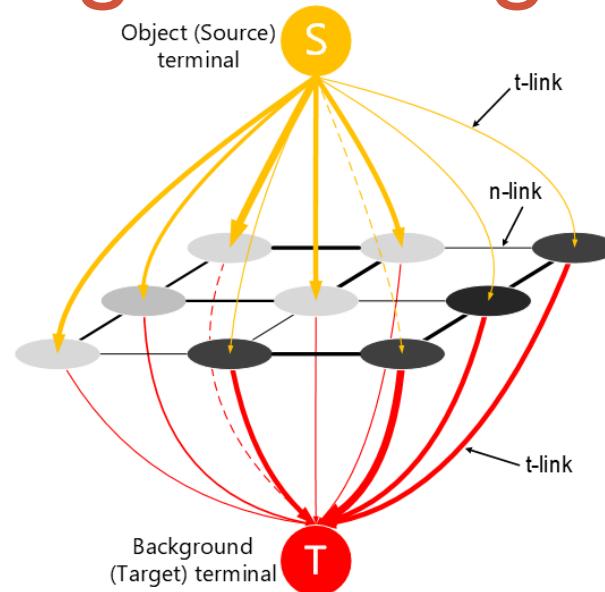
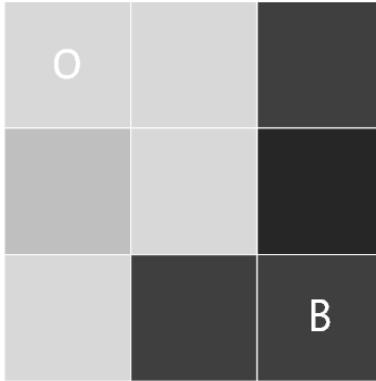
- The user select a rough area in the image to indicate the content she/he would like to segment
- The algorithm segments the area of the user's interest
- We will focus on Graph Cut in this module



Graph Cut: Overview



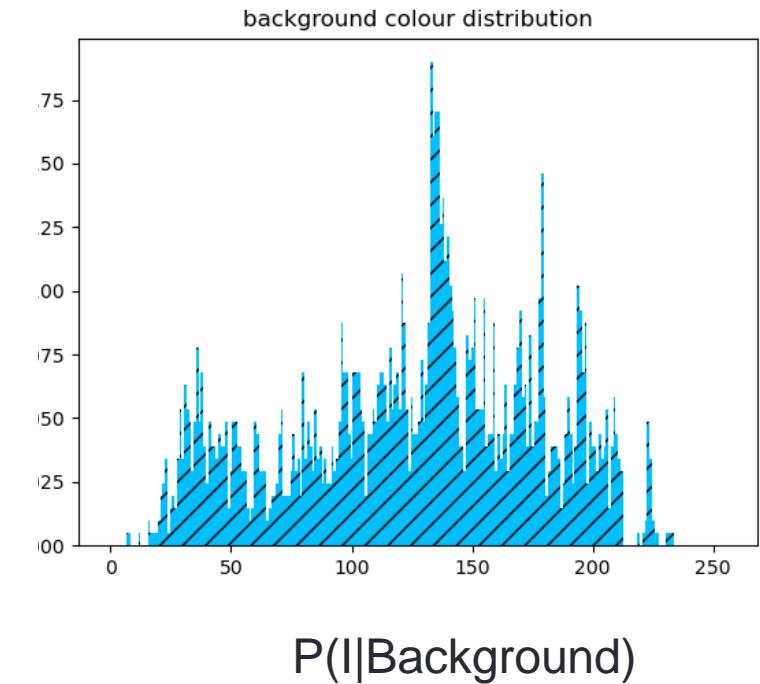
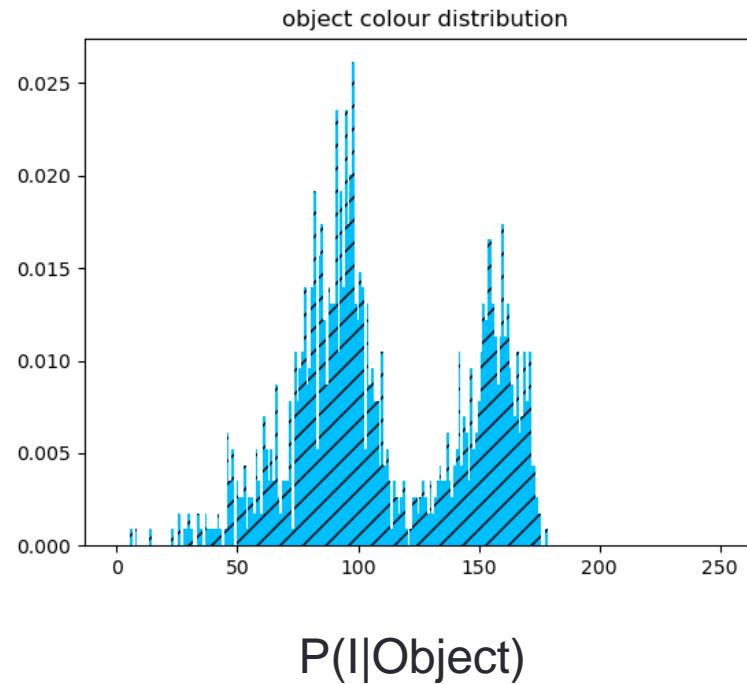
Organizing the image into a graph



- two kinds of links: **t-link** (links between pixels and Source or Target) and **n-link** (links between adjacent pixels)
- thickness of the link indicates its weight - the higher the weight is, the thicker the link is
- t-links are directed, while **n-links** are undirected
- make an n-link thick if values of the two pixels connected by it are closed
- make a t-link from Source to a pixel thick if the pixel likely belongs to the object
- make a t-link from a pixel to Target thick if the pixel likely belongs to the background
- remove (cut) thin links (including both t-links and n-links) to divide the graph into two independent parts with each has a terminal inside
- the summation of weights of removed links should be minimum, also known as **min-cut problem**

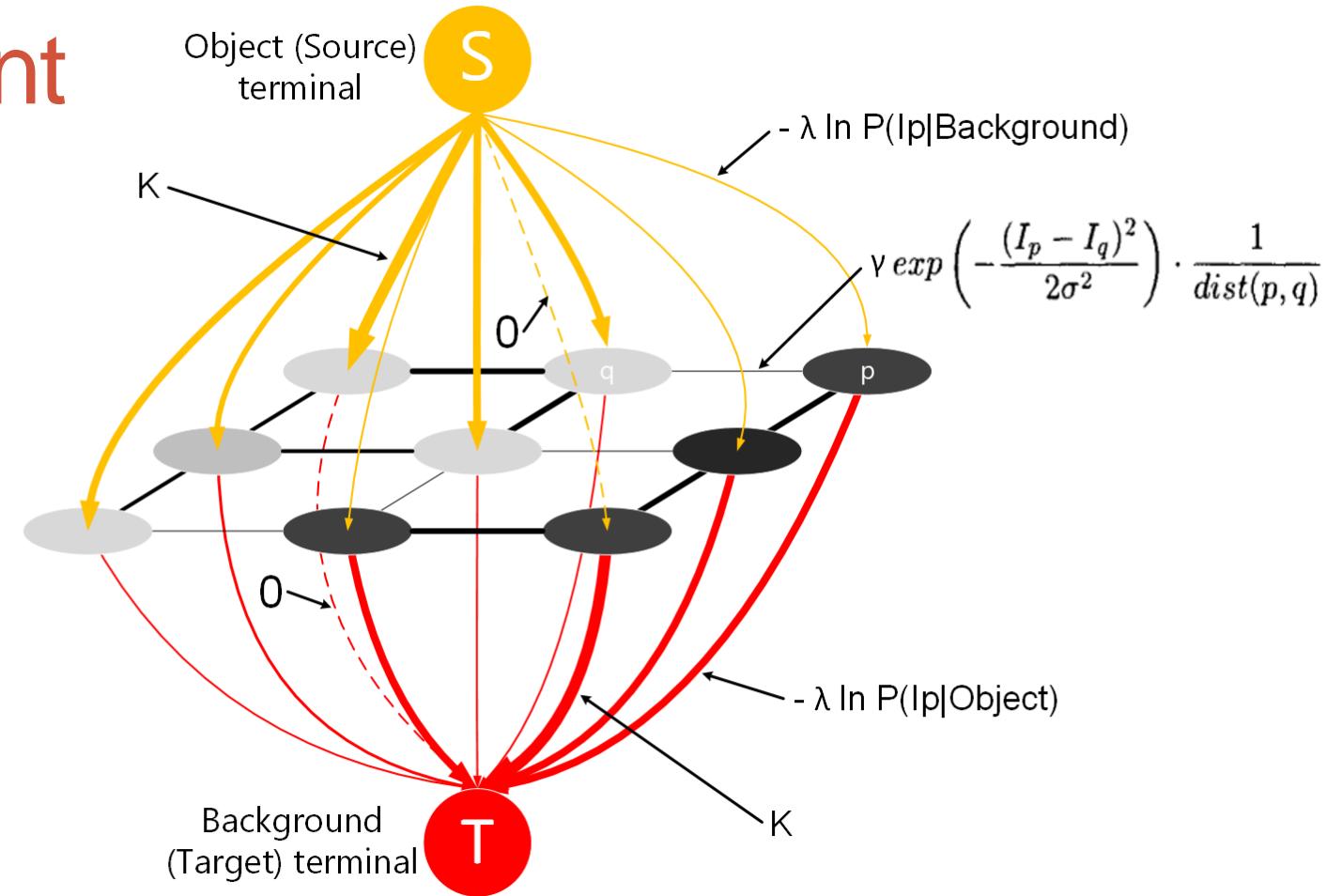
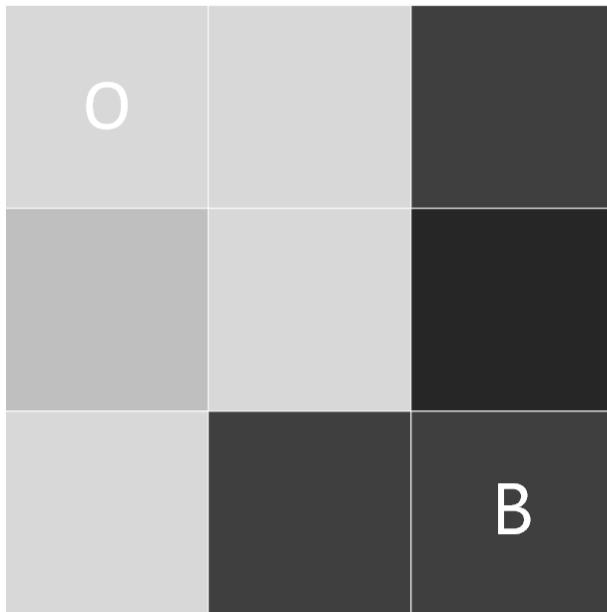
*how to assign weights to links?
how to solve the min-cut problem?*

Preliminary: colour distribution



calculate colour distributions of pixels covered by red curves (foreground) and blue curves (background) separately

Weight assignment

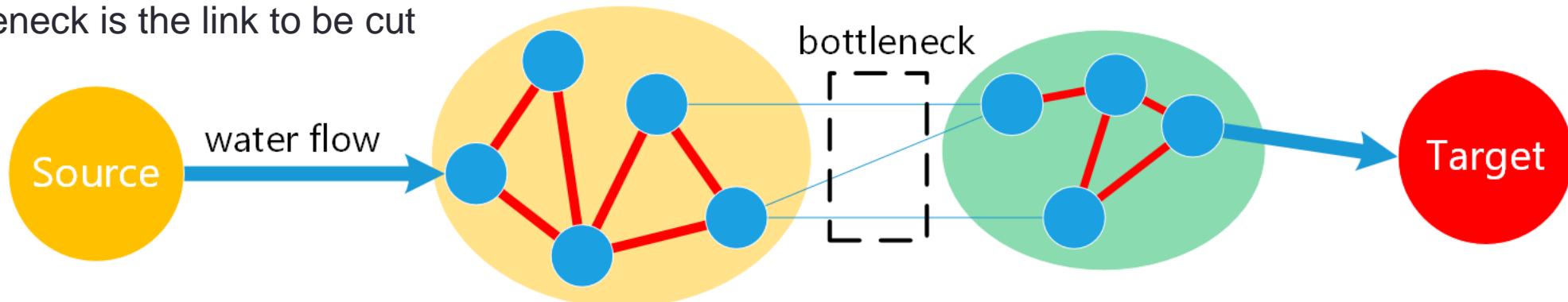


- λ and γ are hyper-parameters
- K should be a large value, can be determined by the formula below

$$K = 1 + \max_{p \in \{\text{pixels in the image}\}} \sum_{q \in \{p's \text{ neighbours}\}} \gamma \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) \cdot \frac{1}{dist(p, q)}$$

Solving min-cut problem

- **min-cut problem** is equivalent to **max-flow problem**
- imagine each link as a tube with different capacities - larger weights, larger capacities
- water is pumped from the Source to the Target, try to maximize the water flow to find the bottleneck
- bottleneck is the link to be cut



An efficient algorithm for solving max-flow problem:

Boykov, Yuri, and Vladimir Kolmogorov. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE transactions on pattern analysis and machine intelligence* 26.9 (2004): 1124-1137.

For further details about Graph Cut, read this paper:

Boykov, Yuri Y., and M-P. Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 1. IEEE, 2001.

COMP2005

Introduction to Image Compression

Image Compression

- Individual images can be large
- Images are easy to acquire, collections increase rapidly
- In some applications, images are gathered automatically
- Luckily, image data is redundant in several ways
 - Coding redundancy
 - Spatial redundancy
 - Psychovisual redundancy



Coding Redundancy

- The grey level histogram of an image gives the probability (frequency) of occurrence of grey level r_k

$$p(r_k) = \frac{n_k}{n} \quad k = 0, 1, 2, \dots, L-1$$

- If the number of bits used to represent each value of r_k is $l(r_k)$, the **average number of bits required to represent a pixel is**

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p(r_k)$$

- To code an $M \times N$ image requires MNL_{avg} bits

Coding Redundancy

- If an m -bit natural binary code is used to represent grey level then
 - all pixels take the same amount of space,
 - $P(r_k)$ values sum to 1, so

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p(r_k) = \sum_{k=0}^{L-1} m p(r_k) = m$$

- and an image occupies MNm bits
- But some pixel values are more common than others.....

Variable Length Encoding

- Assigning fewer bits to the more probable grey levels than to less probable ones can achieve data compression, e.g:

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

- Build a codebook, replace ‘true’ pixel values with code
- Lossless: the process can be reversed by inverting the codebook

Spatial Redundancy

- Sometimes called *Interpixel Redundancy*
- Neighbouring pixels often have similar values
- Compression based on spatial redundancy involves some element of pixel grouping, or transformation
- Simplest is Run-Length Encoding
 - maps the pixels along each scan line into a sequence of pairs $(g_1, r_1), (g_2, r_2), \dots,$
 - where g_i is the i th grey level, r_i is the run length of i th run

A Binary Example

Row 1: (0, 16)

Row 2: (0, 16)

Row 3: (0, 7) (1, 2) (0, 7) encode

Row 4: $(0, 4), (1, 8) (0, 4)$ ←

Row 5: (0, 3) (1, 2) (0, 6) (1, 3) (0, 2)

Row 6: (0,2) (1, 2) (0,8) (1, 2) (0, 2)

Row 7: (0, 2) (1,1) (0, 10) (1,1) (0, 2)

Row 8: (1, 3) (0, 10) (1,3)

Row 9: (1, 3) (0, 10) (1, 3)

Row 10: (0,2) (1, 1) (0,10) (1, 1) (0, 2)

Row 11: (0, 2) (1, 2) (0, 8) (1, 2) (0, 2)

Row 12: (0, 3) (1, 2) (0, 6) (1, 3) (0, 2)

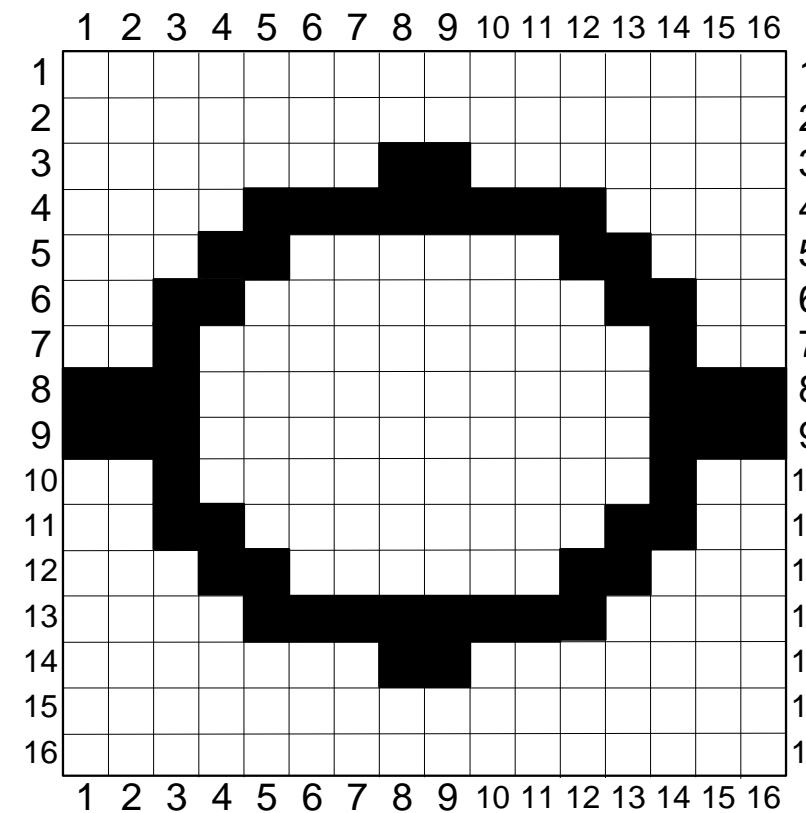
Row 13: (0, 4) (1,8) (0, 4)

Row 14: (0, 7) (1, 2) (0, 7)

Row 15: (0, 16) deco

Row 16: (0, 16) _____

decode



Psychovisual Redundancy

- Some grey level and colour differences are imperceptible; goal is to compress without noticeable change to the image

256 gray levels



16 gray levels



16 gray levels



A simple method:
add a small
random number to
each pixel before
quantization

Evaluating Compression

- Fidelity Criteria: success is judged by comparing original and compressed versions
- Some measures are objective, e.g. root mean square error (e_{rms}) and signal to noise ratio (SNR)
- Let $f(x,y)$ be the input image, $f'(x,y)$ be reconstructed input image from compressed bit stream, then

$$e_{rms} = \left(\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (f'(x, y) - f(x, y))^2 \right)^{1/2}$$

$$SNR = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (f'(x, y))^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (f'(x, y) - f(x, y))^2}$$

Fidelity Criteria



$$e_{rms} = 6.93$$

$$e_{rms} = 6.78$$

$$SNR_{rm} = 10.25 \quad SNR_{rm} = 10.39$$

Fidelity Criteria

- e_{rms} and SNR are convenient objective measures
- Most decompressed images are viewed by human beings
- Subjective evaluation of compressed image quality by human observers are often more appropriate

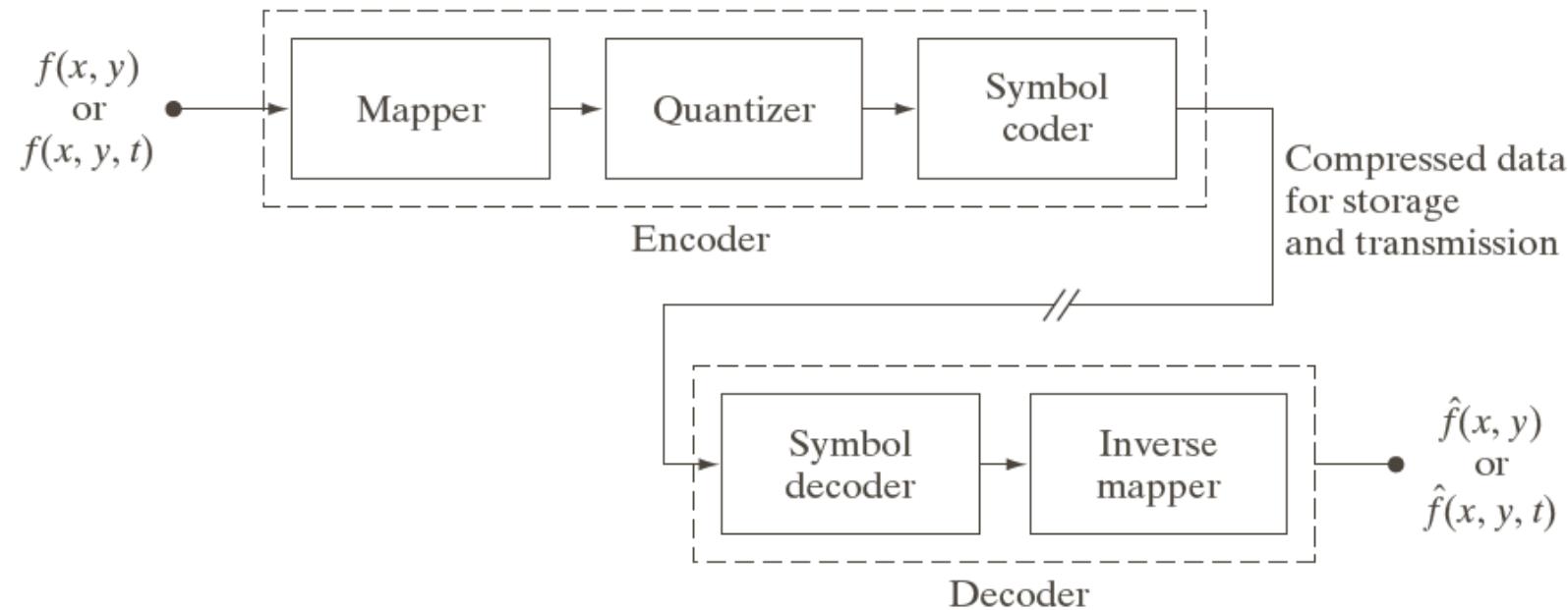
Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Rating scale of
the Television
Allocations Study
Organization.
(Frendendall and
Behrend.)

