

A dark blue background featuring a glowing, semi-transparent circular digital interface with concentric rings and data points. Overlaid on this is large, white, sans-serif text that reads "Image Processing and Computer Vision Fundamentals". The text is partially obscured by the digital interface, with some binary code visible at the bottom left and right edges.

# Image Processing and Computer Vision Fundamentals



## Some Areas of Interest

- Image representation, coding, and transmission
  - GIF, JPEG, Multimedia applications, HDTV
- Image enhancement
  - Making images easier for humans to interpret
  - sharpening and false colouring of medical images
- Image alteration
  - often for aesthetic reasons, but sometimes to meet scientific needs
  - image warping, morphing, correcting for geometrical distortions
- Image analysis
  - computer-based analysis of images
  - computer vision, image segmentation
  - deep learning



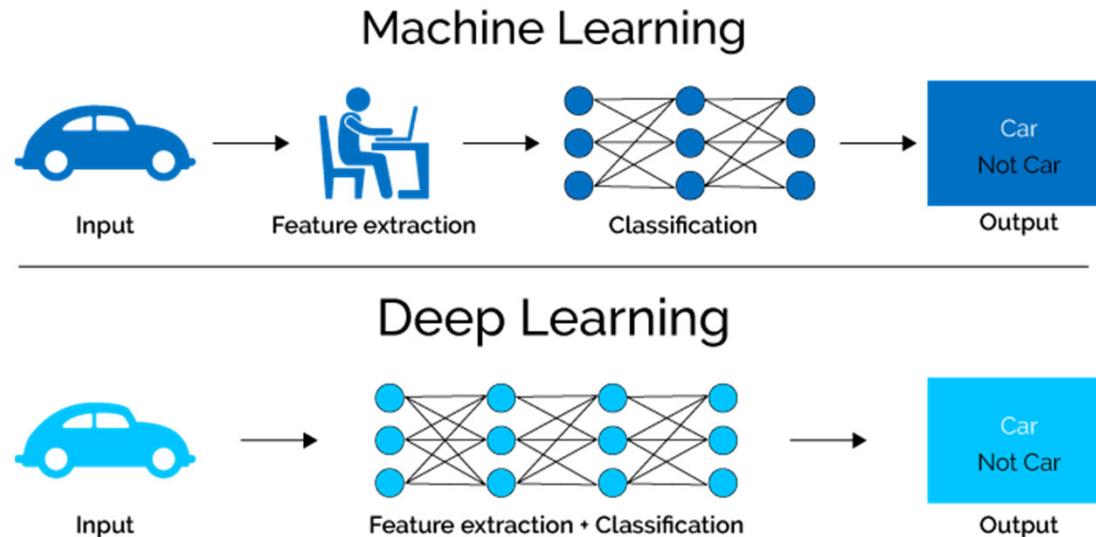
# What is Deep Learning (DL) ?

A machine learning subfield of learning **representations** of data. Exceptionally effective at **learning patterns**.

Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers of artificial neurons**

If you provide the system **lots of information**, it begins to understand it and respond in useful ways.

Why is it called deep? Because it has many layers.



<https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png>



# Example Problem: Number Plate Recognition

- Since humans can easily perform these tasks, sometimes the problems look deceptively simple



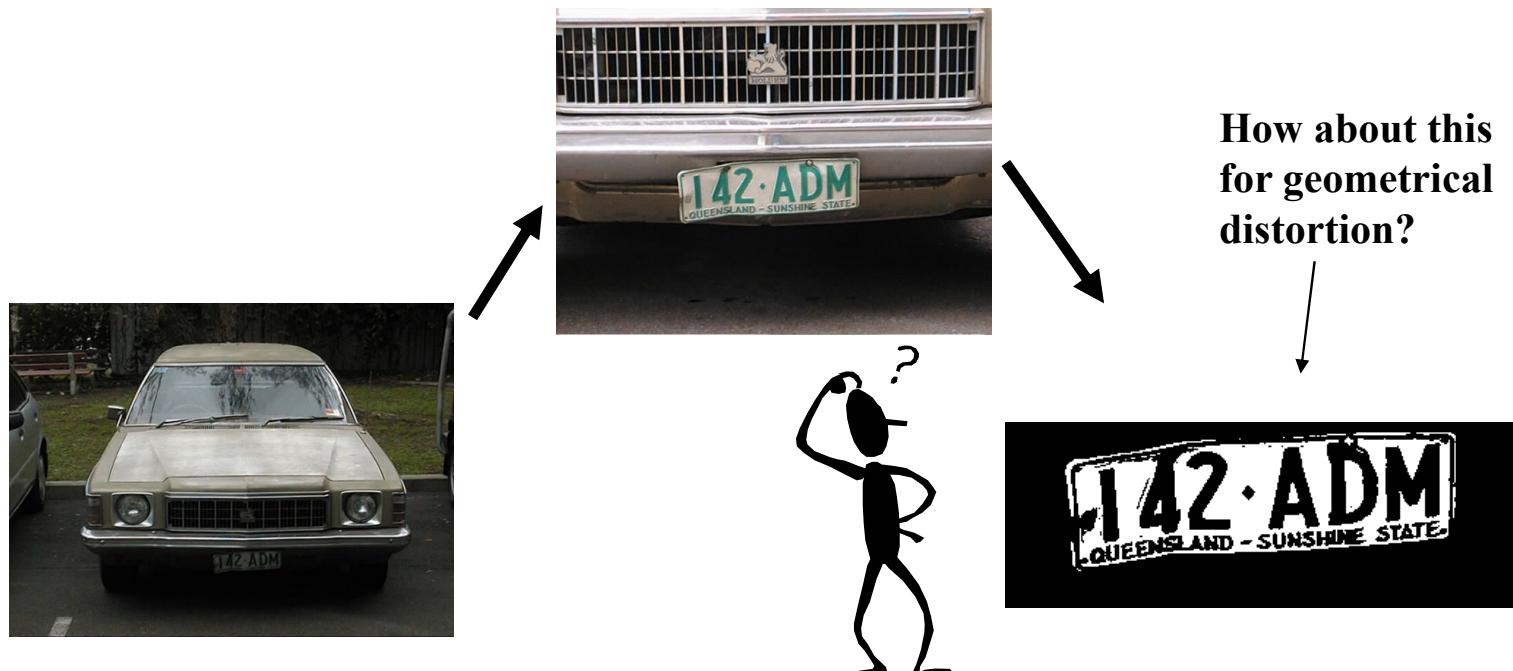
Still a bit noisy





## Example: Number Plate Recognition

- Robust real world computer vision has to be able to cope with much more complicated examples





## Much Harder Problem

- How do we recognize cats?
- Deep Learning comes to the rescue





# Why is Computer Vision Hard?

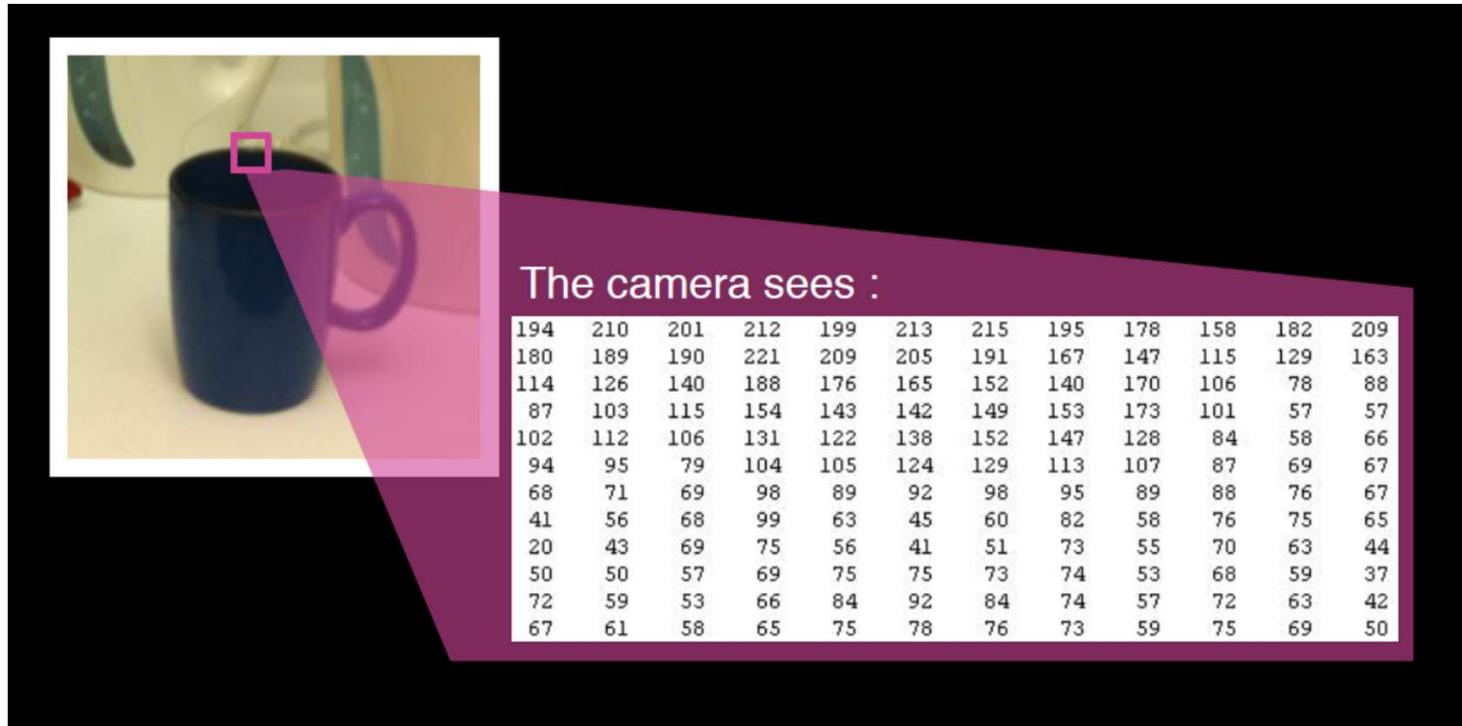
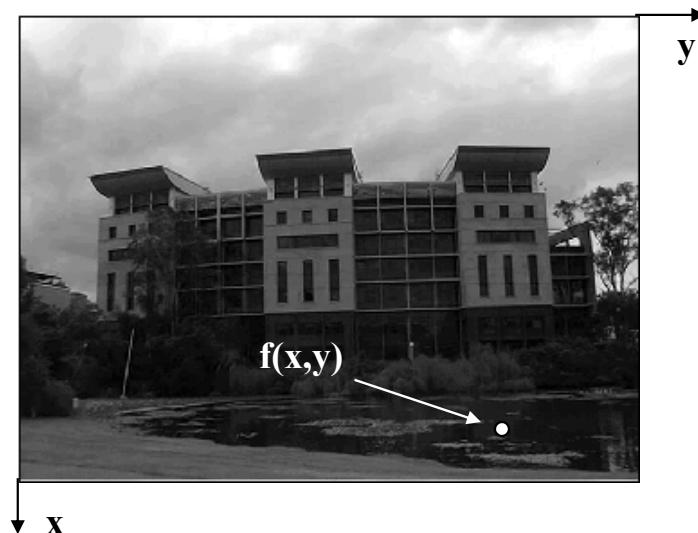


Image courtesy Andrew Ng CS229 Notes

# Image Representation

- A monochrome image or simply image, refers to a 2D light intensity function  $f(x,y)$  where  $x$  and  $y$  denote spatial coordinates and  $f$  is proportional to the brightness or grayscale (gray level) at that point.

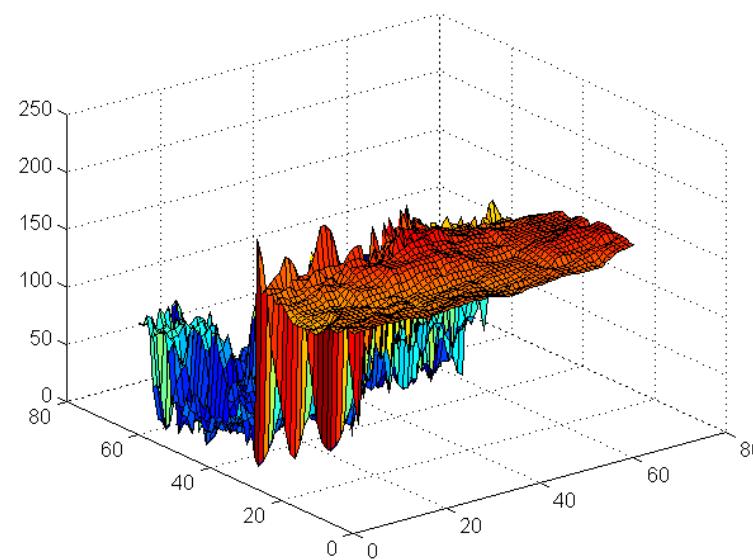


Note the direction  
of the axes!



# Image as a Surface

- Sometimes it is more useful to think of an image as a surface where gray level is represented by height





# Digital Images

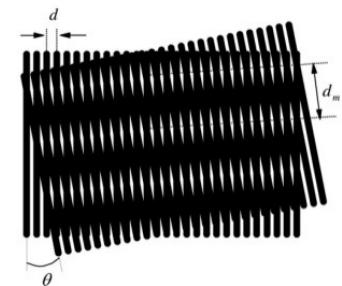
- A digital image is an image  $f(x,y)$  that has been discretized in both spatial coordinates and brightness
- The most common way to discretize spatial coordinates is regular sampling on a rectangular grid as is usually obtained from CCD and CMOS cameras.
- A digital image can be considered as a matrix where matrix element values represent grayscale
- Each element at a given spatial location of the digital image is called a pixel
- A typical size comparable in quality to a monochrome Analog TV image is 512 x 512 with 128 gray levels (7-bits). This is sometimes referred to as standard definition (SD) and is about 0.3 MP.
- Modern Digital TVs are 2MP (HD) or 4MP (4K).





# Digital Image Formats

- Binary Format,  $f \in \{0,1\}$ 
  - Intensity is indicated by just a 0 or 1
  - Good for representing text and line diagrams
  - Can be used to represent grayscale image data with half toning for printing and photocopying. Hard to scan due to Moire fringe effects!
  - Not easy to scale and resize due to aliasing effects
- Grayscale Format,  $f \in \mathbb{Z}^+$ 
  - Intensity is typically represented in 8 bits, 0..255
  - Good for representing monochrome pictures
  - may be used to represent colour images if each grayscale is mapped to a colour through a palette (pseudo-colour format)
  - relatively easy to scale images and text
- Colour Format,  $f \in (\mathbb{Z}^+)^3$ 
  - Intensity is typically represented by 3 8-bit numbers representing, say, red, green, and blue intensities.