Image Compression Systems

- Mapper (spatial redundancy)
 - transforms input data in a way that facilitates reduction of interpixel redundancies
 - reversible
- Quantiser (psychvisual redundancy)
 - transforms input data in a way that facilitates reduction of psychovisual redundancies
 - **not reversible**
- Symbol coder (coding redundancy)
 - assigns the shortest code to the most frequently occurring output values
 - reversible

Image Compression Systems



Functional block diagram of a general image compression system

Image Compression: Huffman Coding

Exploiting Coding Redundancy

- These methods are derived from information theory:
try to maintain a high level of information in compressed images
- Not limited to images, are applicable to any digital information.
 - speak of **symbols instead of pixel values** and **sources instead of images**
 - exploit nonuniform probabilities of symbols (nonuniform histograms) and use a variable-length code.
- Evaluation requires a measure of the information content of a source: *Entropy*

Entropy

- The idea: associate information with probability
- A random event E with probability $P(E)$ contains:

$$I(E) = \log\left(\frac{1}{P(E)}\right) = -\log(P(E))$$

units of information

- Suppose that grey level values are generated by a random variable, then r_k contains:

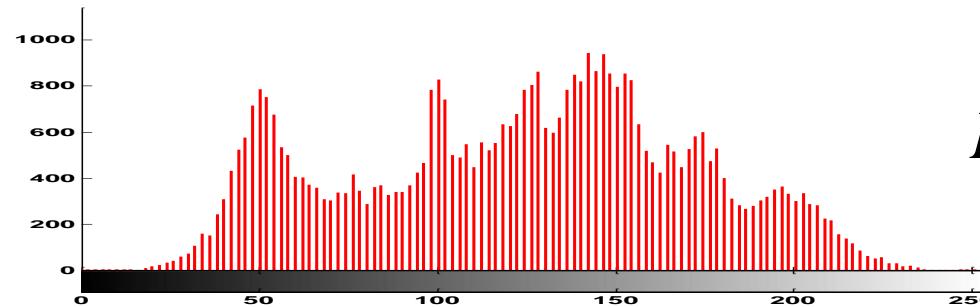
$$I(r_k) = -\log(P(r_k))$$

Note: $I(E)=0$ when $P(E)=1$

units of information

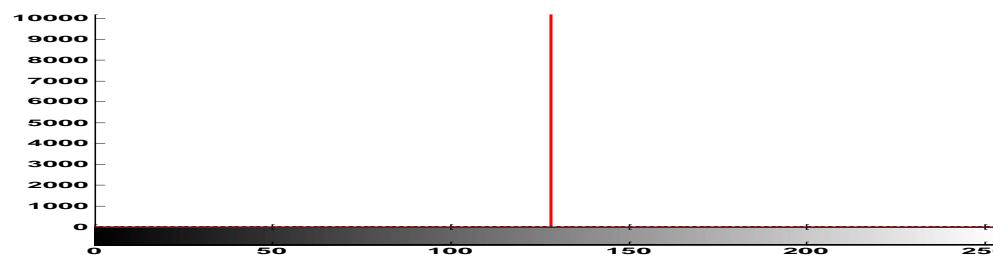
Entropy

- Entropy is the average information content of an image, a measure of histogram dispersion



$$H = -\sum_{k=0}^{L-1} p(r_k) \log_2 p(r_k)$$

entropy=7.4635



entropy=0

- Can't compress to less than H bits/pixel without losing information

Huffman Coding

- Compute probabilities of each symbol by histogramming the source
- Process probabilities to pre-compute codebook:
 $\text{code}(i)$
 - codebook is static (fixed)
- Encode source symbol-by-symbol:
 $\text{symbol}(i) \rightarrow \text{code}(i)$
- Transmit coded signal and codebook
- The need to pre-process (histogram) the source before encoding begins is a disadvantage

Huffman Coding

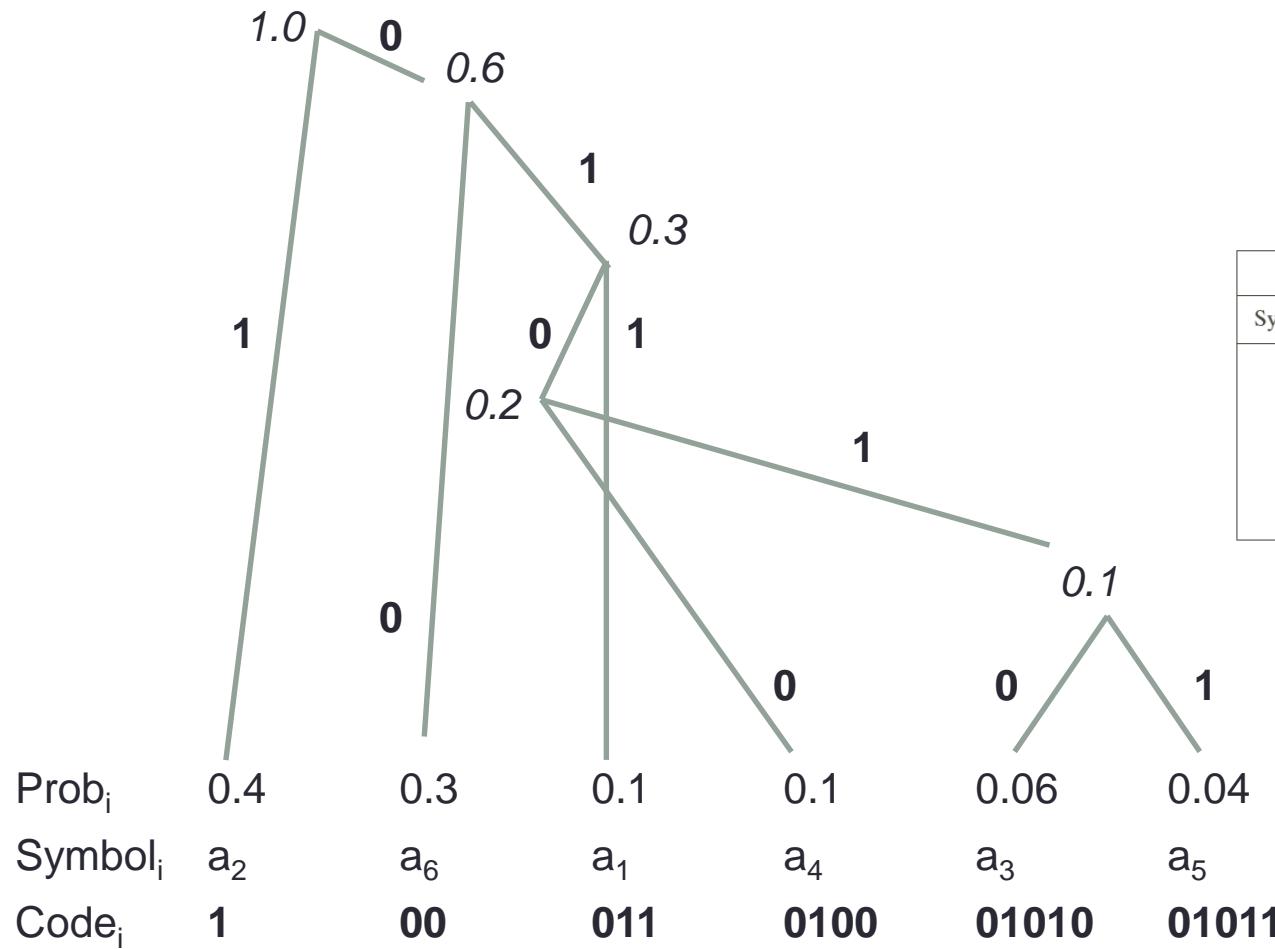
- Builds a binary tree in which symbols to be coded are nodes
- The algorithm:
 - Create a list of nodes, one per symbol, sorted in order of symbol frequency (or probability)
 - REPEAT (until only one node left)
 - Pick the two nodes with the lowest frequencies/probabilities and create a parent of them
 - **Randomly** assign the codes 0,1 to the two new branches of the tree and delete the children from the list
 - Assign the sum of the children's probabilities to their parent and insert it in the list
 - Path from root to node gives code for corresponding symbol

Huffman Coding

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

- Create parent node of a_3 and a_5
- Create parent of new node and a_4
- New parent is higher up list (0.2) than a_1 (0.1)

Huffman Coding



Huffman Coding

- The algorithm systematically places nodes representing high probability symbols further up the tree: their paths (and so codes) are shorter
- No code is a prefix to any other – don't need to mark boundaries between codes
 - e.g. 01101010 must be a_1a_3
- In this example
 - Average length of the code is 2.2.bits/symbol
 - The entropy of the source is 2.14 bits/symbol
- In general
 - Break image into small (e.g. 8 x 8) blocks
 - Each block is a symbol to be encoded

A Huffman Code Example

From a past exam paper:

An image has the following normalized histogram. Derive a Huffman code for each pixel value, showing how you obtained your code

10 mins

Pixel Value	Normalised Frequency
0	0.1
1	0.1
2	0.15
3	0.35
4	0.2
5	0
6	0.05
7	0.05

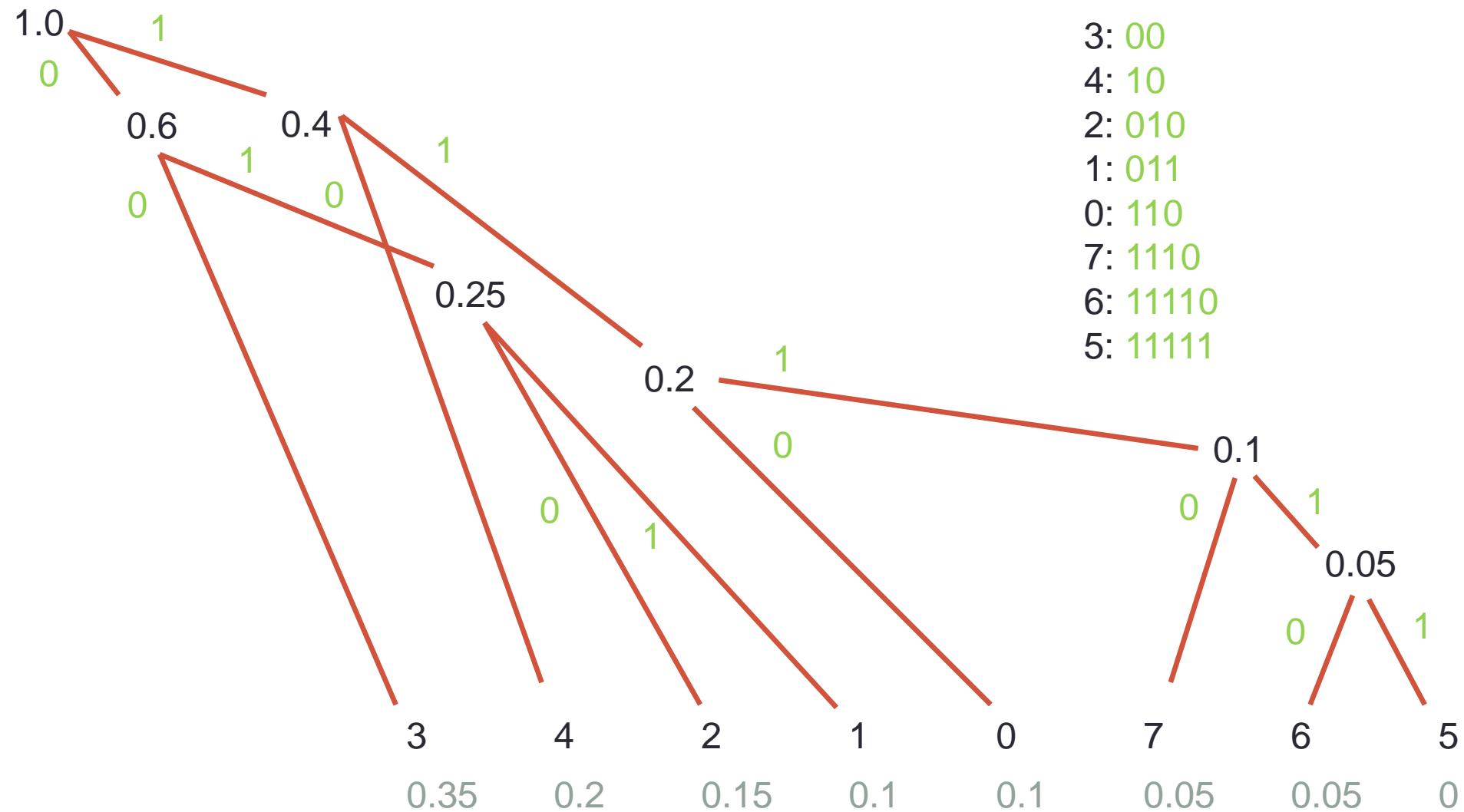
A Huffman Code Example

Pixel Value	Normalised Frequency
0	0.1
1	0.1
2	0.15
3	0.35
4	0.2
5	0
6	0.05
7	0.05

Sort
→

Pixel Value	Normalised Frequency
3	0.35
4	0.2
2	0.15
1	0.1
0	0.1
7	0.05
6	0.05
5	0

A Huffman Code Example: Tree View



A Huffman Code Example: The Code

		Code length	Normalised Freq,
3: 00	0: 110	3	0.1
4: 10	1: 011	3	0.1
2: 010	2: 010	3	0.15
1: 011	3: 00	2	0.35
0: 110	4: 10	2	0.2
7: 1110	5: 11111	5	0
6: 11110	6: 11110	5	0.05
5: 11111	7: 1110	4	0.05

Mean bits/pixel $L_{avg} = 0.3 + 0.3 + 0.45 + 0.70 + 0.4 + 0 + 0.25 + 0.2 = 2.6$

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p(r_k)$$

Without compression $L_{avg} = 3$

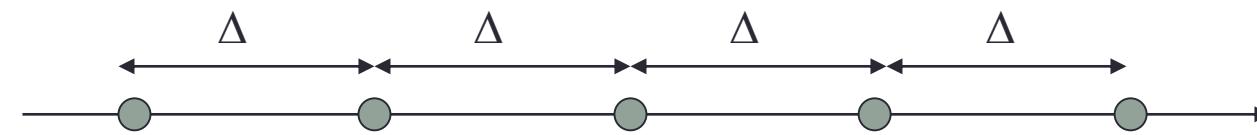
Compression ratio = $2.6/3 = 0.86$

Image Compression: GIF

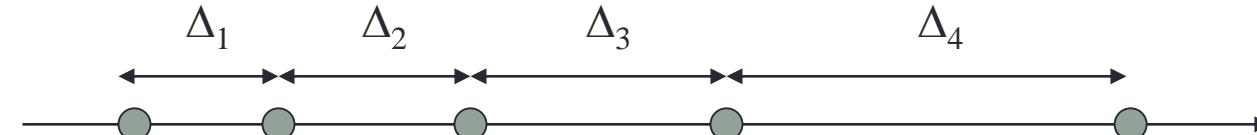
Using Psychovisual Redundancy

- Quantisation is widely used
 - Represent areas of grey level/colour space with fewer bits
 - Lossy: cannot be inverted
 - Find the best tradeoff between maximal compression \leftrightarrow minimal distortion
- Scalar Quantisation (i.e. quantising scalar values)

Uniform scalar quantization:



Non-uniform scalar quantization:



Vector Quantisation

- Palettised images (gif)
 - Map vector values (R,G,B) onto scalar values
 - Multiple vectors map to each scalar



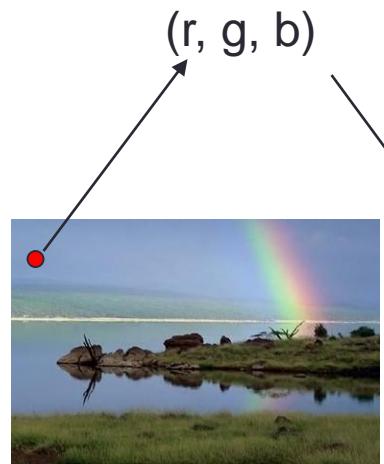
→
Vector
quantisation



True colour R,G,B
8 bits each
1677216 possible colours

gif
8 bits per pixel
256 possible colours

Paletised Images



For each pixel in
the original image

Find the closest
colour in the
Colour Table

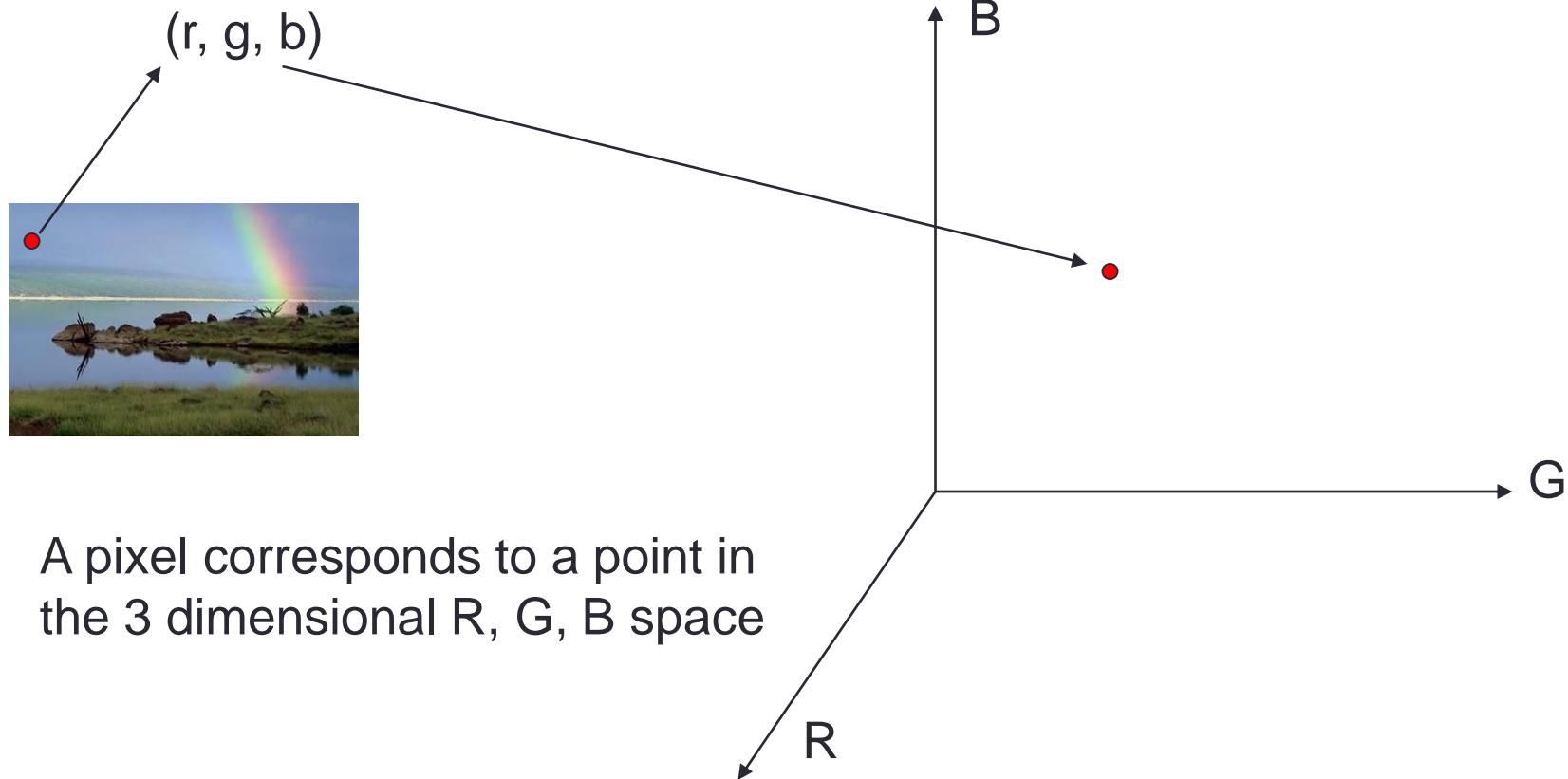
r_0	g_0	b_0
r_1	g_1	b_1
⋮		
r_{255}	g_{255}	b_{255}
Colour Table		



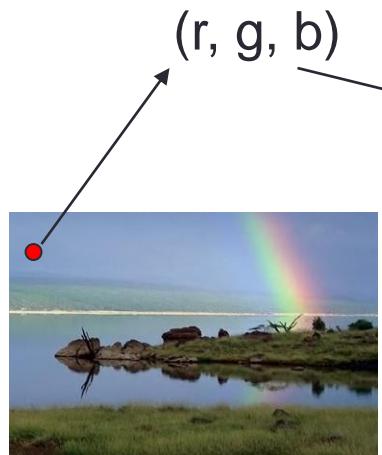
Record the index of
that colour (for
storage or
transmission)

To reconstruct the
image, place the
indexed colour from
the Colour Table at
the corresponding
spatial location