# Examples



Original



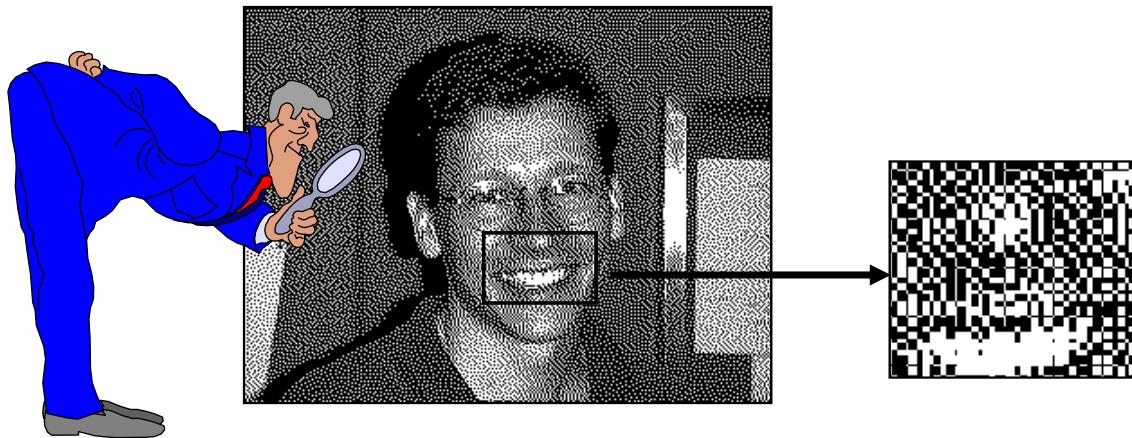
Binary  
Thresholded



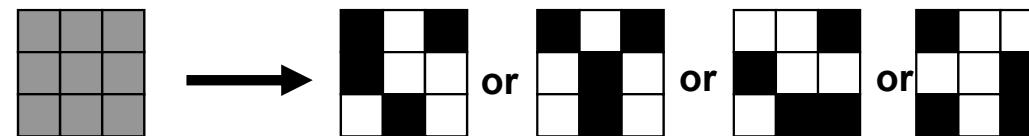
Binary  
Halftoned



# Half toning



**A block of grayscale pixels replaced by a block of binary pixels with the same average intensity.  
The binary pixels are randomized to reduce visible artifacts.**





# Binary File Formats

- Simple formats
  - ascii text: inefficient
  - raw: 1 bit per pixel written out a byte at a time
- Some standard formats
  - Portable bitmap format
    - pbm and pbmraw as above, with headers
    - part of netpbm utilities
    - uncompacted
  - TIFF
    - Tagged Image File Format, compacted, run length encoding
  - G3 Fax standard
    - Used in fax machines, compacted, run length encoding
  - JPEG
    - For photography, not so good for text

```
0101001  
1100101  
1010011
```



## Simple Image Compaction

- Ordinary images often contain runs of the same gray level over large areas. It suffices to precede the gray level by the run length or number of repeats.
- This is called run length encoding
  - Example: a grayscale image may contain the following sequence of gray levels
    - 003 004 004 004 008 008 006 007
  - Preceding each gray level with the **number of repeats** yields
    - **000 003 002 004 001 008 000 006 000 007**
  - This is actually a little longer in this example, but would usually be much shorter when there is a great deal of repetition



## TIFF Format

- Another format for RLE is the Tagged Image File Format (TIFF) where the run length is indicated by a preceding tag equal to 256 minus the number of repeats (as far as 127).
- Then to save too frequent repetition of 000, where a byte is not repeated, a tag in the range 1-127 indicated the number of following bytes which are to be read literally.
- Thus

003 004 004 004 008 008 006 007

becomes

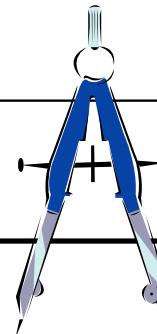
001 003 254 004 255 008 002 006 007

We still use TIFF for medical images  
but the files are very large.



# Perception

- We generally process visual information with our eyes and brain, but we can also use instruments. Often the two do not agree due to the different processing mechanisms.

Perceived Quantity	Physical Quantity
Brightness	Light Intensity
Colour	Light Frequency 



# Colour Image Processing

- Why use Colour?
  - Colour is a powerful descriptor that often simplifies object identification and extraction from a scene
  - Humans can discern thousands of colour shades and intensity compared to about two dozen shades of gray.
- Two major areas
  - True colour
    - needs a full colour sensor such as a colour TV camera
    - may be converted to a reduced colour palette (e.g., 256 colours for GIF images); sometimes called pseudo-colour
  - False colour
    - assigning a shade of colour to a particular grayscale or range of grayscale; for example colour CAT or MRI scan or X-ray baggage inspection.





# Examples



True Colour



Grayscale



False Colour



# Colour Spectrum

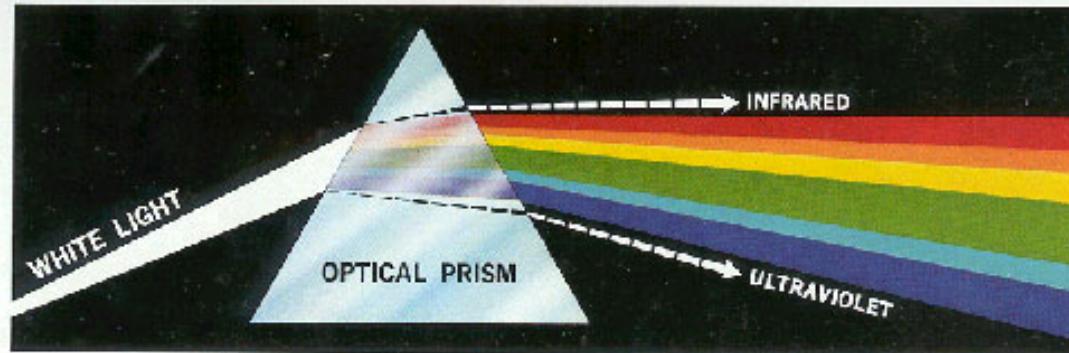
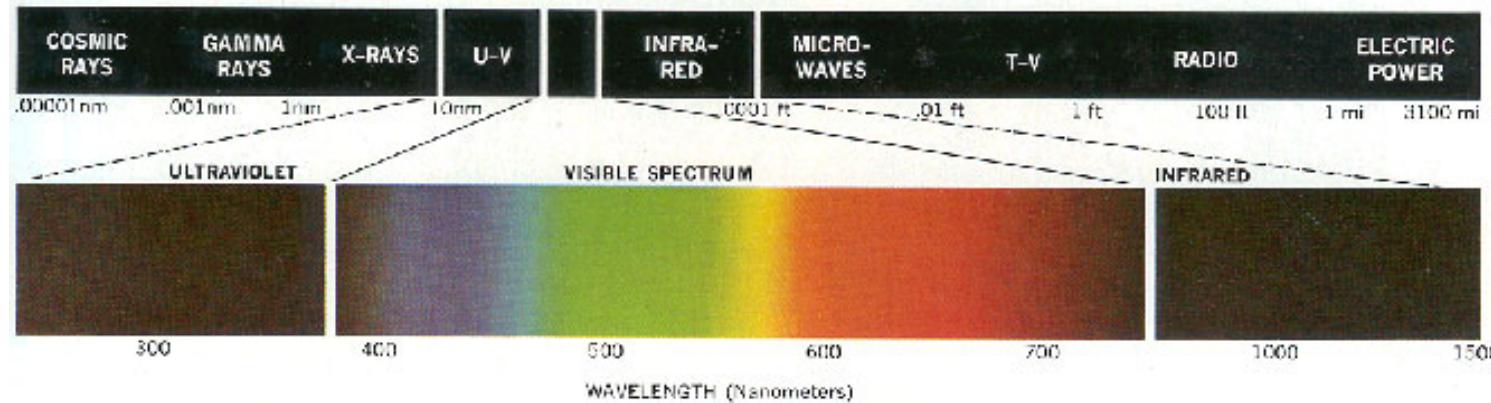
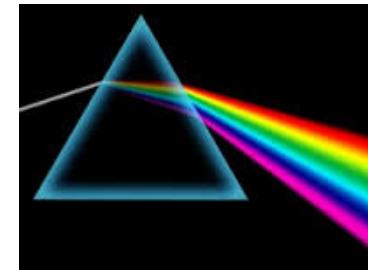


Plate I. Color spectrum seen by passing white light through a prism. (Courtesy of General Electric Co., Lamp Business Division.)





# Fundamentals



- When a beam of light is passed through a prism, the emerging light is not white; instead it is a continuous spectrum of colours ranging from red at one end to violet at the other.
- Light consists of electromagnetic radiation in the wavelength range of about 400 (violet) to 700 (red) nm.
- Colours that humans perceive are determined by the nature of the light reflected from the object
- A body that reflects light that is relatively balanced in all visible wavelengths appears *white* to the observer



## Fundamentals

- For example, green objects reflect light with wavelengths primarily in the 500 to 570 nm range.
- If light is achromatic (void of colour), its only attribute is its intensity or value. This is sort of light that is seen emanating from a B&W TV and is implicit in grayscale processing.
- Chromatic (coloured) light can be described in terms of radiance, luminance, and brightness



# Radiance, Luminance, and Brightness

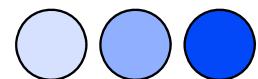
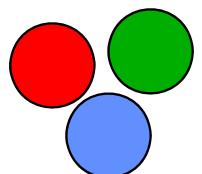
- Radiance is the total amount of energy that flows from the light source (usually expressed in Watts)
- Luminance (measured in lumens (lm)) gives a measure of the energy an observer perceives from a light source. For example, an infrared lamp could have significant radiance but it would be hardly visible; its luminance would be almost zero.
- Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of intensity and is one of the key factors in describing colour sensation.





# Brightness, Hue, and Saturation

- The perceptual attributes of colour are brightness, hue, and saturation.
- Brightness represents perceived luminance
- Hue refers to the redness, greenness etc of the colour. For monochromatic light sources, differences in hue are manifested by differences in wavelength.
- Saturation is the aspect of color that varies as white light is added to monochromatic light. Fully saturated colours are vivid, less saturated colours are described as washed-out or pastel.

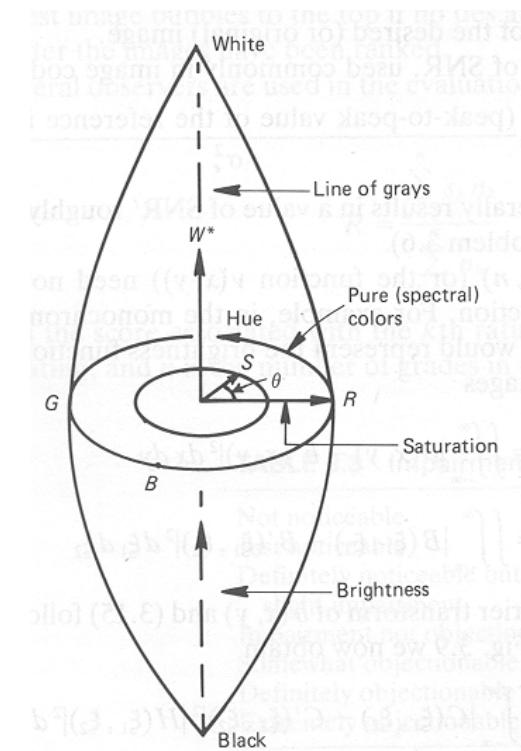


# Perceptual Representation of Colour Space

**The centre axis represents transitions from black to white through all the shades of gray**

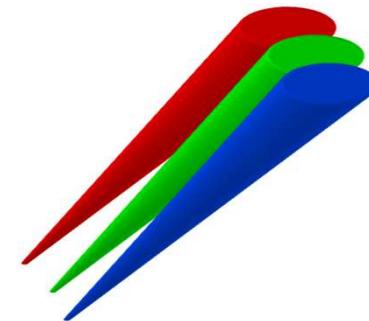
**As we go away from this axis, we have increasingly saturated colours**