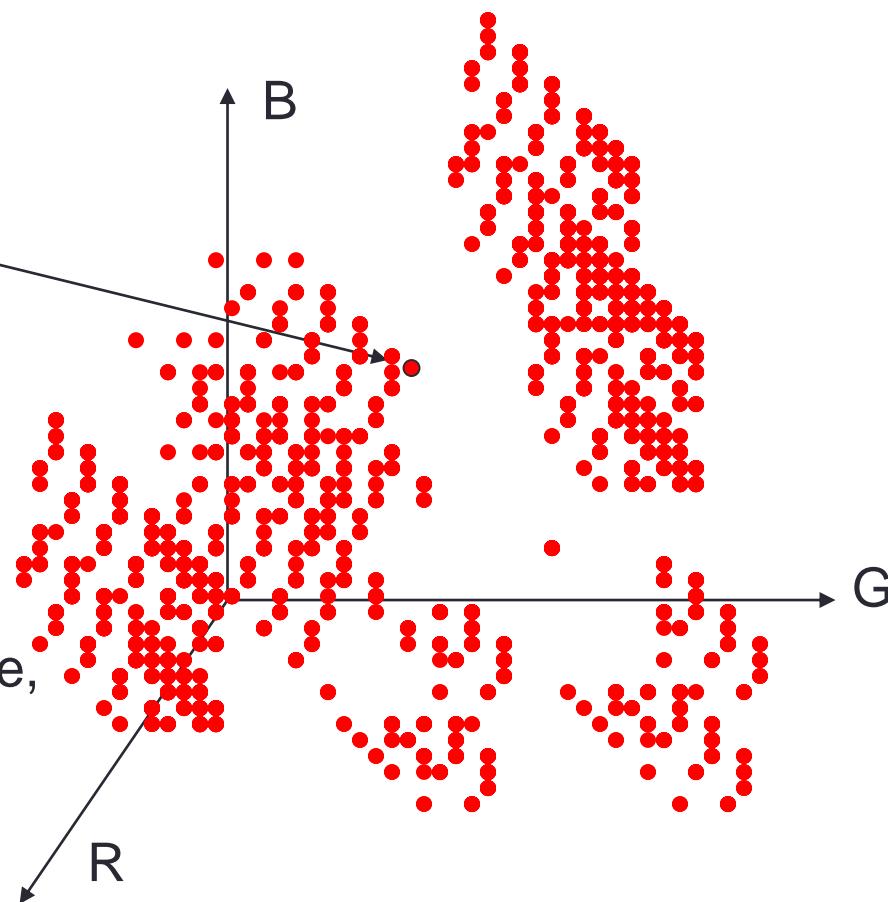
Building a Palette



Building a Palette

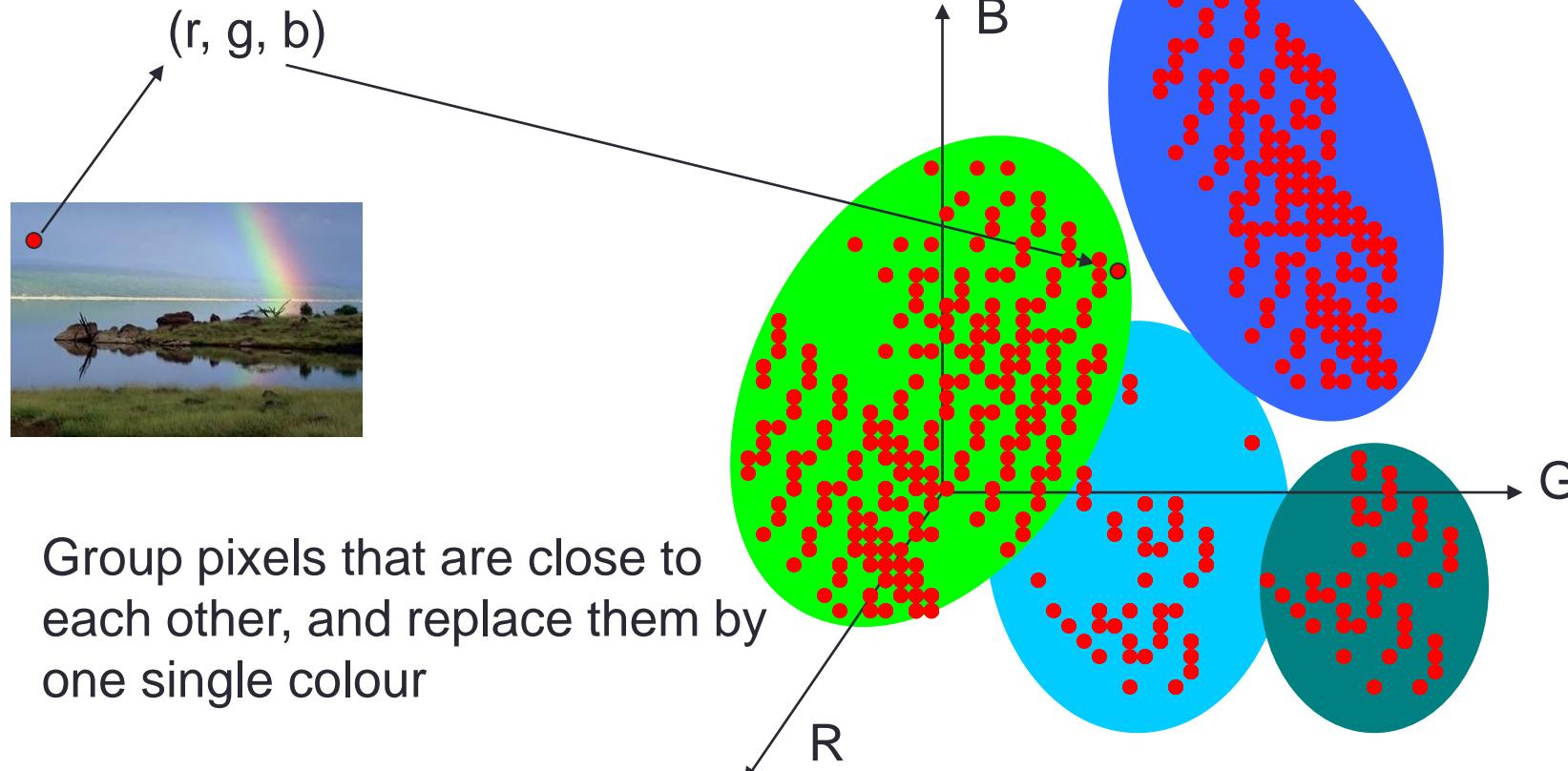


(r, g, b)



Map all pixels into R,G,B space,
Clouds of pixels are formed

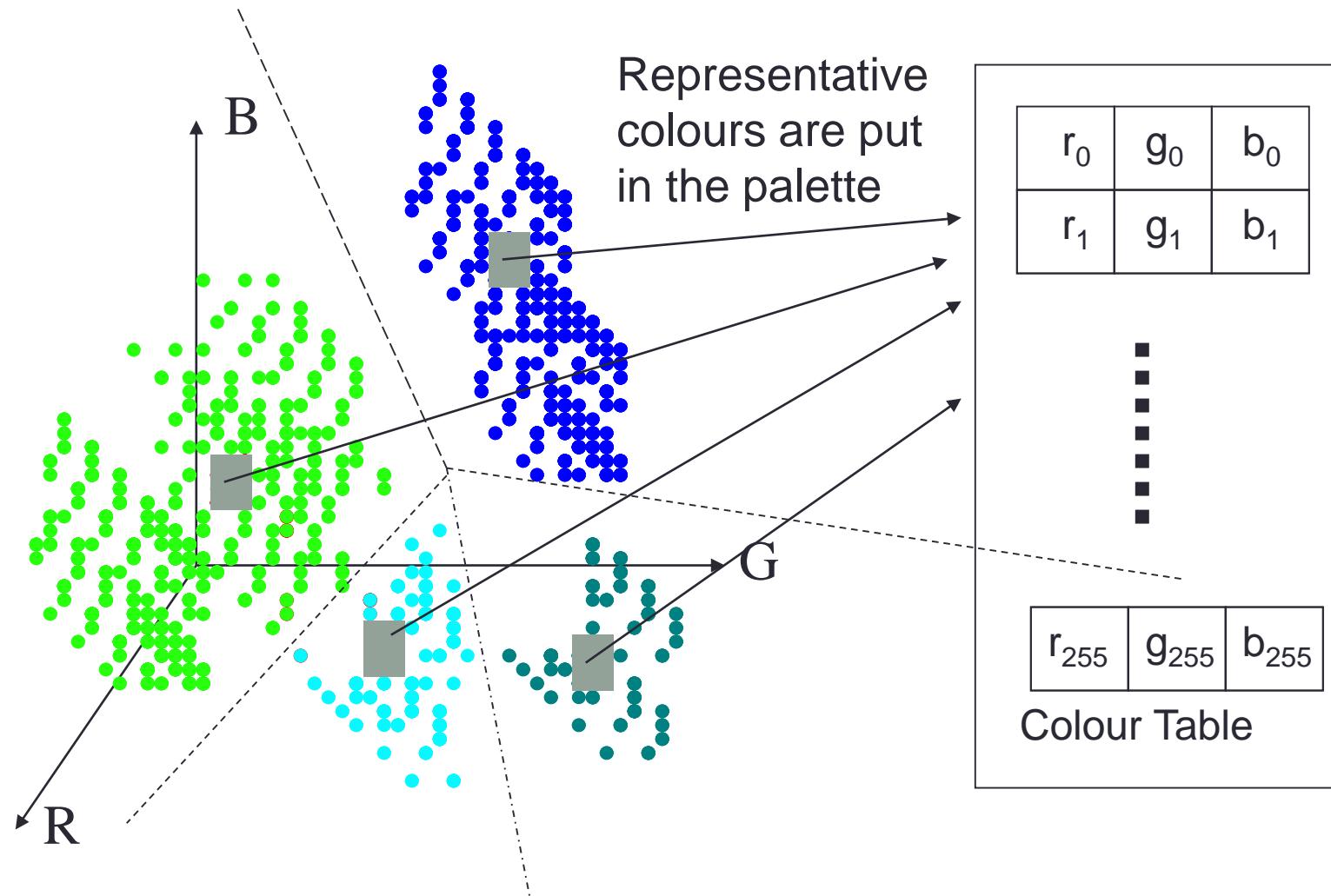
Building a Palette



Group pixels that are close to each other, and replace them by one single colour

Close to each other means of a similar colour

Building a Palette



Building a Palette

- Many clustering algorithms exist
 - supervised
 - unsupervised
- We know how many clusters we need: one per palette entry
- We need clusters that are spread across the colour space
- A supervised method....

- K-Means Clustering

- Start with estimates of the mean of each cluster $\mu_1, \mu_2, \dots, \mu_k$
- Assign each point, p , to the cluster where $|p - \mu_i|$ is smallest
- Recompute the means
- Repeat until no changes are made to the clusters



COMP2005

Image Compression: JPEG



Exploiting Spatial Redundancy

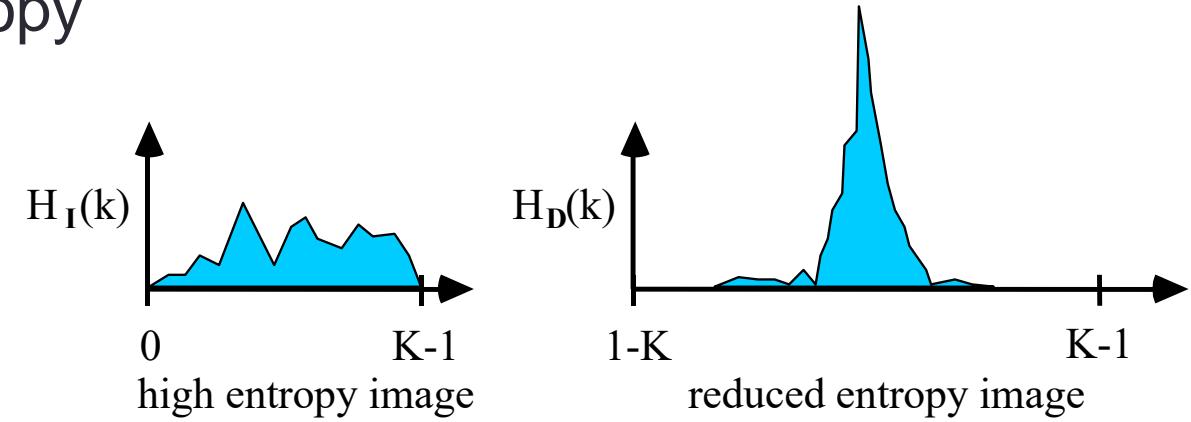
- Run-length encoding needs adjacent pixels to be **equal**
- Pixels are more often **highly correlated** (dependent)
 - Not equal, but can predict the image pixels to be coded from those already coded
- Each pixel value (except at the boundaries) is predicted based on its neighbors (e.g., linear combination) to get a **predicted** image.
- The difference between the original and predicted images yields a **differential** or **residual** image with a reduced set of values.
- The differential image is encoded using Huffman coding, or similar.

Differential Pulse-code Modulation

- Code the difference between adjacent pixels

Original pixels: $82, 83, 86, 88, 56, 55, 56, 60, 58, 55, 50, \dots$ \longrightarrow DPCM: $82, 1, 3, 2, -32, -1, 1, 4, -2, -3, -5, \dots$

- Prediction is that the next pixel value = current one
- Need the first value to provide a point of reference
- Invertible (lossless) and lower entropy



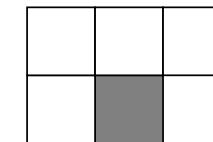
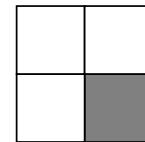
Predictive Coding

- Higher order pattern prediction
- Use both 1D and 2D patterns (to predict shaded pixel)

1D Causal:



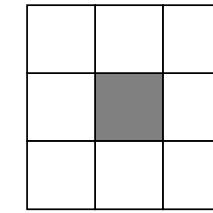
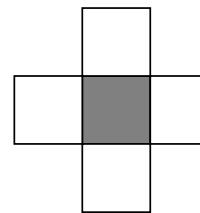
2D Causal:



1D Non-causal:

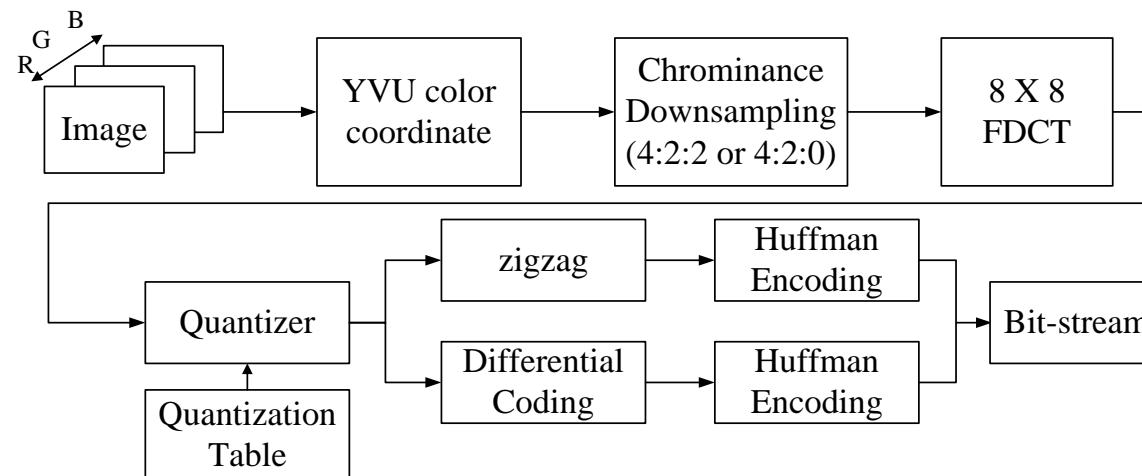


2D Non-Causal:



A Complete System: JPEG

- A set of methods with a common *baseline* system
 - Discrete Cosine Transform
 - Quantisation
 - Variable length encoding
- A JPEG-compatible product must only include support for the baseline



JPEG Compression

- Increasing the amount of quantisation reduces file size but introduces artefacts: blocks become visible



100 dpi low JPEG compression



File size:
248K

100 dpi medium JPEG compression



File size:
49K

100 dpi high JPEG compression



File size:
22K

NEXT WEEK

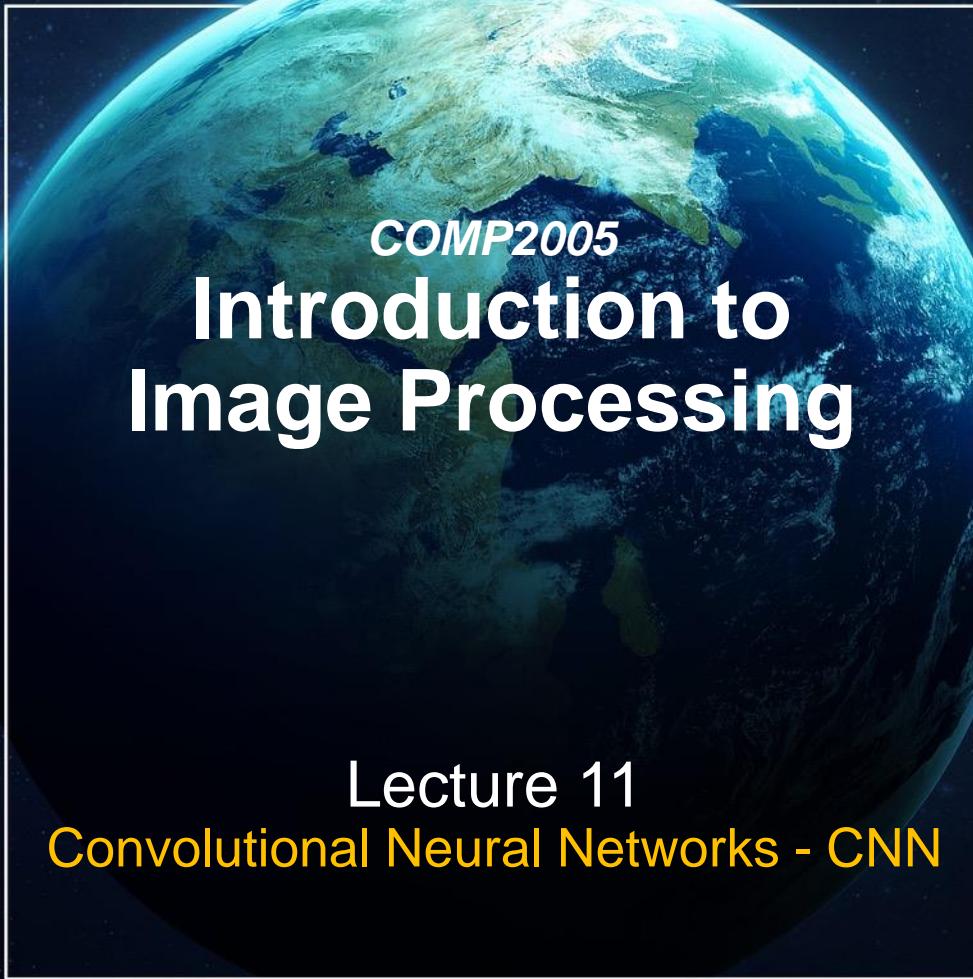
CNN

Conclusion and Exam Revision



University of
Nottingham

UK | CHINA | MALAYSIA



COMP2005
**Introduction to
Image Processing**

Lecture 11
Convolutional Neural Networks - CNN



University of
Nottingham

UK | CHINA | MALAYSIA

COMP2005: Introduction to Image Processing
Week 33 – 10:00am Friday – 10 May 2024



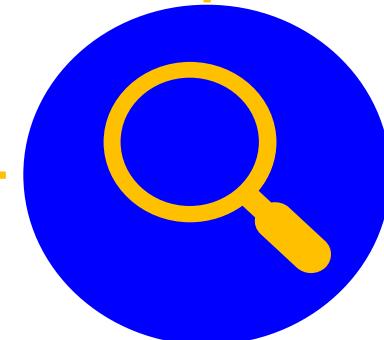
valid for 65 minutes from 9:55am
generated 2024-01-20 03:09



Learning Outcomes

IDENTIFY

- 1. Background
- 2. Neural Networks
- 3. Convolutional Neural Network
 - Convolutional layers
 - Pooling Layers
 - Activation Functions
 - Final stage : Softmax
- 4. Applications
- 5. Future