**The angle of departure represents the hue**

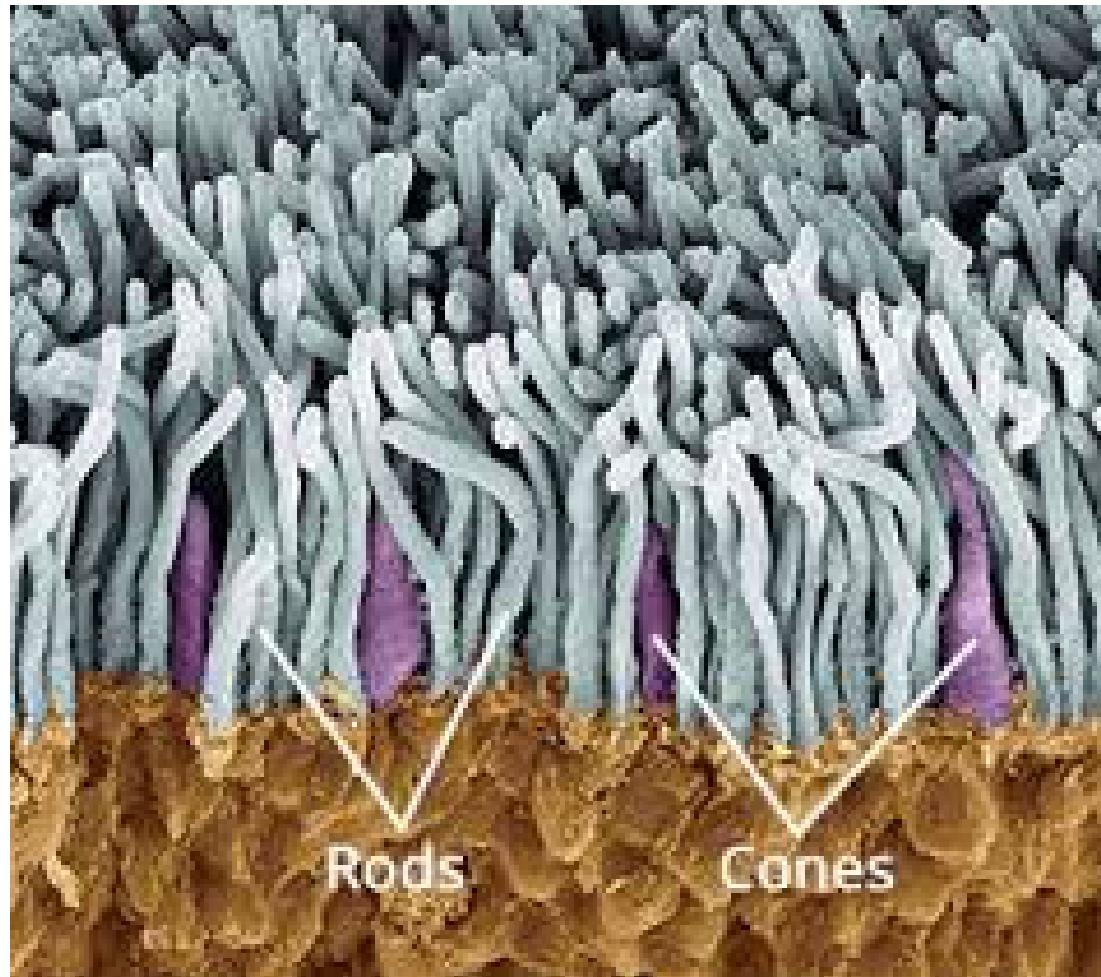




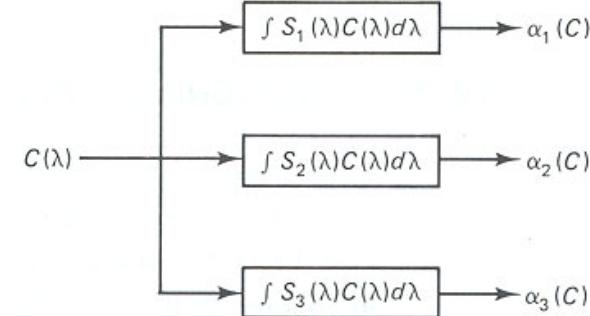
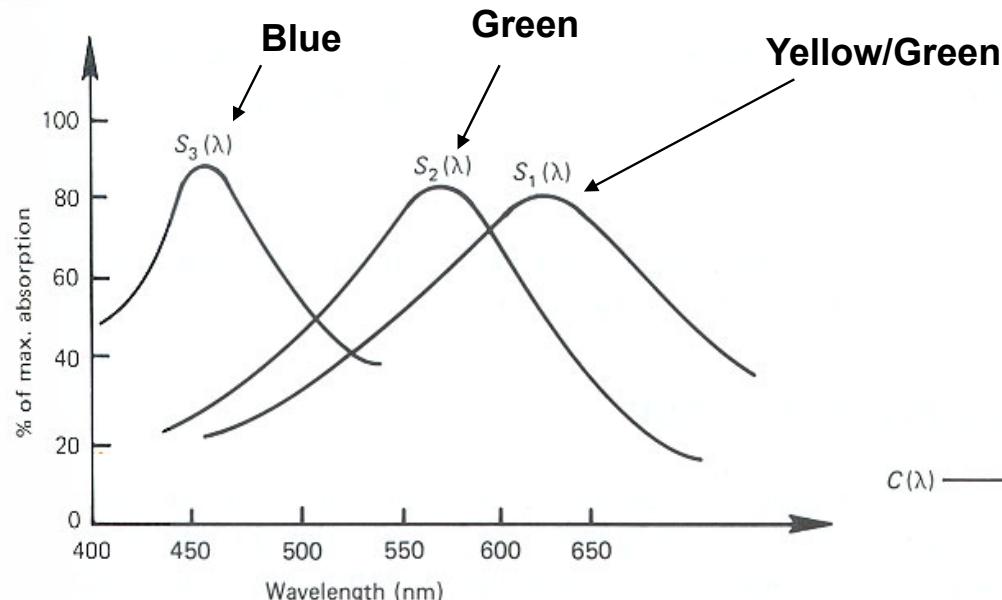
# Human Eye



- The eye has two types of light receptors: rods and cones
- The rods measure light intensity only and are used mainly for night and peripheral vision
- There are three types of cone receptors; each one sensitive to a different wavelength of light: red, green, and blue
- Thus the three cones form a type of light spectrum analyzer that is used to interpret incoming visible light in terms of perceived colour.



# Sensitivities of Cone Receptors



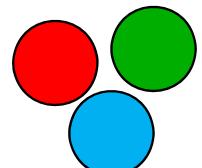
**Figure 3.11** (a) Typical absorption spectra of the three types of cones in the human retina; (b) three-receptor model for color representation.



# Primary Colours



- For the purpose of standardization, the CIE (Commission Internationale de l'Eclairage – International Commission on Illumination) assigned the following wavelengths to the three primary colours
  - red=700nm, green=546.nm, blue=435.nm
- Note that no single colour can be called red, green, or blue. These attributes apply to a range of wavelengths
- By mixing the light from three primary colour light sources, we can create a very wide range of colours – but not all possible colours. This is an example of the *additive* primary colour model as used in colour TV sets and computer monitors.



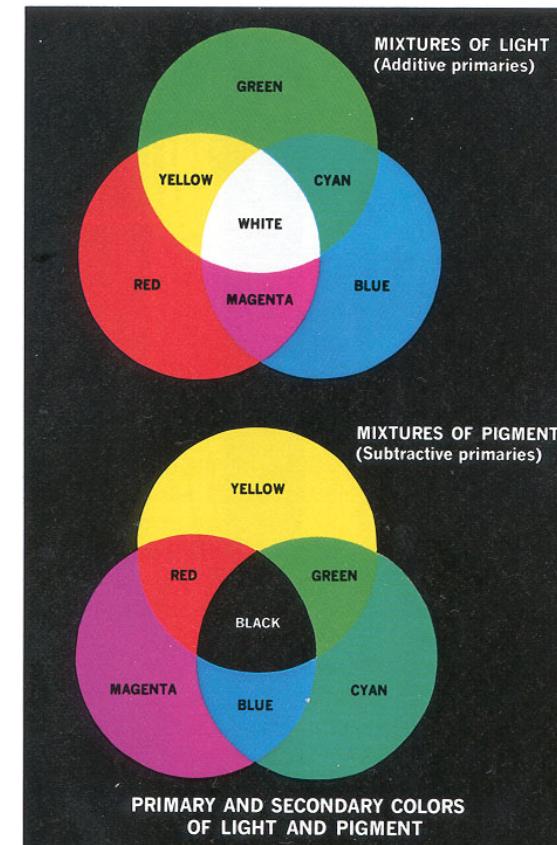


# Additive and Subtractive Models

If we add the primary colours pair wise, we get the colours cyan, yellow, and magenta.

These are called the subtractive primary colours which is the model required for mixing paint and pigment.

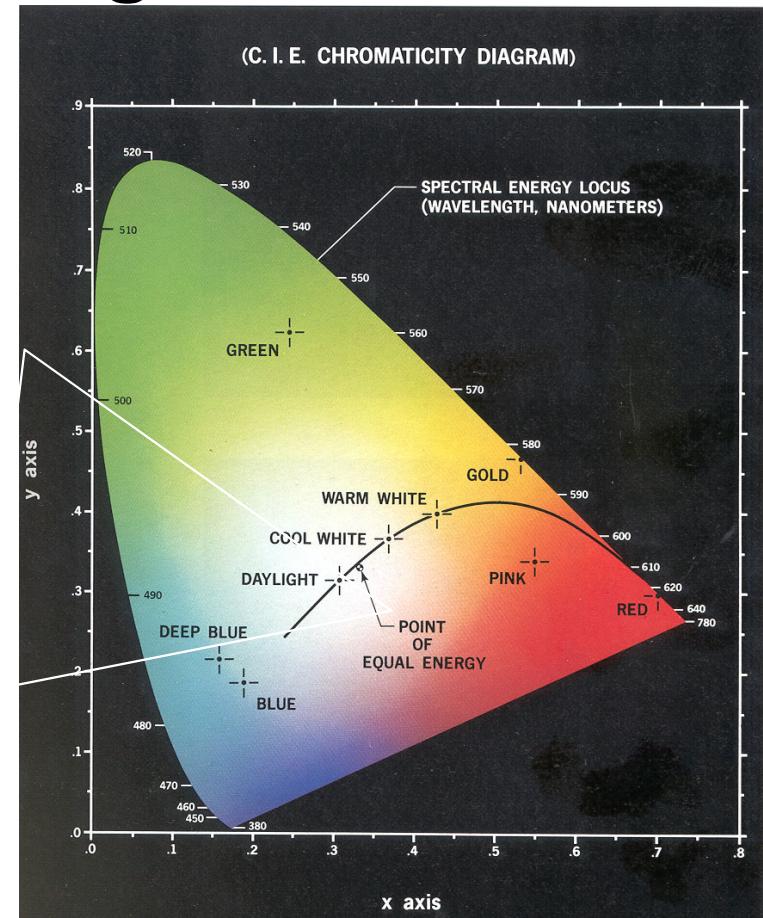
Thus many printers use the CYMK model where CYM stands for the subtractive primaries and K stands for black ink.





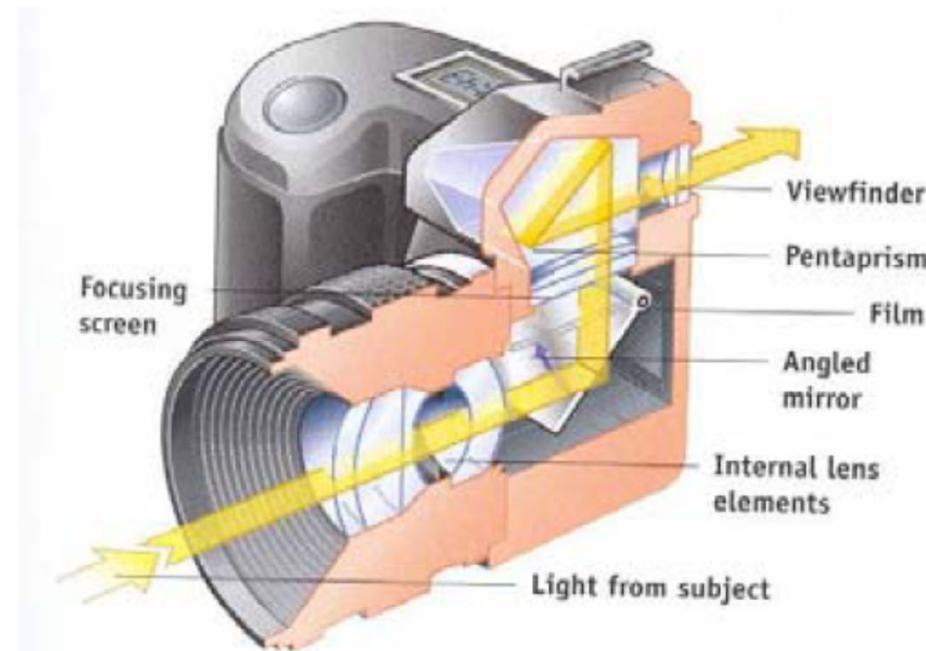
# CIE Chromaticity Diagram

**Only colours within the triangle can be represented by mixing the three primary colours red, green, and blue. Some computer monitors have 4 phosphors to improve the colour gamut.**





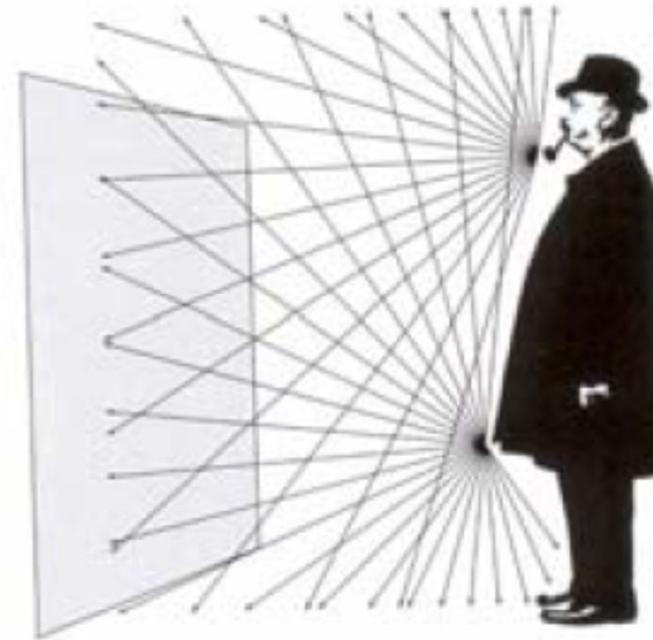
# Image Formation





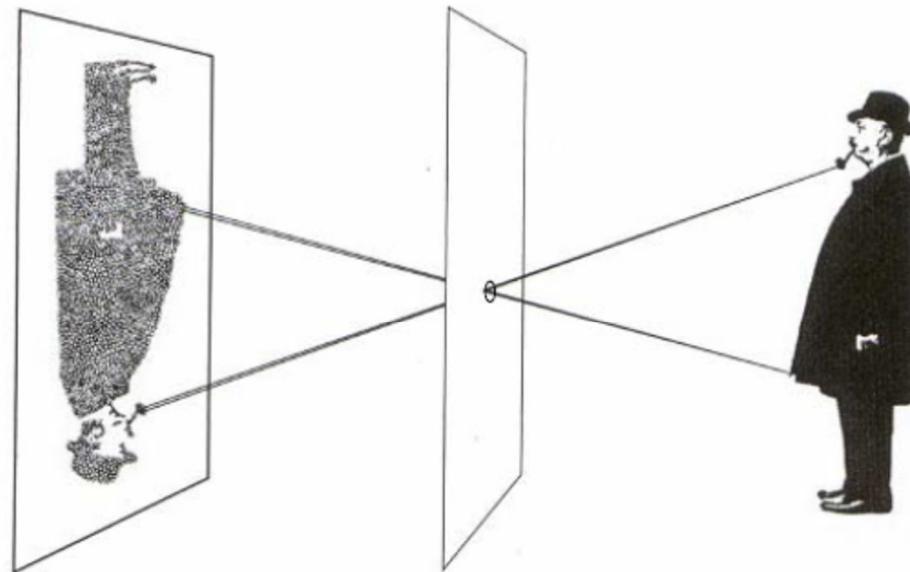
# Why is there no image on a White Piece of Paper?