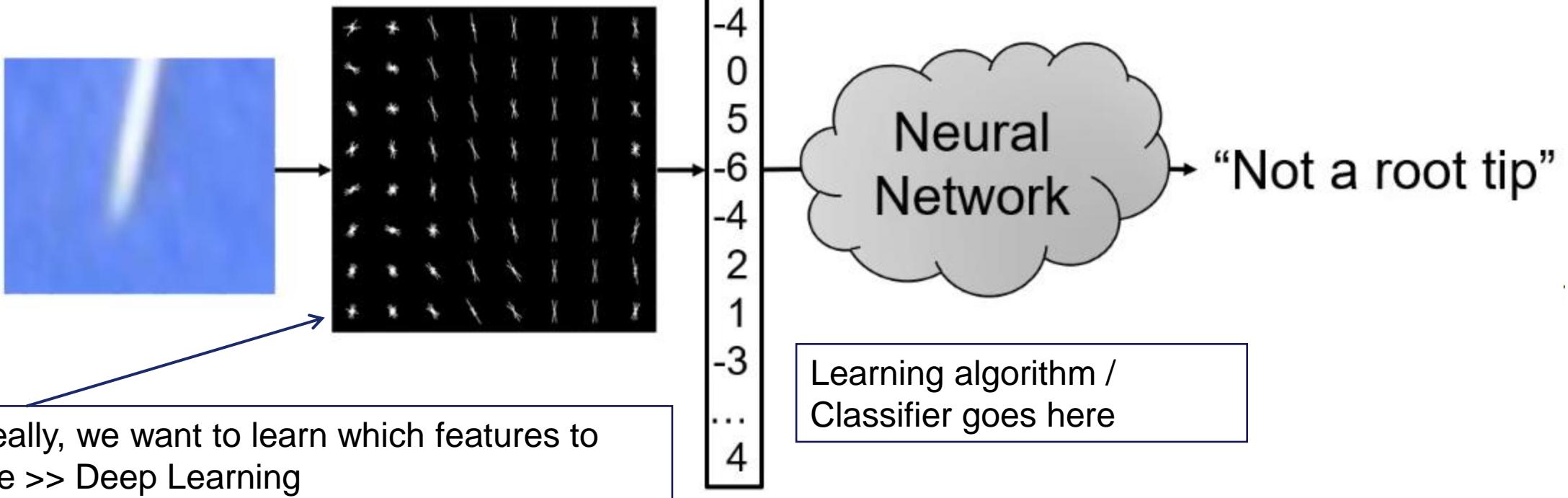


Background



Traditional Pipeline for Machine Learning

A classifier on the feature vector



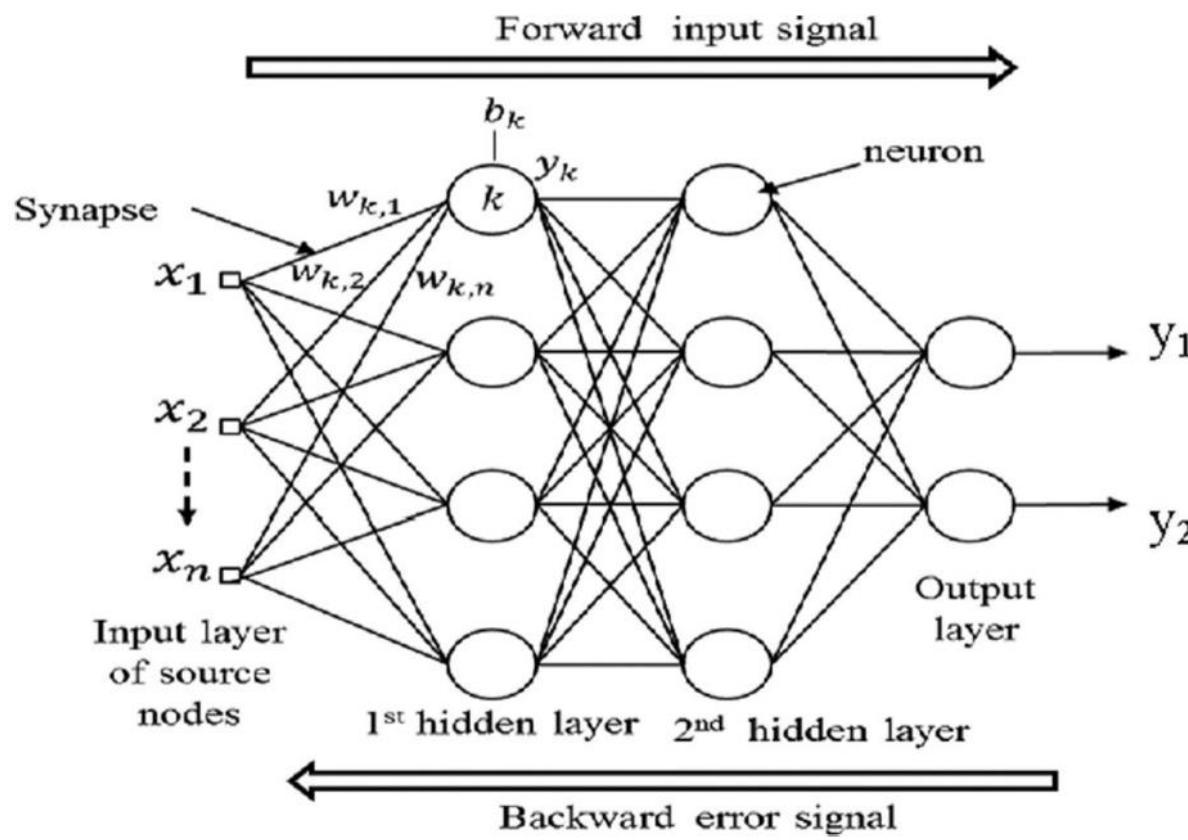


What is Deep Learning

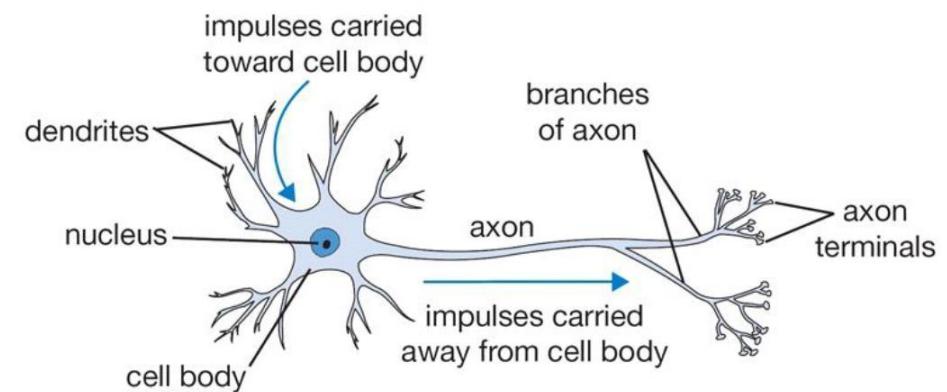
- Deep Learning is a popular AI technique
- Essentially a kind of Neural Network
- Deep refers to ability of having Many Layers in the network
 - Which was not possible in traditional Neural Networks
- What led to the development of deep learning ?
 - Several factors including :
 - GPU development
 - Algorithm improvement
 - Availability of large training image sets
 -
- Looking into CNN as the main technique for deep learning



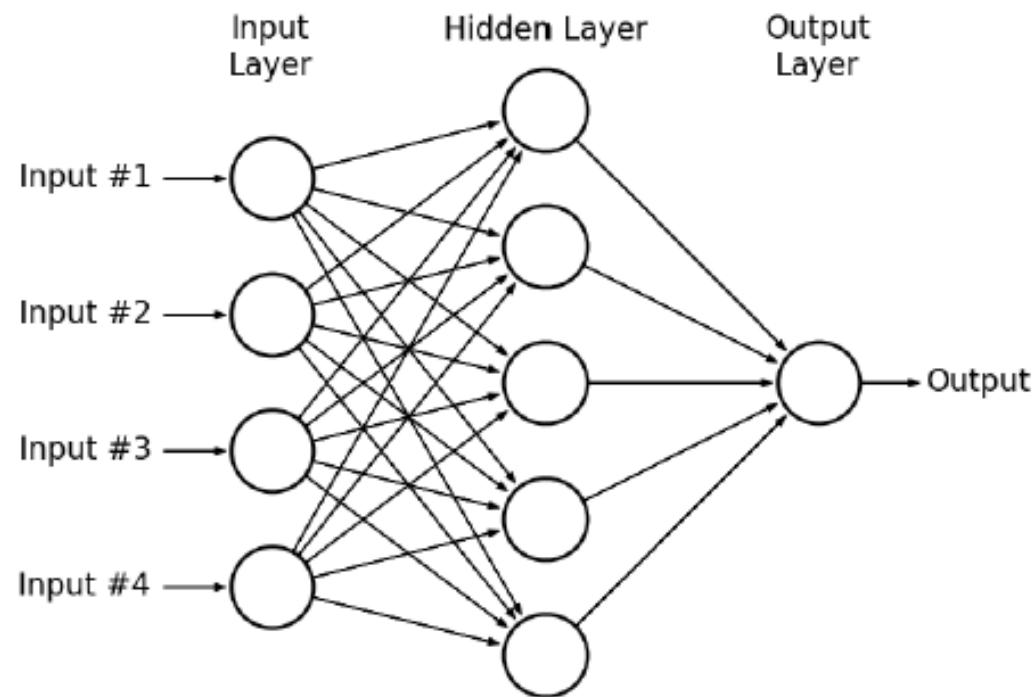
Classical Neural Network



Human Neurons



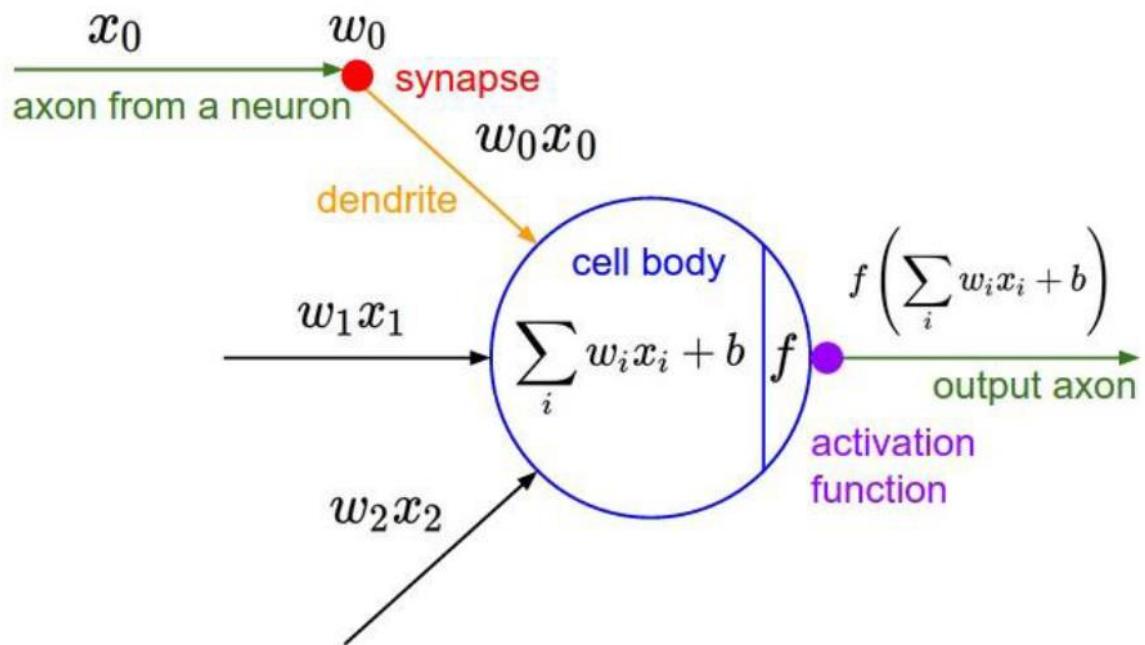
Humans have $\sim 100\text{-}1,000$ trillion connections in their brains





Classical Neural Network

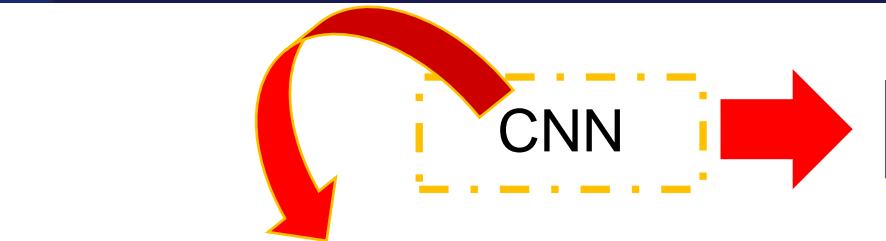
Modelling Neurons



Modern artificial networks tend to use 100k – 10b connections



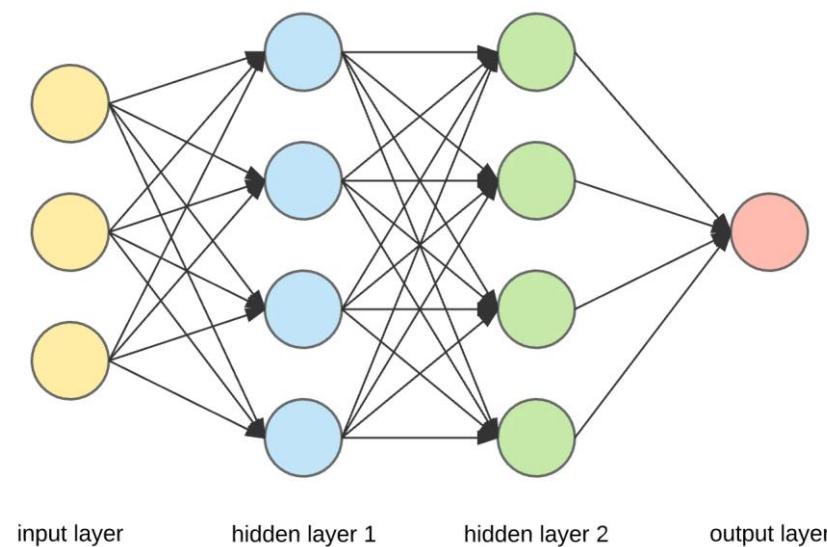
Inspiration...



Convolutional Neural Network

DID YOU KNOW: Each of the convolutional layers perform the image processing techniques that we learned throughout this module. Techniques such as *convolution/filtering*, *re-sizing*, *noise removal* and *edge detection* to name a few.

Interesting FACT



Inspired by

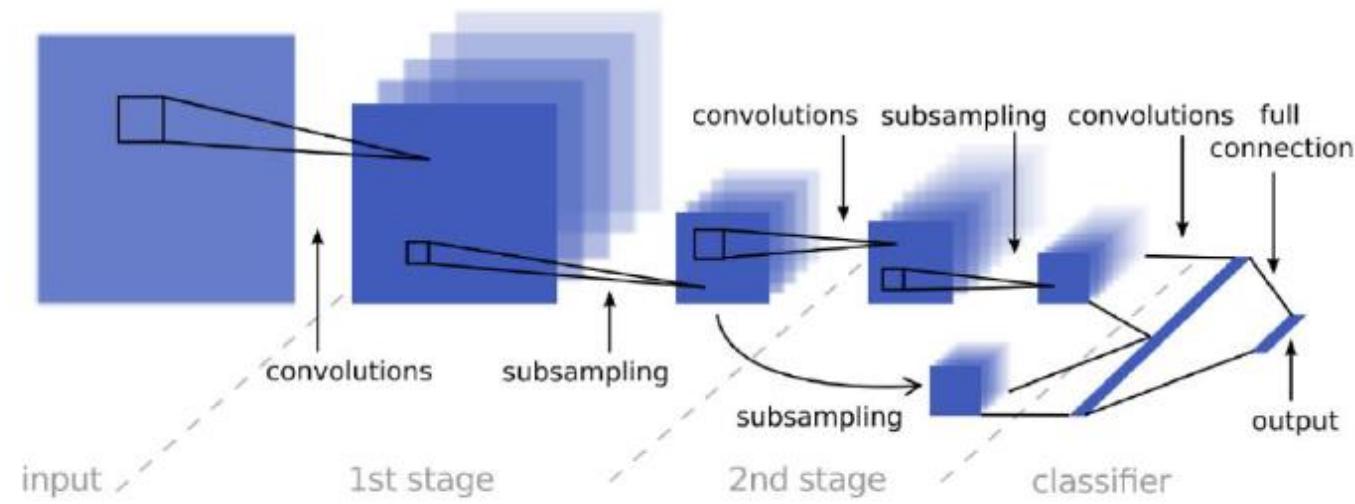
Artificial Neural Network¹

¹ Dertat, A. (8 August 2017). *Applied Deep Learning – Part 1: Artificial Neural Network*. Towards Data Science. <https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6>



Convolutional Neural Network

- Make the assumption the input is an *image*
- Neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- An end-to-end learned solution to many vision tasks
- Local analysis matches the natural structure of the most images
- Learn hierarchical models of image content





Multilayer Perceptron Network - MLP

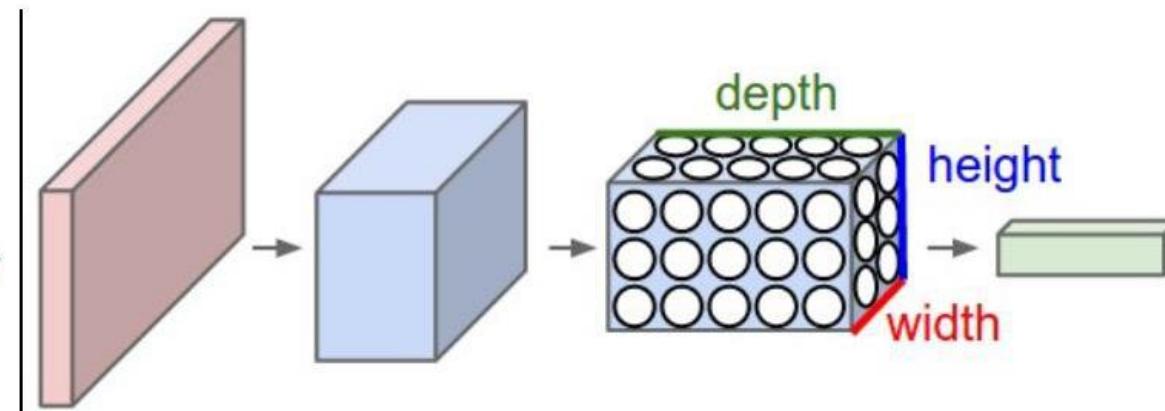
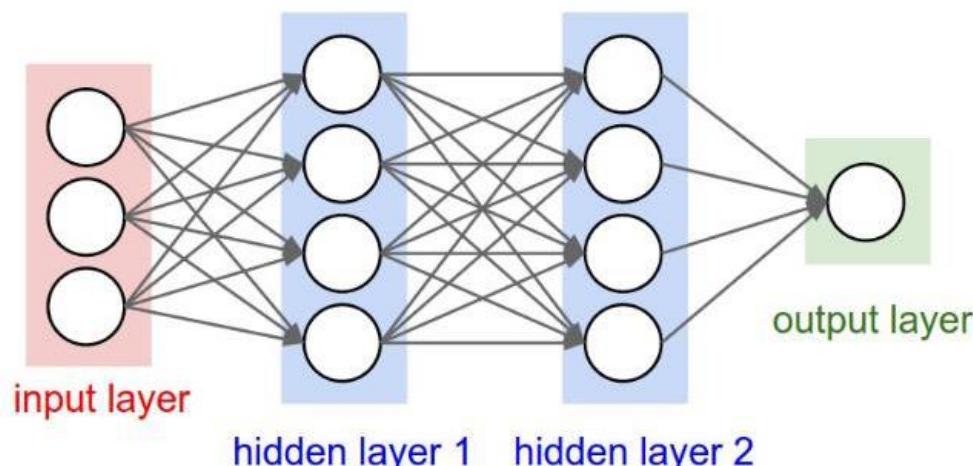
Wide application scenario—not just images

Neurons are **fully connected**—can't scale well to large size data (e.g.images)

Convolutional Neural Network - CNN

Neurons are arranged in '3D', each neuron is only connected to a small region of previous layer