**It receives Light from all directions.**





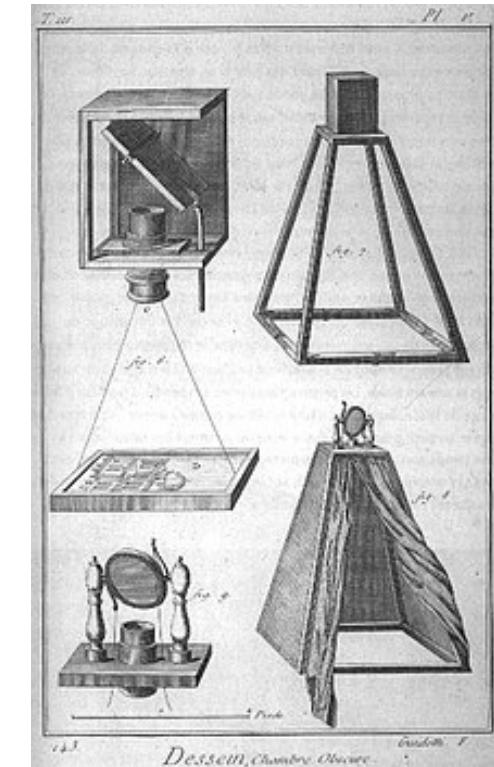
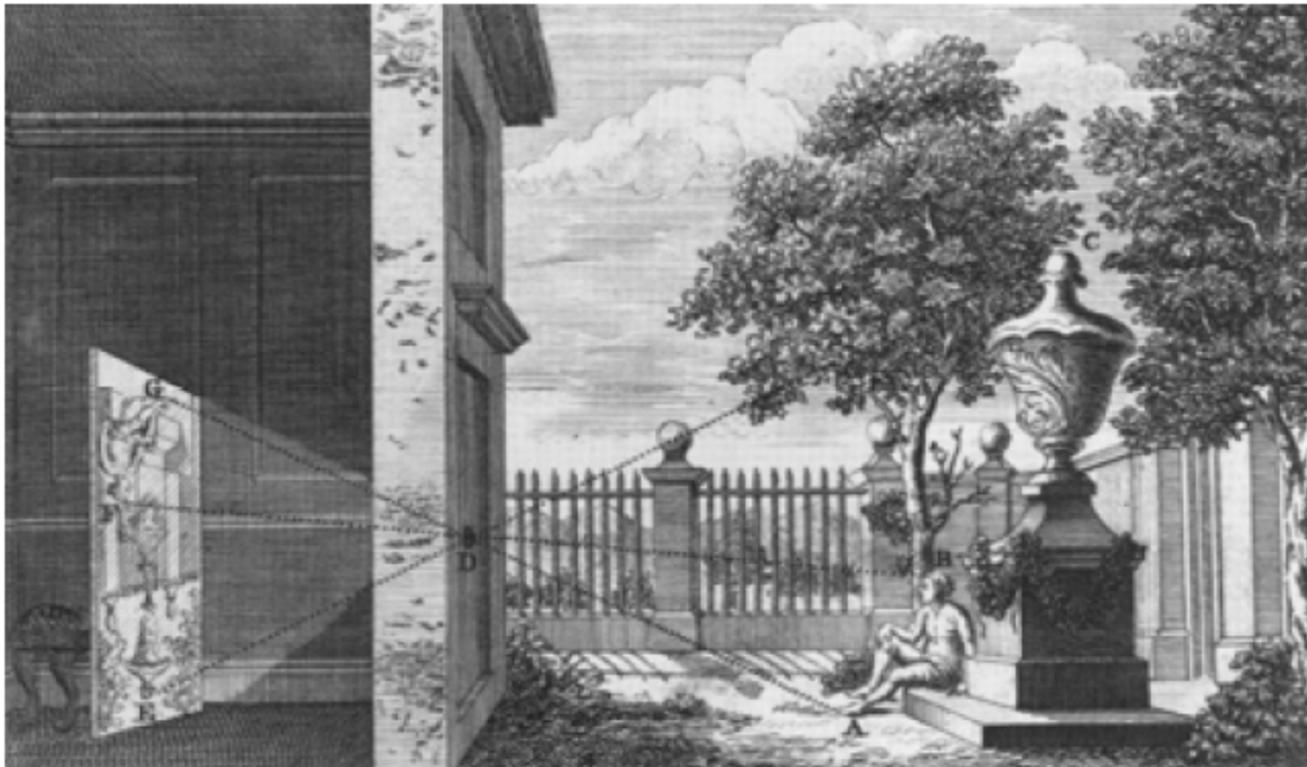
# Pinhole Image Formation



From Photography, London et al.



# Camera Obscura

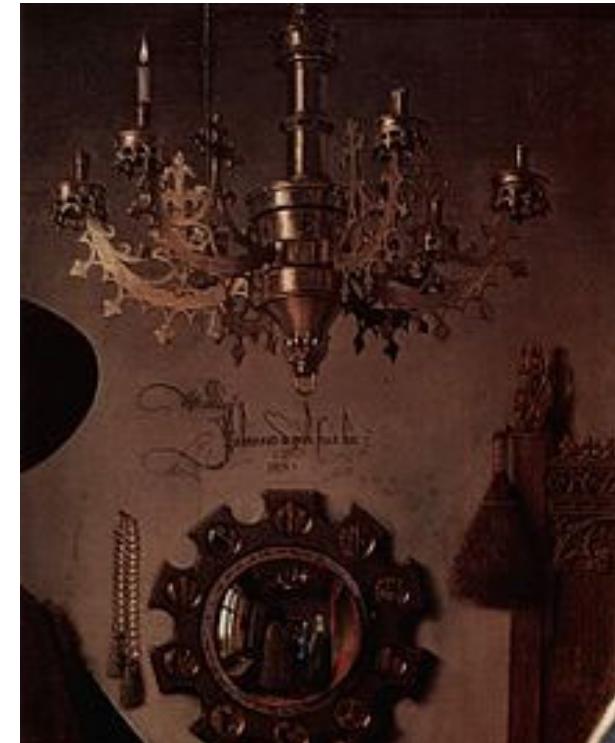




## Can Also use Concave Mirrors

This could explain sudden increase in realism paintings from about 1420. Photographic techniques may have guided these artists.

Examples of out-of-focus paintings  
Also left-right reversed paintings  
Many portraits of left-handed people

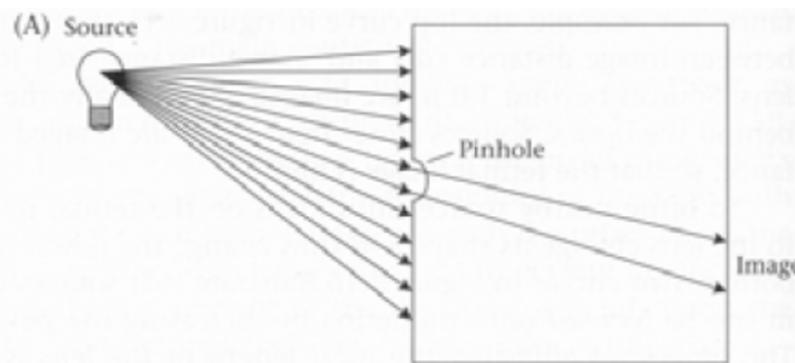


[https://en.wikipedia.org/wiki/Hockney%E2%80%93Falco\\_thesis](https://en.wikipedia.org/wiki/Hockney%E2%80%93Falco_thesis)

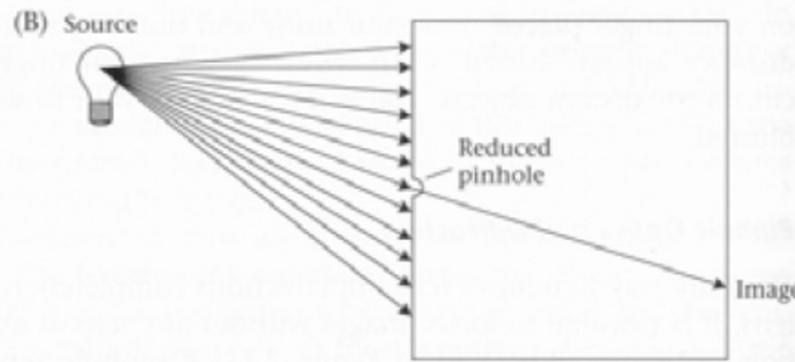
Detail of the chandelier and mirror from Jan van Eyck's 1434 *Arnolfini Portrait*, one of Hockney's key examples