Typical CNN structure: **Input-conv-activation-pool- fully connected- output**

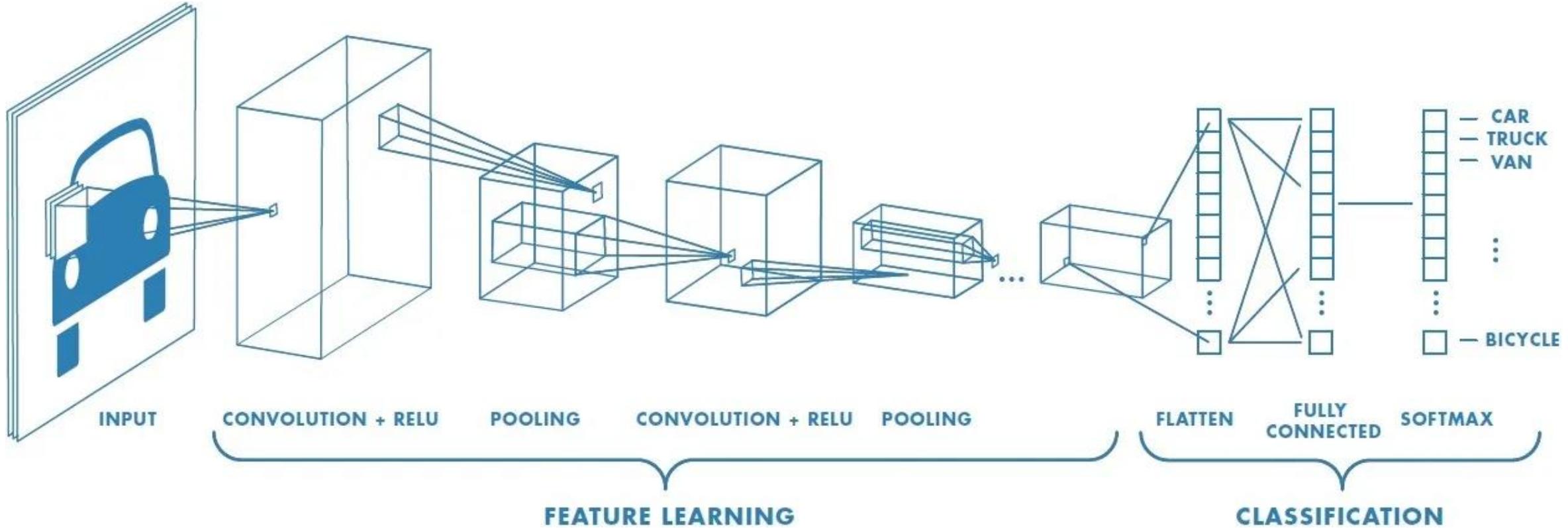


Convolutional Neural Network

Multilayer Perceptron (MLP)
Network
Convolutional Neural Network



Convolutional Neural Network Architecture



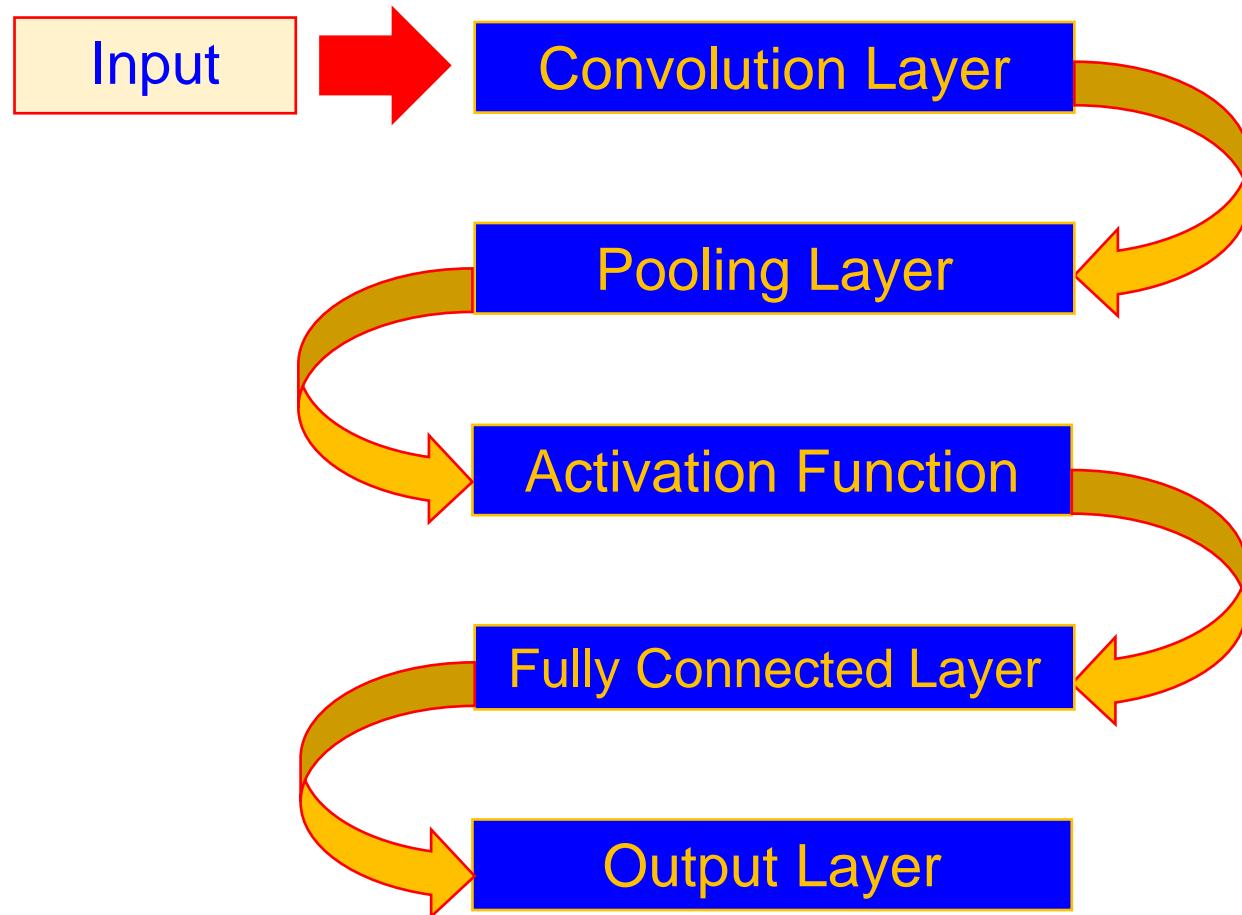
Extracted from: Raghav, P. (4 May 2018). *Understanding of Convolutional Neural Network (CNN) – Deep Learning*. Medium. <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>



Components of CNN



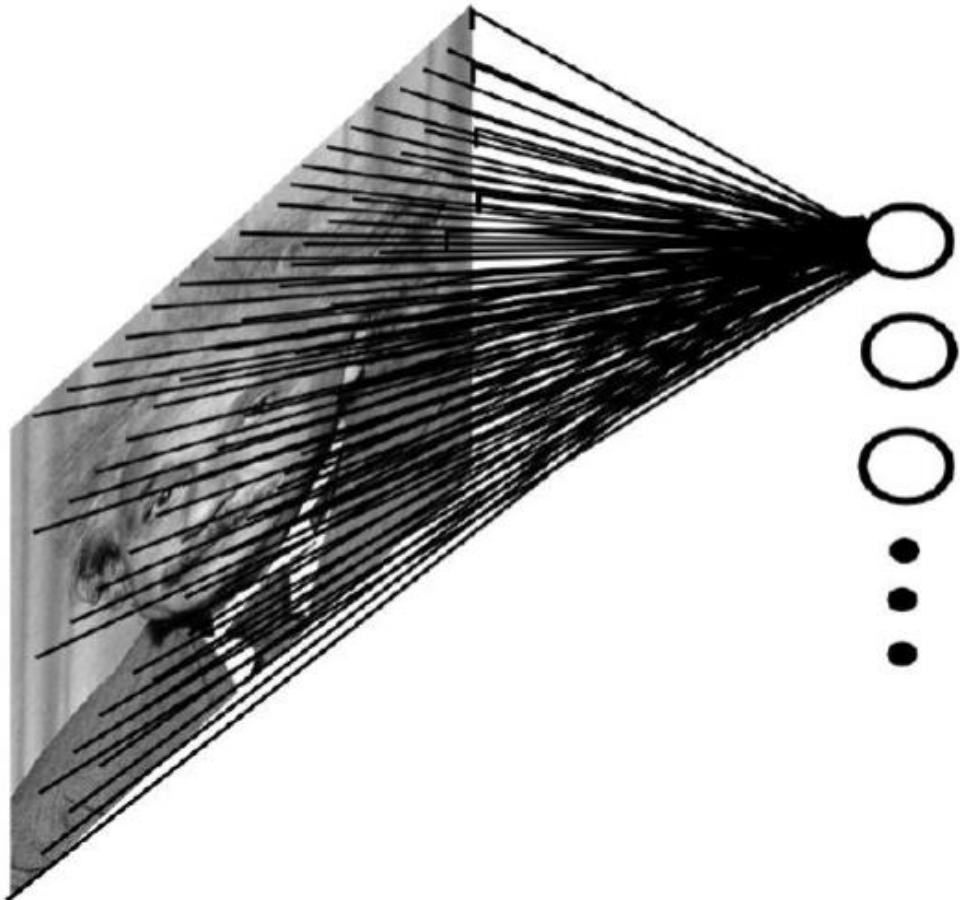
CNN Components





Locally Connected Layers – Traditional NN

Classical NN with fully connected hidden layers

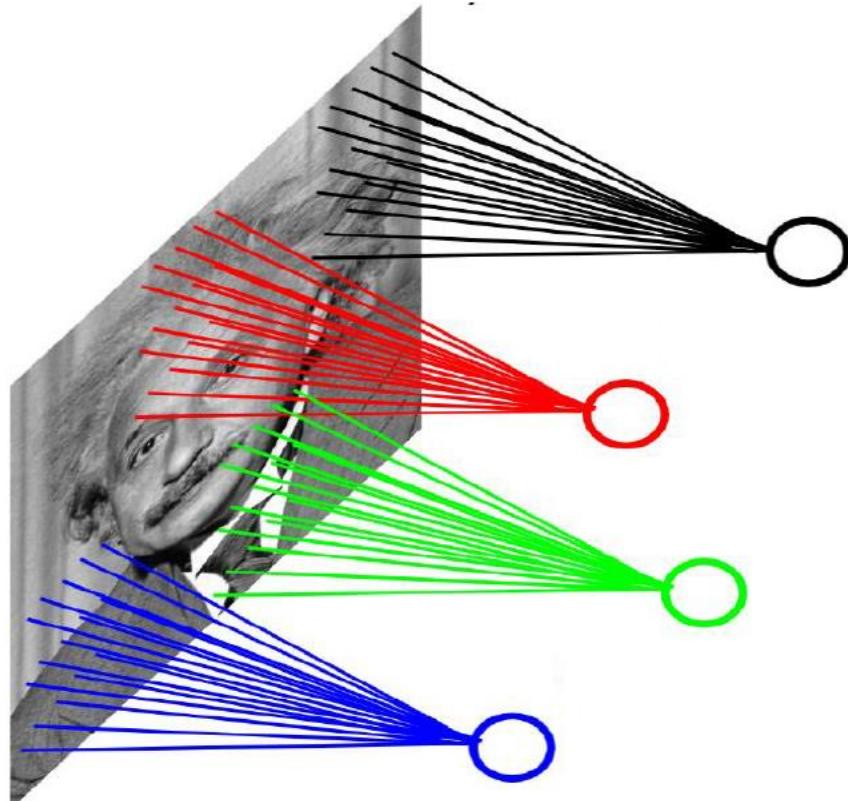


- 200x200 image, 40K hidden units (1 per pixel) means 1.6B weights to learn
- Waste of resources
Spatial correlation is local
most of the weights would be 0
- Would require an impractically large training set to learn this many weights



Locally Connected Layers

Classical NN with fully connected hidden layers



- CNNs' early layers are ***locally connected***

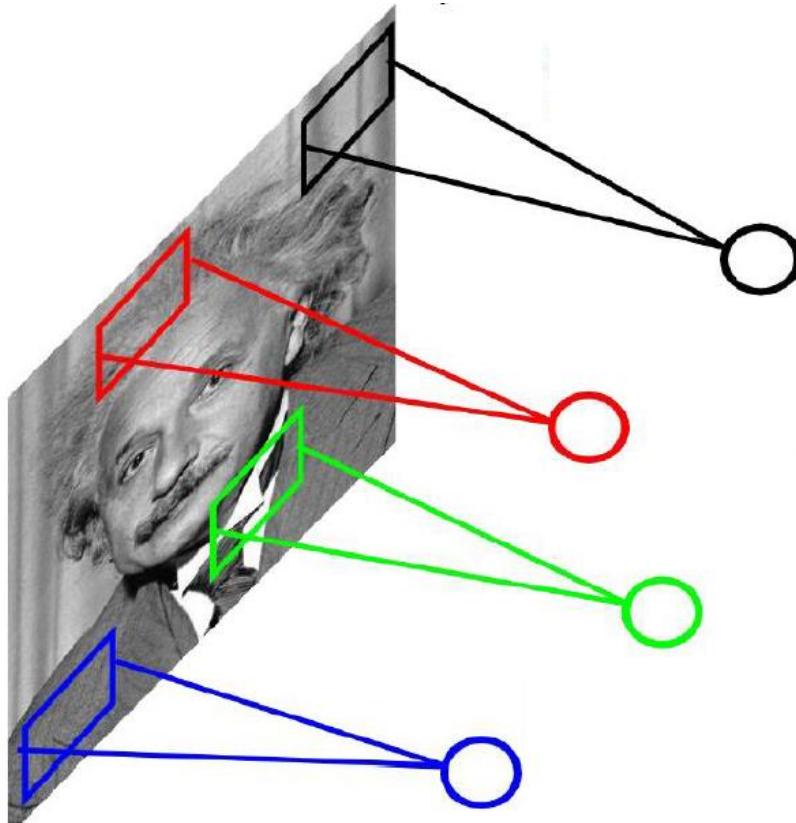
e.g. 200x200 image, 40K hidden units, fed by 10 x 10 filters, means 4M weights to learn

= much fewer than 1.6B



Convolution Layers

Classical NN with fully connected hidden layers

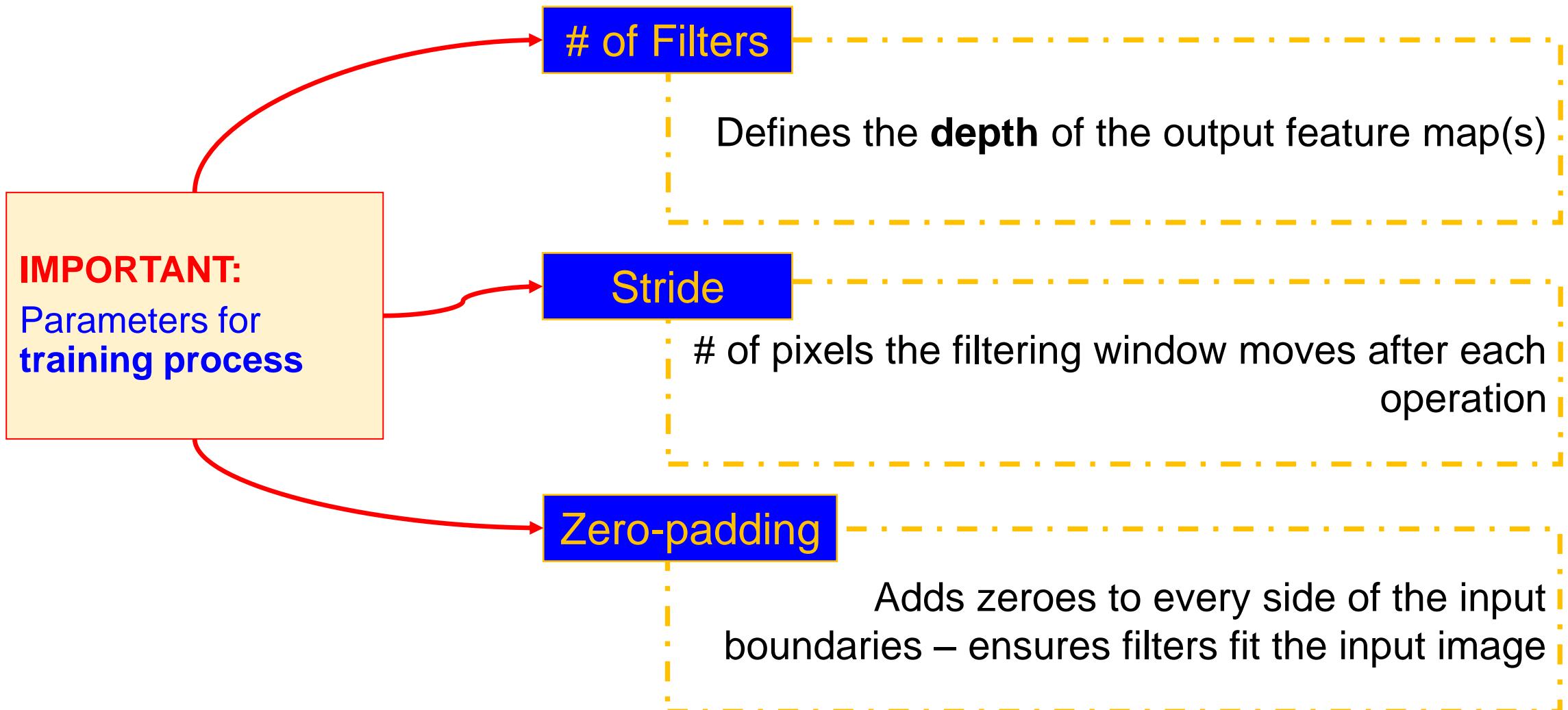


- In image processing/vision we usually want to apply the same convolution mask at each location
- Each neuron has the same weights

e.g. 200x200 image, 40K hidden units, 10 x 10 mask means only 100 weights to learn



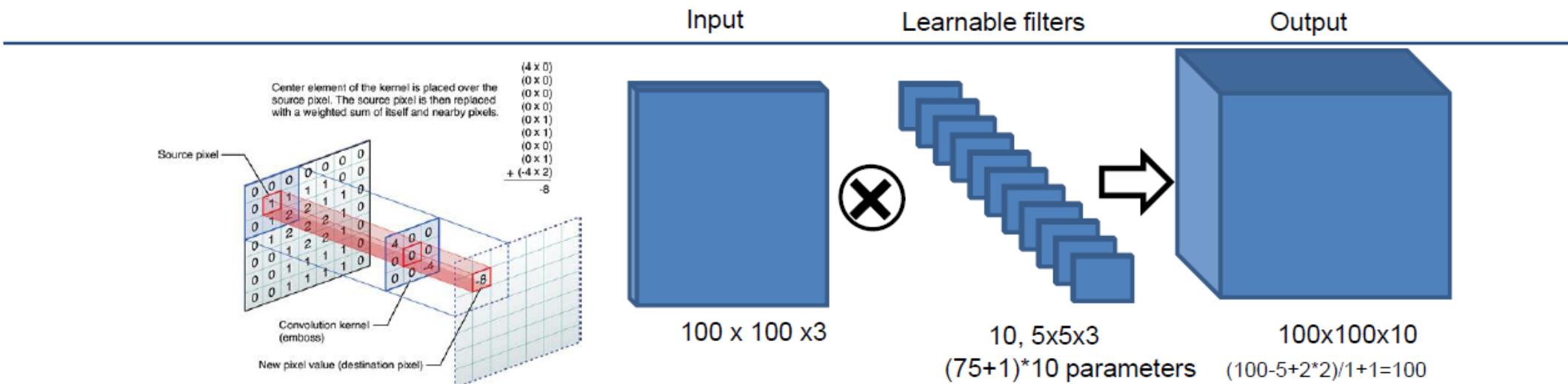
Convolutional Layer





Convolution

- We wish to *convolve* the input image with a set of *learnable, small-size filters*
- size(W); filter size(F); zero padding(P); stride(S)
 - F, conv filter size, is like a *capture field*
- Output volume size calculation **(W-F+2P)/S+1**

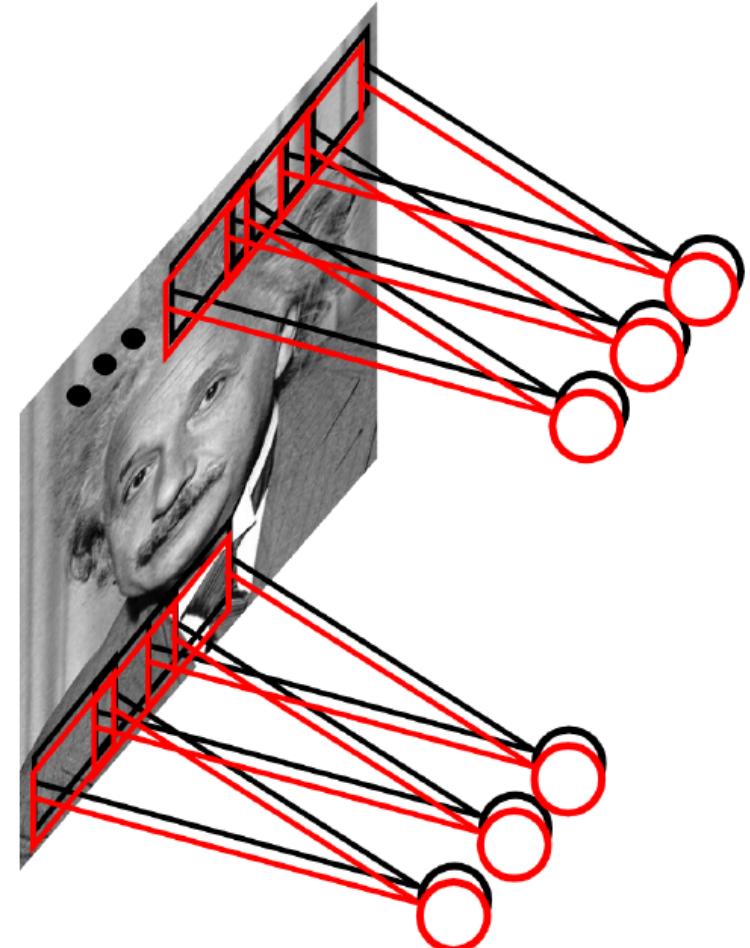


Does this look familiar?



Convolution Layer

- We can afford to learn multiple filters
e.g. 100 10x10 masks is only 10K parameters
- *Convolutional layers* are filter banks performing convolutions with the learned kernels (masks)
- Could be applied to all pixels, or have a small ‘stride’ to spread them out

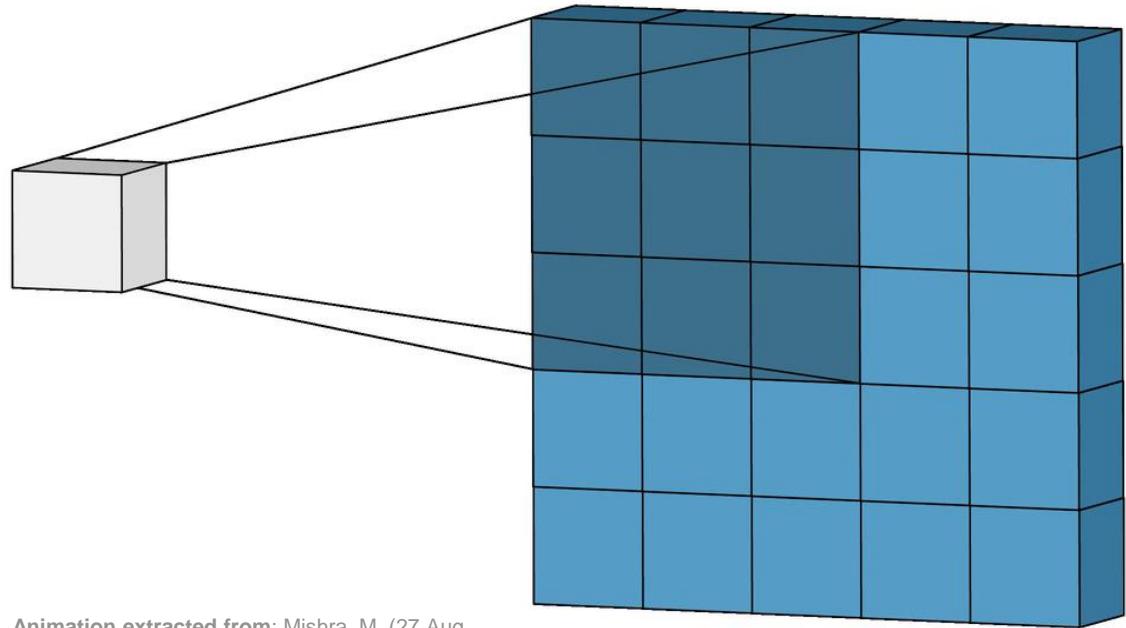




Convolutional Layer

- Most important layer
- Performs major computations
- Takes in **input image**, performs **filtering** which produces **feature map(s)**
- Filters using image processing techniques (e.g., **edge detection**, **blur** and **sharpen**)
- Filtering is performed using 3x3 kernels to perform the dot product

Resulting array [with weights] (*shown in grey in the animation below*) is known as **feature map** or **activation map**



Animation extracted from: Mishra, M. (27 Aug 2020). *Convolutional Neural Networks, Explained*. Medium.
<https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

Filter weights are only **adjusted** via backpropagation & gradient decent during the training process

Note



Convolutional Layer

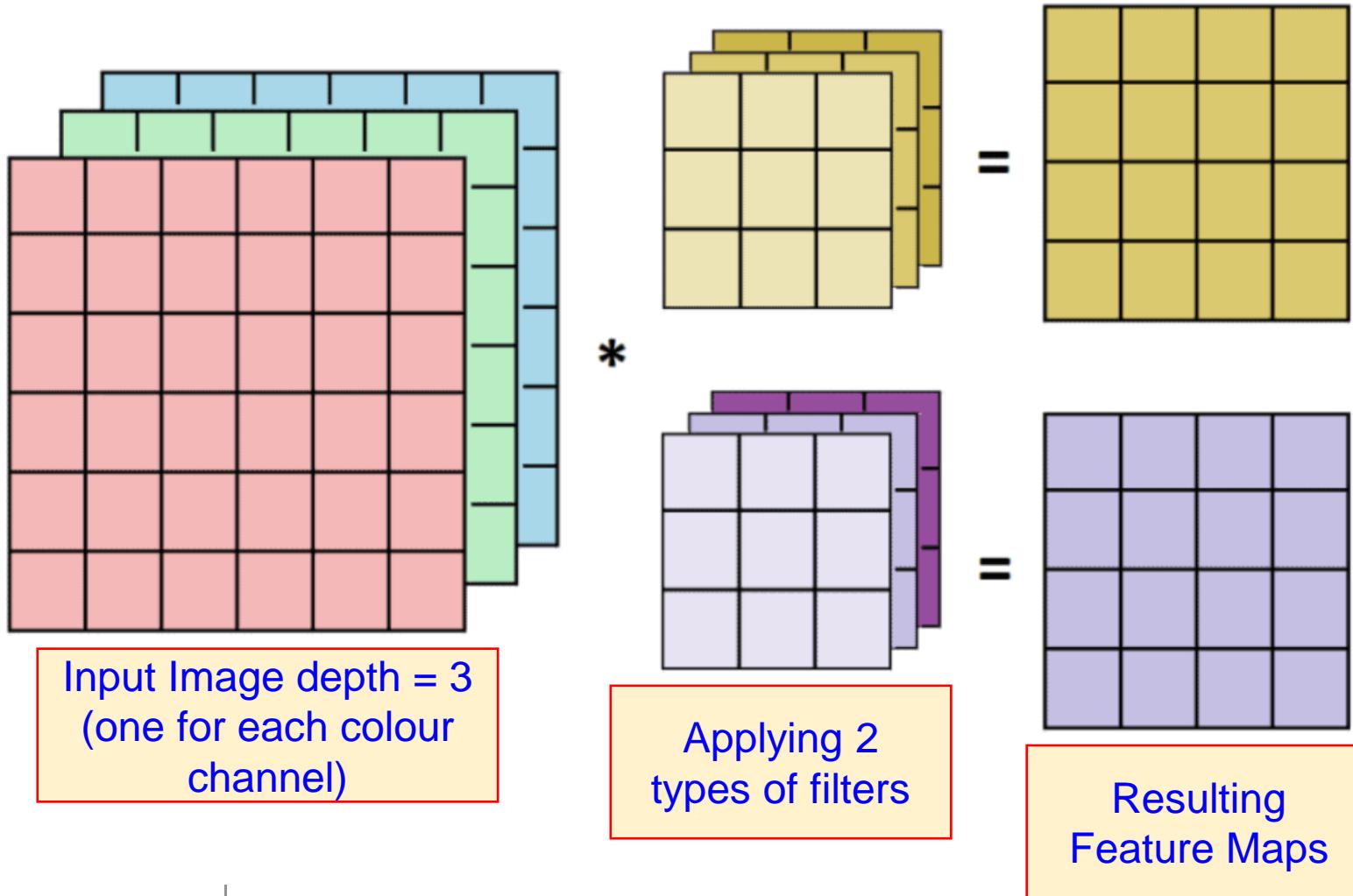
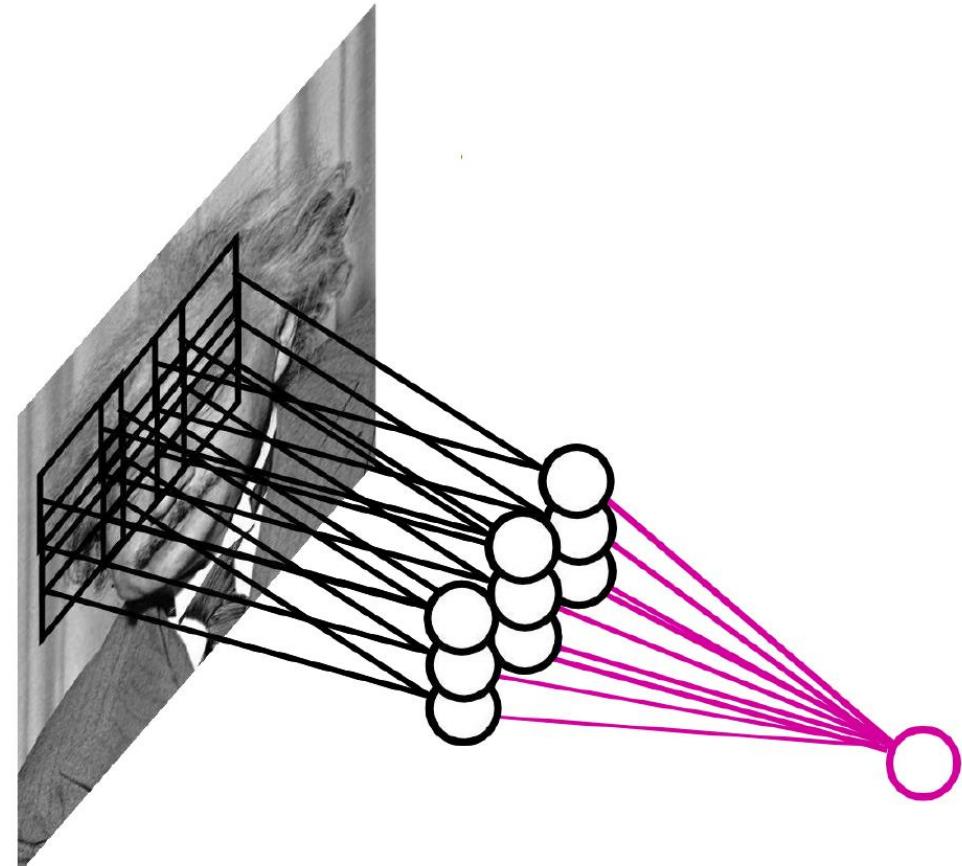


Illustration extracted from: Zvornicanin, E. (18 March 2024). *What is Depth in Convolutional Neural Network?*
<https://www.baeldung.com/cs/cnn-depth>



Pooling Layer

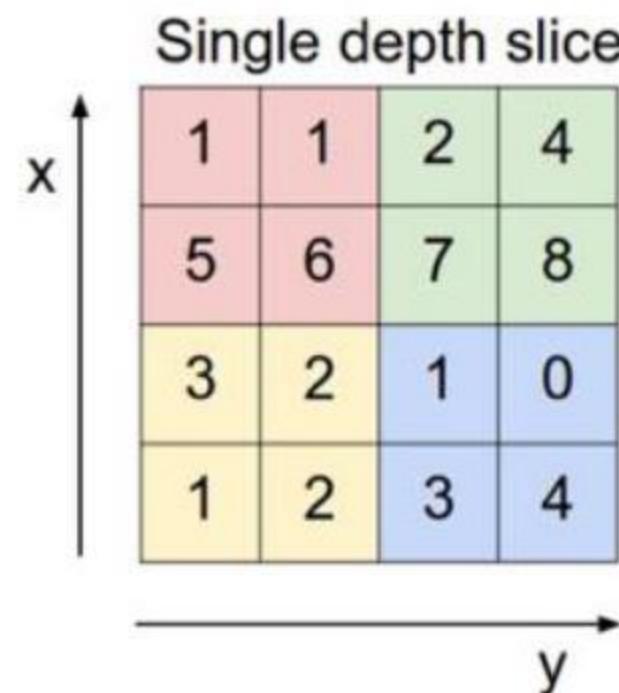
- Suppose one of our convolutions is an **eye detector** – how can we make the net robust to the exact location of the eye?
- By pooling (e.g. taking the max) filter responses at different locations
- Don't pool different features





- A number of pooling methods exist, including subsampling
- Effect is to reduce the resolution of the filter outputs
- *Subsequent convolutional layers therefore access larger areas of the image*

Name	Pooling formula
Average pool	$\frac{1}{S^2} \sum x_i$
Max pool	$\max\{x_i\}$
L2 pool	$\sqrt{\frac{1}{S^2} \sum x_i^2}$
L_p pool	$\left(\frac{1}{S^2} \sum x_i ^p \right)^{\frac{1}{p}}$



max pool with 2x2 filters
and stride 2

6	8
3	4



Pooling Layer

?

Occurs after a convolutional layer. Reduces the dimensionality of the resulting conv layer

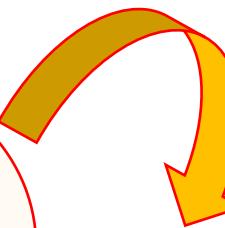
12	23
9	20

Average Pooling

4	25	44	10
8	14	8	33
17	2	16	34
5	13	24	7

25	44
17	34

Max Pooling



Famously Used

Three Types of Pooling Operations

16

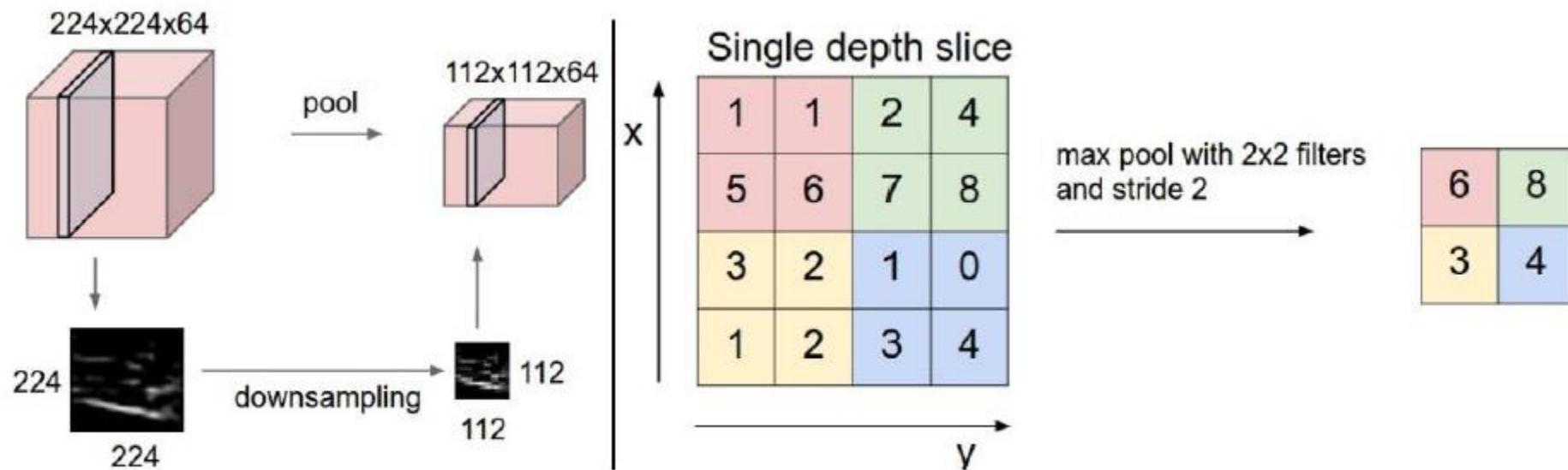
Global Average Pooling

Extracted from: Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, 8, pp.1-74



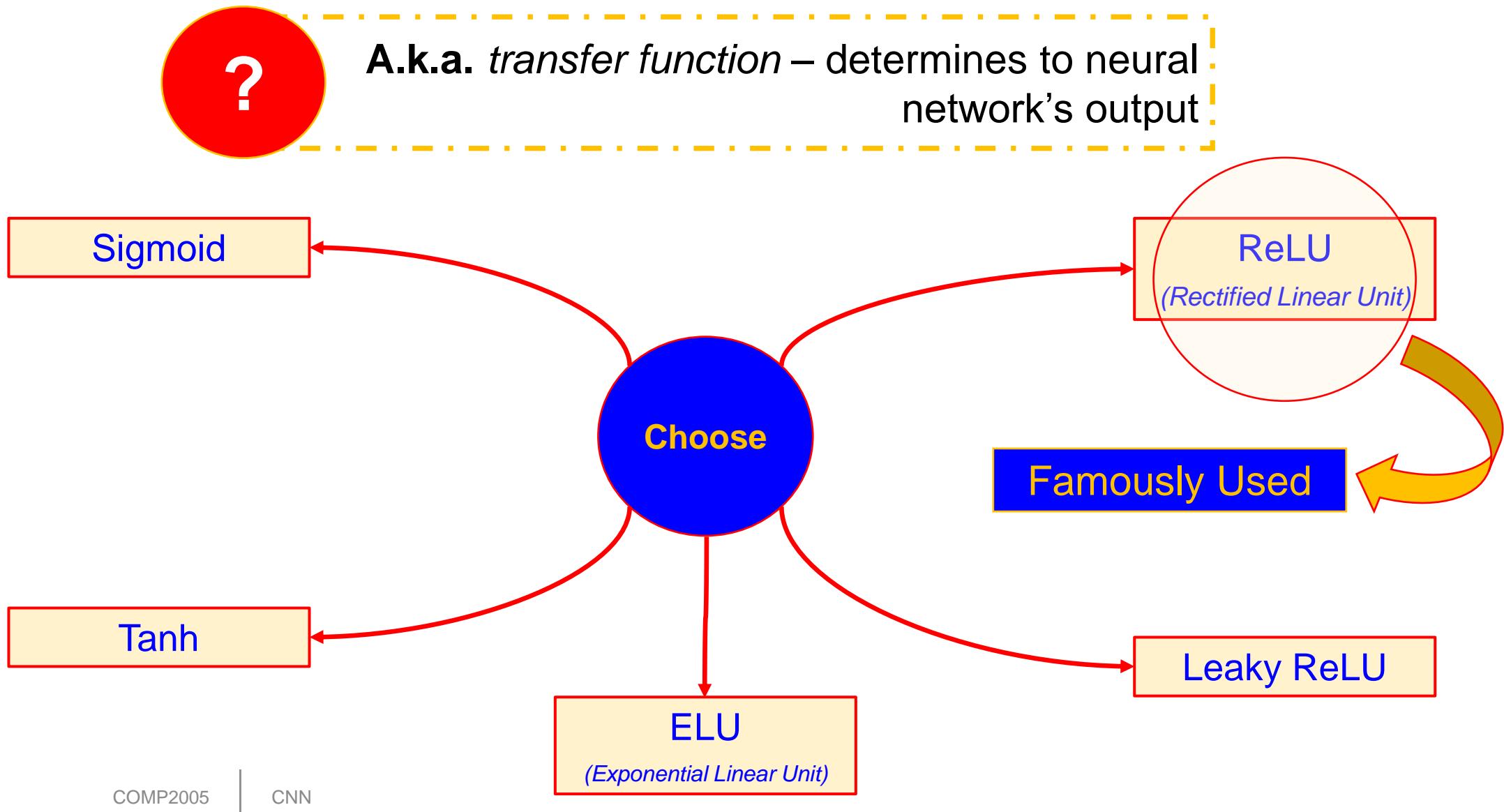
Effect of Pooling

- Reduce the spatial size of the representation and reduce the amount of parameters
- Effectively down-sampling the input to increase the receptive field size
- Max operation with stride of 2 is a popular choice



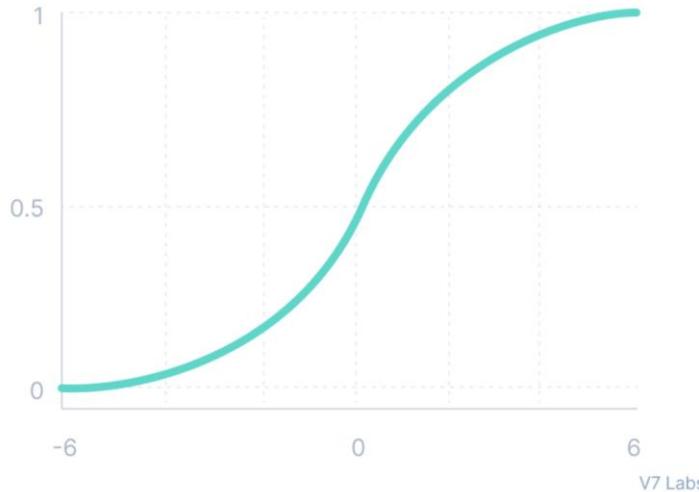


Activation Function

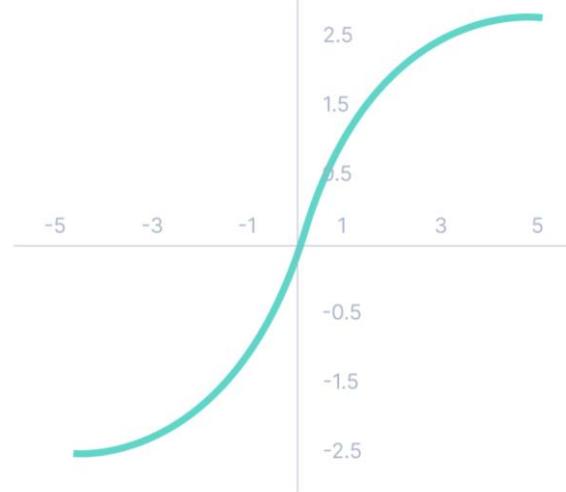




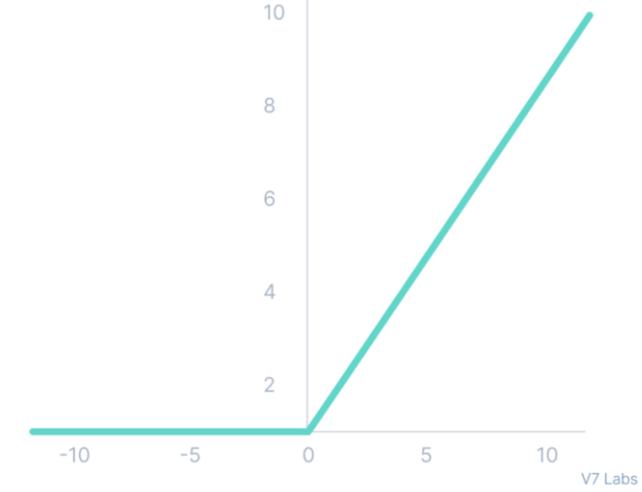
Sigmoid



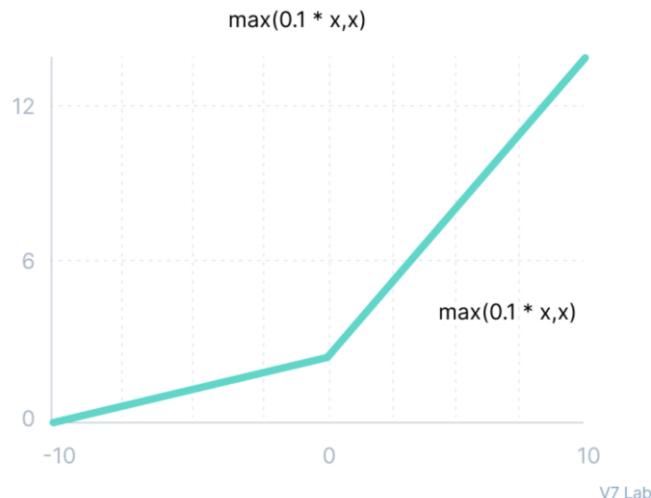
Tanh



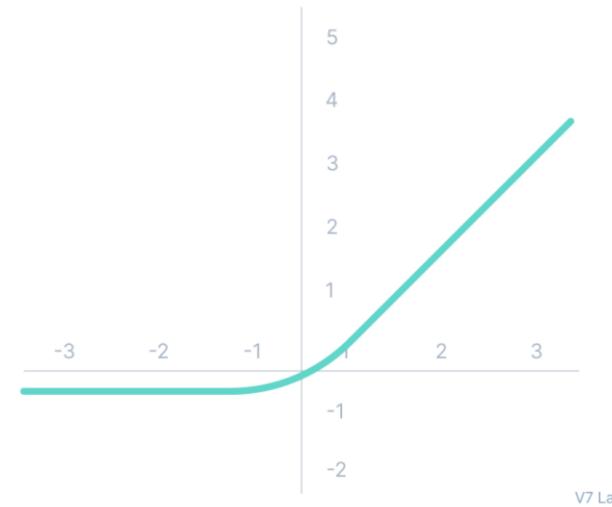
ReLU



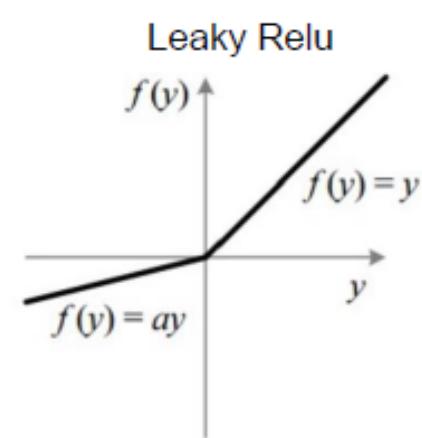
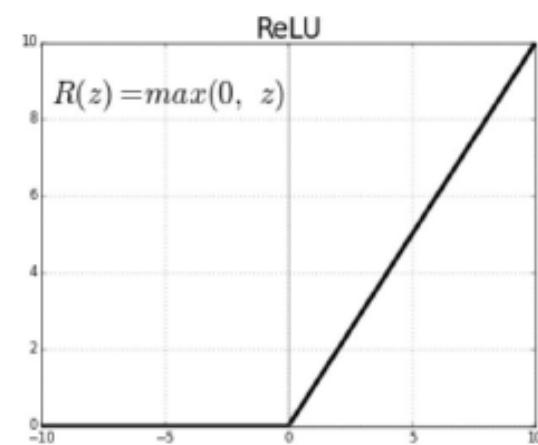
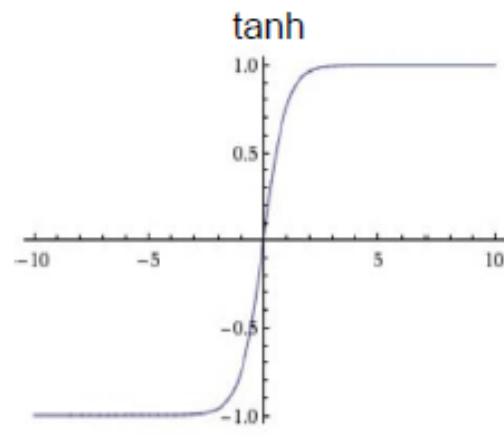
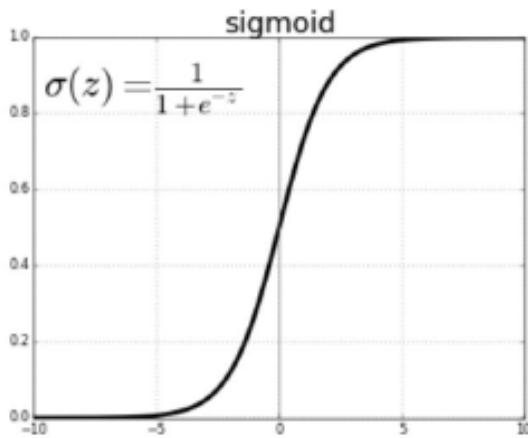
Leaky ReLU



ELU



Graphs extracted from: Baghetti, P. (27 May 2021). *Activation Functions in Neural Networks [12 Types & Use Cases]*. V7. <https://www.v7labs.com/blog/neural-networks-activation-functions>



- ✓ Transfer the range to $[0, 1]$
- ✗ Saturate and kill gradients: small gradient at region of 0 and 1

- ✓ Zero centred range: $[-1, 1]$
- ✗ Saturate and kill gradients: small gradient at region of 0 and 1

- ✓ Solves vanishing/exploding gradients
- ✓ Simple to calculate
- ✗ Some neurons can be 'dead' with negative input

- ✓ Overcome the 'dying neuron' problem
- ✗ Performance not consistent



Fully Connected Layer

- **FINALE** layer
- Utilises features extracted from previous layers, performs task classification

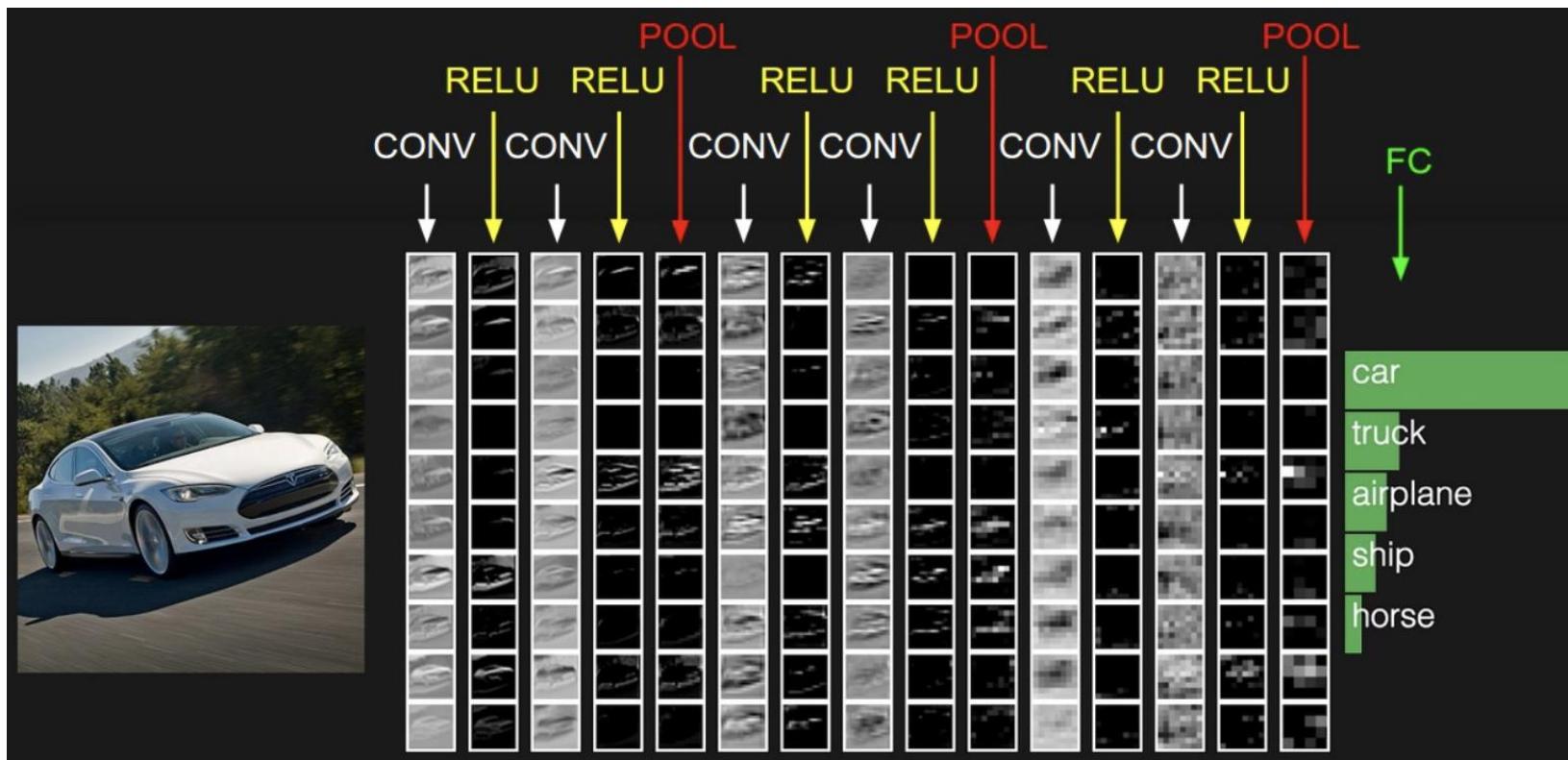
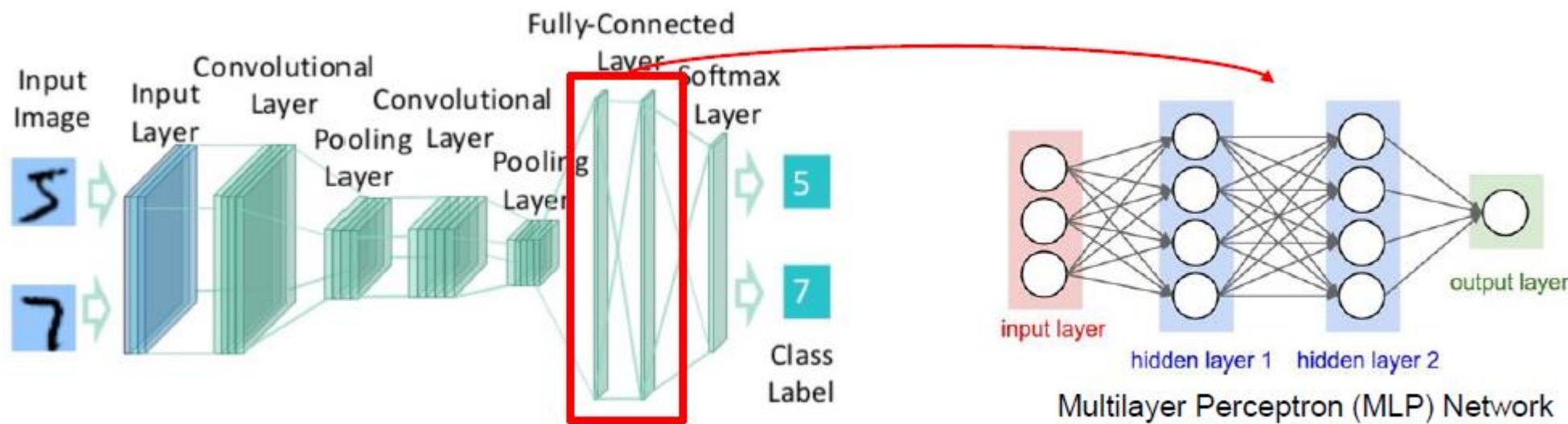


Image extracted from: Standard University.
(n.d.). CS231n Convolutional Neural Networks for
Visual Recognition.
<https://cs231n.github.io/convolutional-networks/#fc>



Fully connected layer and softmax

- Global feature learning: fully connected to all activations in the previous layer, as the same in MLP.
- Softmax - converts the prediction to the range of [0..1] for each class



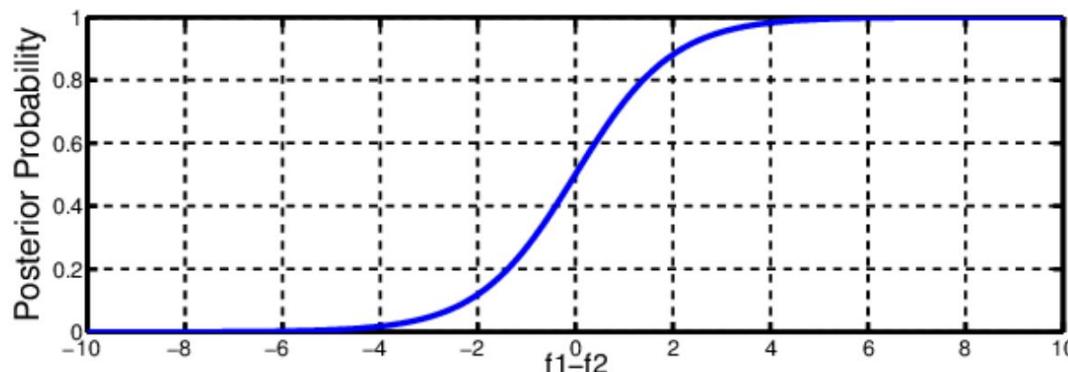
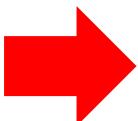


Output Layer



- Performs a *logistic function* to classify tasks
- Uses **Softmax** activation function where probability ranges from 0 to 1 – scores given to each class
- Sometimes, **embedded** within the FC layer

Softmax
Activation
Function



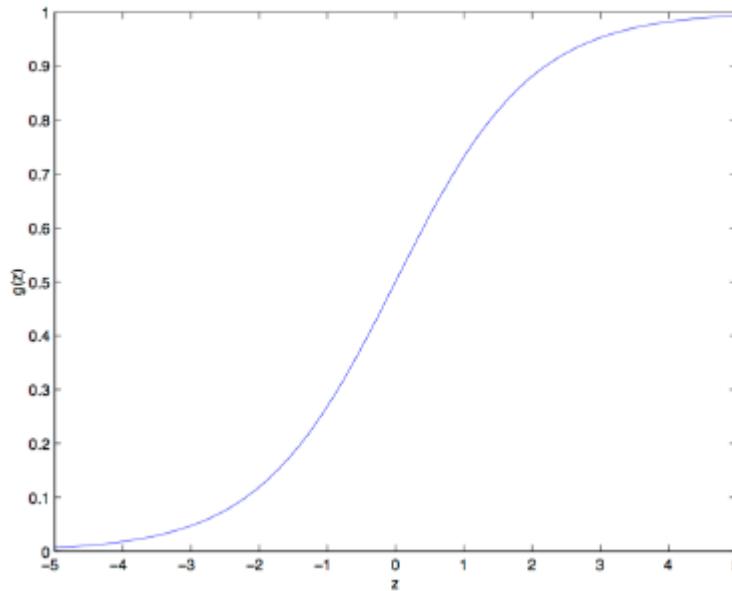
Graphs extracted from: Chen, B., Deng, W., & Du, J. (2017). Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5372-5381)



Softmax

- With *Linear regression*, we try to predict a real value y , for an input value x
Perhaps we are predicting someone's age from a picture
- Linear regression is not good if we are predicting if input data should be assigned class A or class B
Perhaps we are predicting Happy or not-Happy from a picture
- To do this there are more suitable kinds of regression
E.g. We can use *Logistic regression*
This can be used to “squash” a number into one of two classes

Example: *sigmoid function* →





Softmax

So why do we need a softmax function?

- We need a different function if we have *multiple* (ie.>2, non-binary) classes
 - Works a bit like logistic regression, but for multiple classes
- A softmax function takes a set of numbers as an input, and outputs a probability distribution spread over a set of k classes
 - We want to know which class k is the most likely given a data point
 - e.g. predicting if a person is happy, sad, grumpy, sleepy, etc. from a picture.
- Softmax allows us to do this
 - Gives us a probability for each class for a given input in the range 0..1
 - Probabilities sum to 1. e.g:

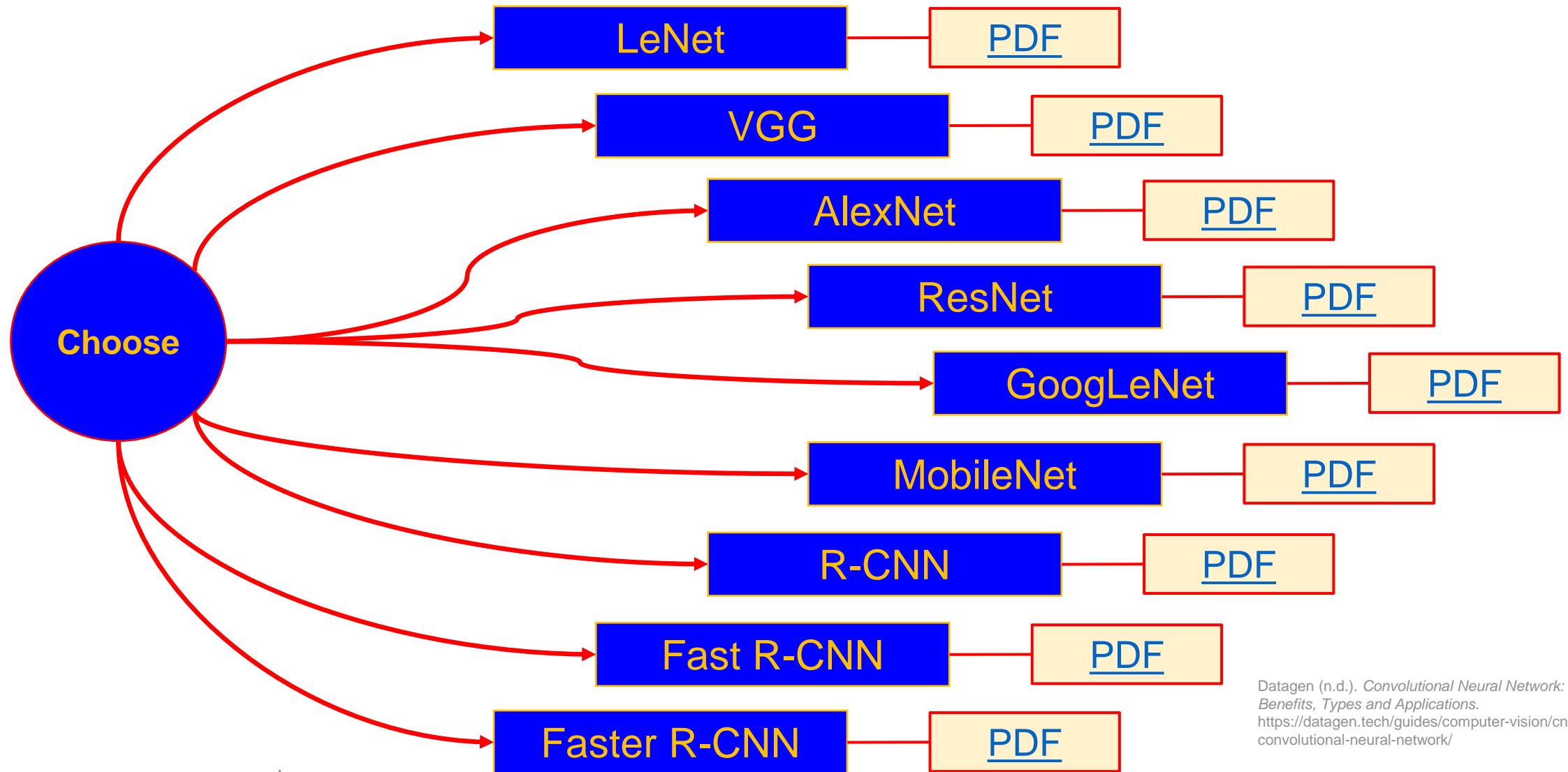
Class	Probability
Happy	0.05
Sad	0.32
Grumpy	0.44
Sleepy	0.19



Applications



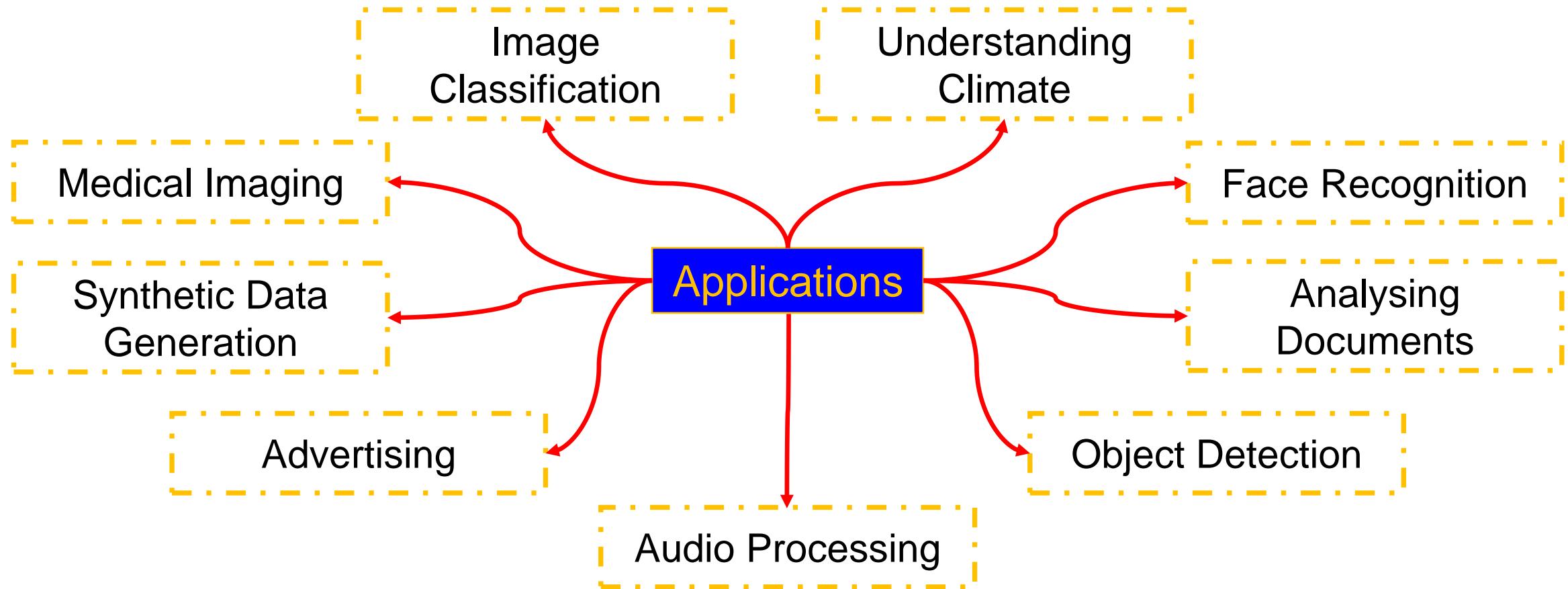
Famous CNN



DataGen (n.d.). Convolutional Neural Network: Benefits, Types and Applications.
<https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/>



Some Practical Applications of CNN



Keita, Z. (Nov 2023). *An Introduction to Convolutional Neural Network (CNNs)*. <https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns>

Ray, P. (14 Jan 2021). *Convolutional Neural Network (CNN) and its application - All you need to know*. <https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5>

MATLAB (n.d.). *What is a Convolutional Neural Network*. <https://www.mathworks.com/discovery/convolutional-neural-network.html>



Future



Vision Transformer (ViT)

- Designed for Computer Vision with remarkable results
- Uses **neural network** to split images into smaller patches, allowing model(s) to capture both local and global relationships within images.¹

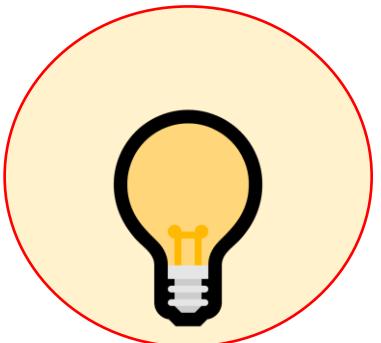


Animation extracted from: Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

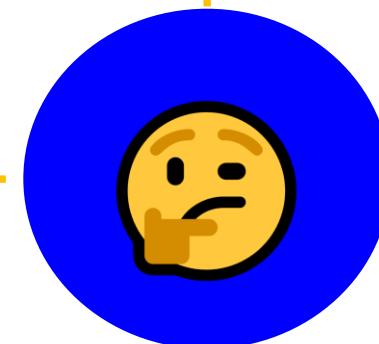
¹ Extracted and modified from: Hettiarachchi, H. (12 August 2023). *Unveiling Vision Transformers: Revolutionizing Computer Vision Beyond Convolution*.
<https://medium.com/@hansahettiarachchi/unveiling-vision-transformers-revolutionizing-computer-vision-beyond-convolution-c410110ef061>



Summary



- 1. Background
- 2. Components
- 3. Applications
- 4. Future





University of
Nottingham

UK | CHINA | MALAYSIA

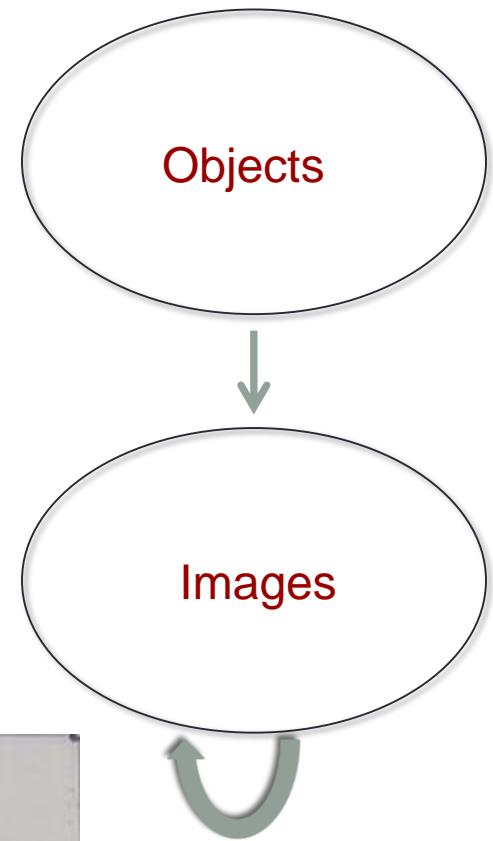
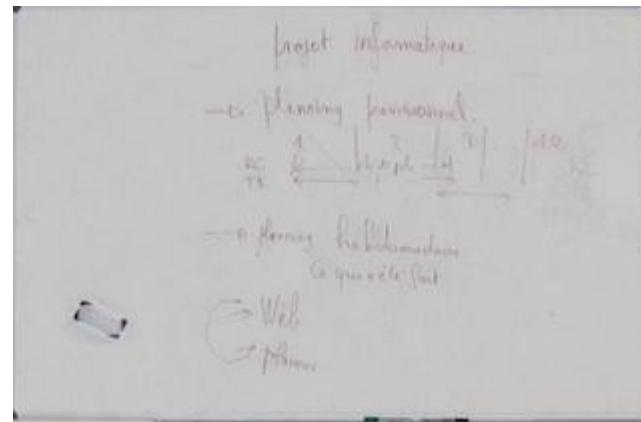
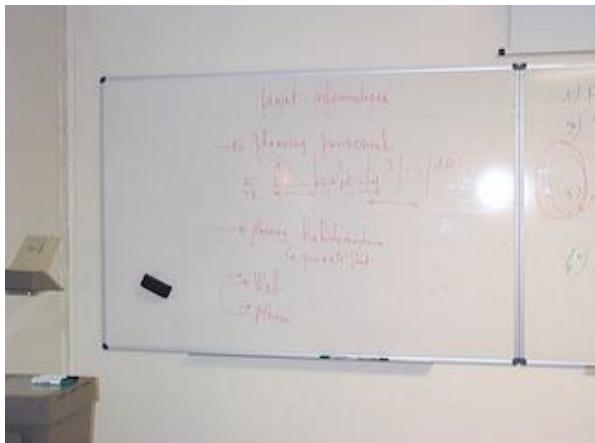
Questions

COMP2005

The End – The Whiteboard Problem Revisited

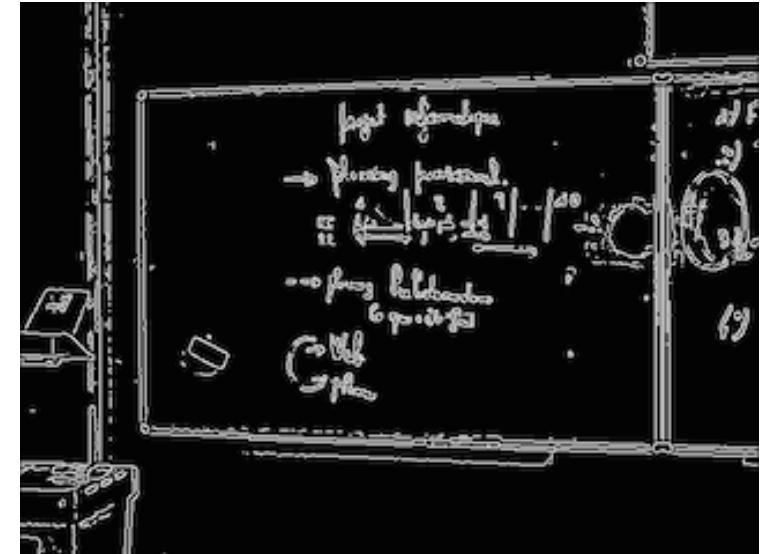
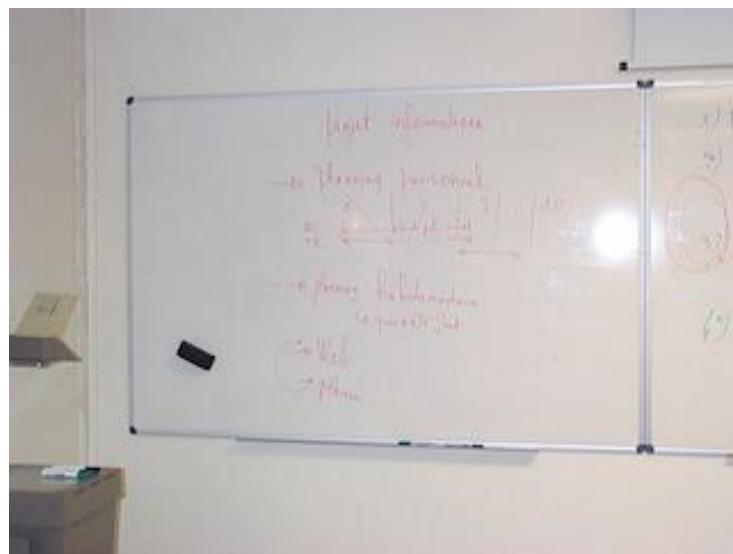
Remember This?

- Image(s) in, image(s) out
- Key information more easily seen/extracted
- More aesthetically pleasing



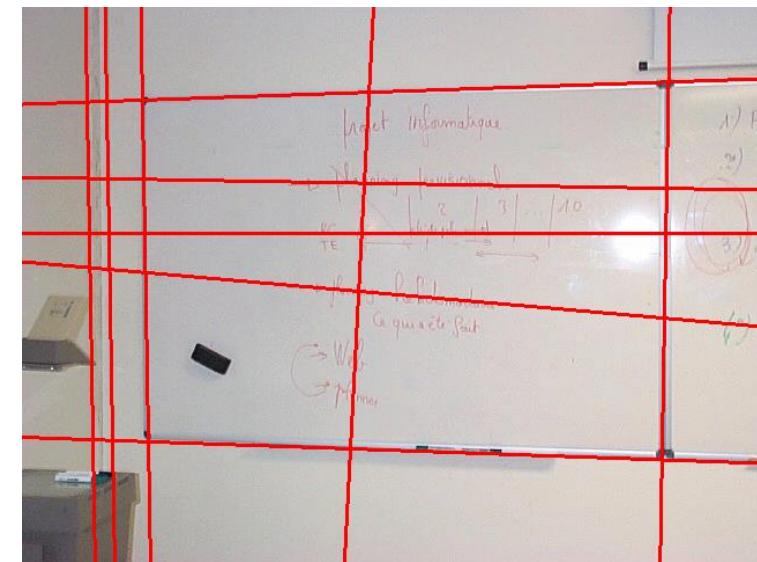
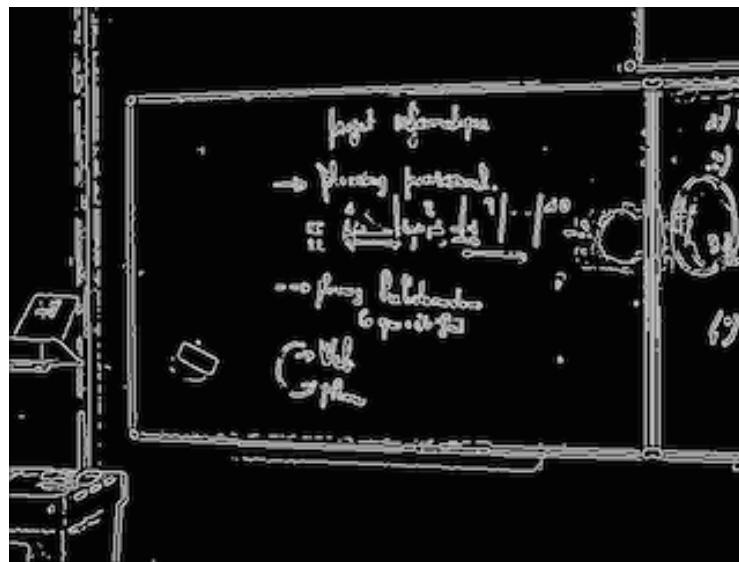
Whiteboard Problem Revisited

- Step 1: Edge detection
 - Achieved here with a variation on the Canny operator



Whiteboard Problem Revisited

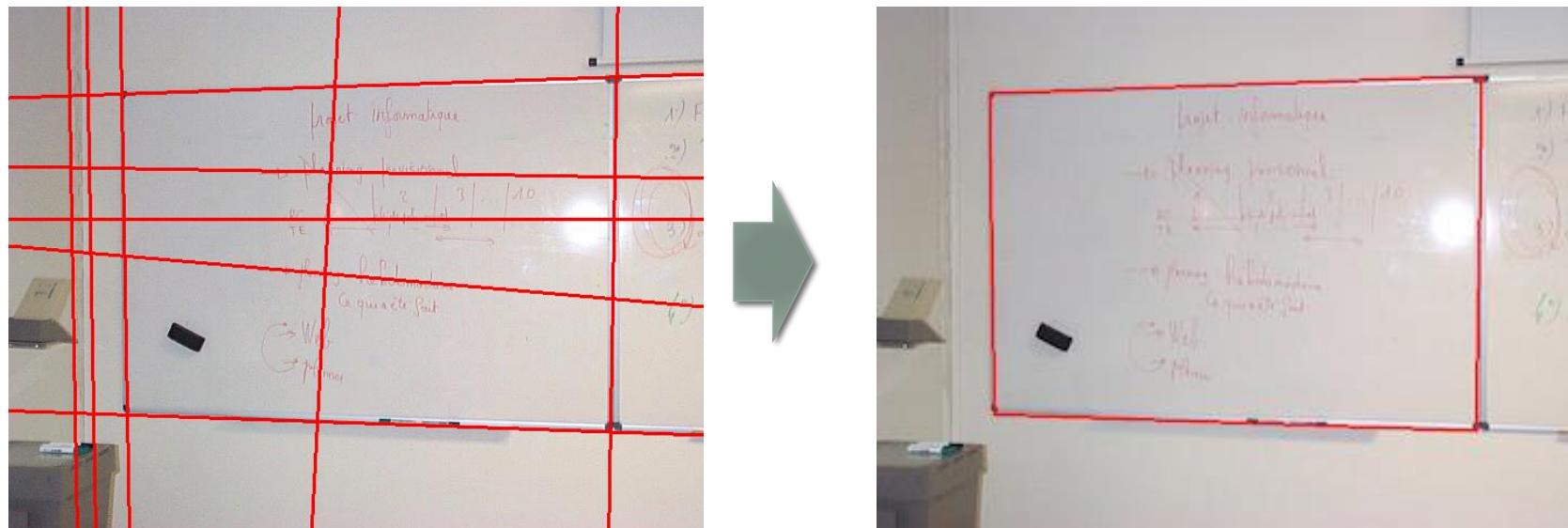
- Step 2: Line Finding
 - Two **Hough Transforms**; one detecting near horizontal (20° to -20°) and one near vertical (70° to 110°)



- Keep only the 5 longest horizontal and vertical lines

Whiteboard Problem Revisited

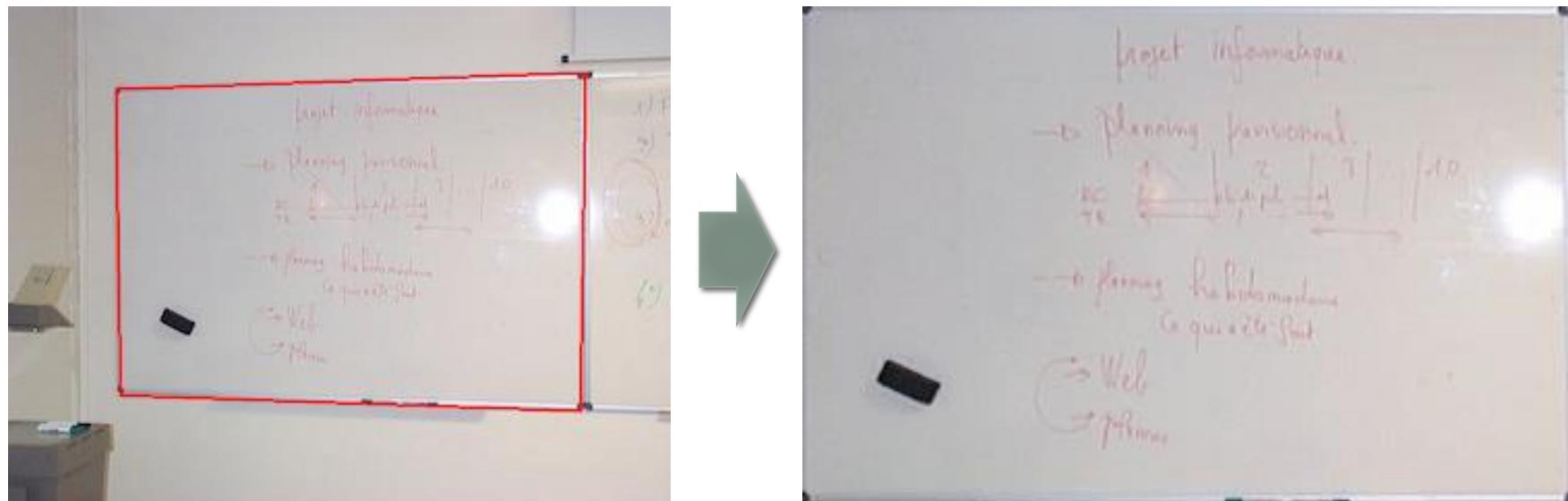
- Step 3: Detect whiteboard border
 - Find quadrilaterals above threshold size with neighbouring edges at 90° and opposites sides oriented the same ($\pm 30^\circ$)



- Pick the one which lies over the most edges

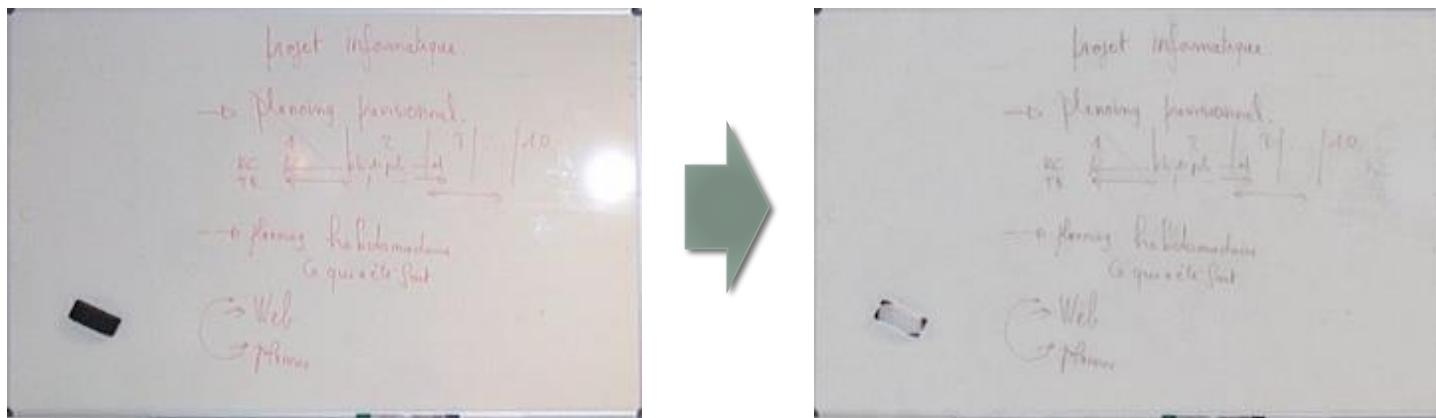
Whiteboard Problem Revisited

- Step 4: Distortion Correction
 - This requires **Geometric Transformation** of the image



Whiteboard Problem Revisited

- Step 5: Illumination Correction
 - Need to **enhance** the high frequencies (writing)



- Frequency domain processing is one way
- In **spatial domain**:
 - Approximate low frequency component by **smoothing**
 - **Subtract** smoothed image from original to approximate high frequencies
 - Add mean of “low frequency” image to “high frequency” image to make it bright enough

The End

- Image formation and acquisition
 - Background material: cameras, Bayer patterns
 - Basic terminology: sampling and quantisation
 - A little image processing: re-sampling, re-sizing, re-quantisation
- Colour spaces
 - RGB, HSV, LAB etc
 - Image pre-processing: choosing a colour space is a key step in practical applications, but not really IP
- Point transforms
 - Gain, bias, contrast stretching, gamma correction
 - Simplest methods, *but useful*

The End

- Spatial Filtering
 - Convolution is key
 - Noise removal: mean, Gaussian filtering
 - Enhancement: unsharp masking, Laplacian filtering
 - Edge detection: Roberts, Sobel, Marr-Hildreth, Canny
- Non-linear filtering
 - Median, anisotropic diffusion, bilateral filtering

*Linear and non-linear filters are at the heart of
the image processing toolbox*
- Thresholding and Binary Images
 - Otsu, Unimodal thresholding
 - Adaptive Thresholding
 - Connected components, morphology (erosion, dilation)

The End

- Histogram methods
 - Histogram equalisation
 - Comparing images: histogram intersection, histogram ratio
 - An application: image retrieval
- Frequency domain(not covered this year)
 - An overview, broad structure of frequency domain methods
 - Something you need to be aware of

The End

- Compression
 - *Increasingly important*
 - Types of redundancy: coding, spatial, psychovisual
 - Structure of compression systems
 - Components and complete schemes: Huffman coding, GIF, JPEG
- Segmentation and Line finding
 - Region growing, split and merge (quadtrees), watersheds
 - Hough transforms
- Convolutional Neural Network

A set of tools that can be used to create image processing pipelines

Primary Text Book

R.C. Gonzalez and R.E. Woods. (2018). [Digital Image Processing. \(Fourth Edition\)](#). Prentice Hall.

[Available in the Library]

For more details on each topic please refer to the priary text book for this module

Module Aims

- To introduce the fundamentals of digital image processing - theory and practice.
 - **Assessed by exam: how do techniques work, and what do they do**
- To gain practical experience in writing programs to manipulate digital images.
 - **Assessed by coursework – no coding in the exam**
- To lay the foundation for studying advanced topics in related fields.
 - **G53VIS next year?**

The Exam

- 1 hour exam
- Answer All question
- Focus on image processing methods
 - What they do
 - How they work
 - When they are appropriate
 - *Knowledge, Comprehension, Application*
 - Questions are (loosely) structured, topic area indicated

Read and answer the question

Take no. of marks available into account