# Pinhole Size



**Blurred but more light**



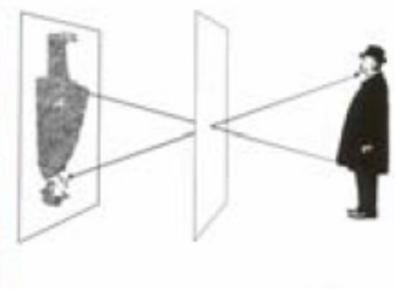
**Clear but very dark**

Wandell, Foundations of Vision, Sinauer, 1995

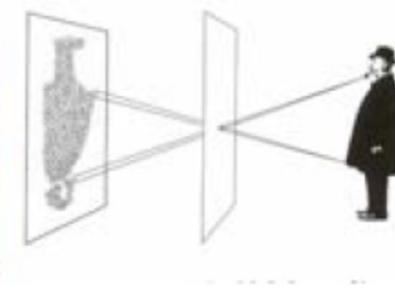


# Pinhole Size?

Photograph made with small pinhole



Photograph made with larger pinhole

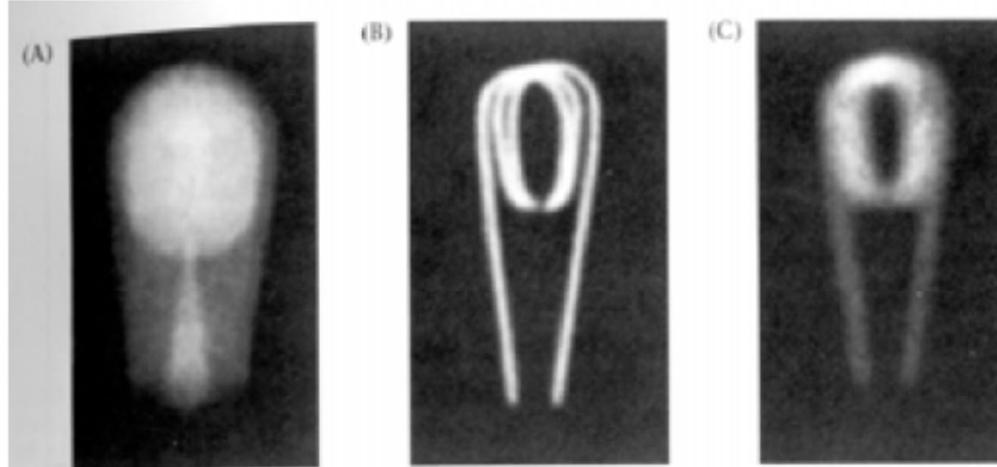


From Photography, London et al.



# Diffraction Limit

- Optimal size for visible light:
  - $\text{sqrt}(f)/28$  (in millimeters) where  $f$  is focal length



**2.18 DIFFRACTION LIMITS THE QUALITY OF PINHOLE OPTICS.** These three images of a bulb filament were made using pinholes with decreasing size. (A) When the pinhole is relatively large, the image rays are not properly converged, and the image is blurred. (B) Reducing the size of the pinhole improves the focus. (C) Reducing the size of the pinhole further worsens the focus, due to diffraction. From Ruechardt, 1958.

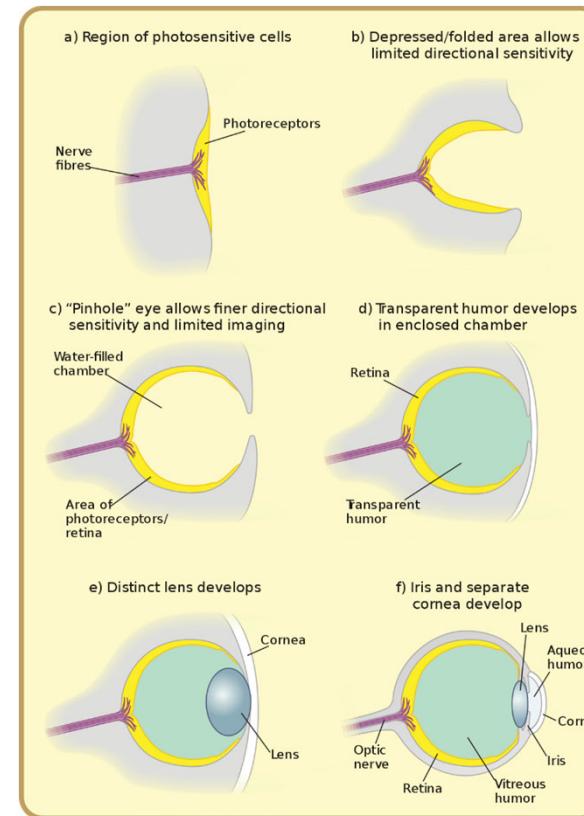
---

From Wandell



# Evolution of the Eye

Some people claim they the eye is so complex that it could not have arisen by chance. This image suggests the various stages in the evolution of the eye. The development of vision caused an explosion in Life on Earth in 537 million years BC.





## Problems with Pinhole

- Not enough light
  - Requires long exposure and leads to motion blur
- Diffraction limits sharpness of images

### Solution

- Refraction and Reflection
- Refraction is responsible for image formation by lenses and the eye
- Reflection by mirrors is used in large telescopes



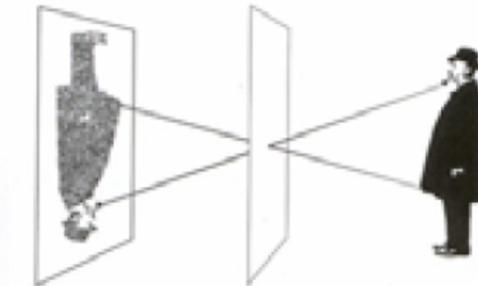
# Lenses

- Gathers more light
- But needs to be focussed

Photograph made with small pinhole

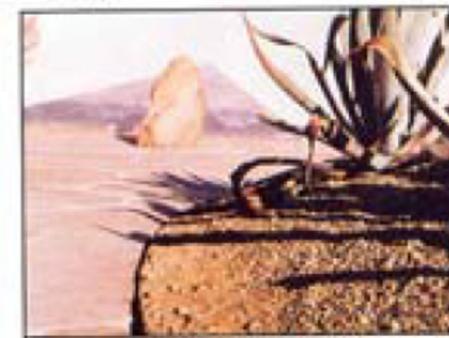


To make this picture, the lens of a camera was replaced with a thin metal disk pierced by a tiny pinhole, equivalent in size to an aperture of f/182. Only a few rays of light from each point on the



subject got through the tiny opening, producing a soft but acceptably clear photograph. Because of the small size of the pinhole, the exposure had to be 6 sec long.

Photograph made with lens



This time, using a simple convex lens with an f/16 aperture, the scene appeared sharper than the one taken with the smaller pinhole, and the exposure time was much shorter; only 1/100 sec.

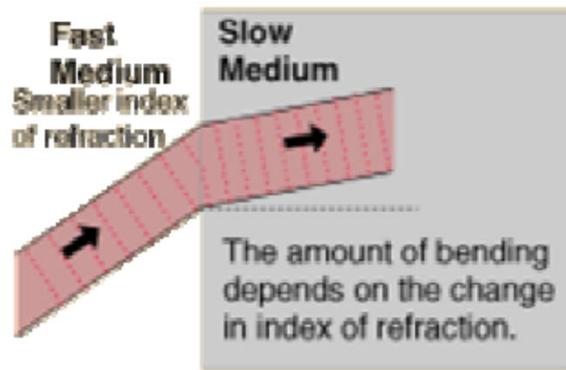


The lens opening was much bigger than the pinhole, letting in far more light, but it focused the rays from each point on the subject precisely so that they were sharp on the film.

From Photography, London et al.



# Refraction



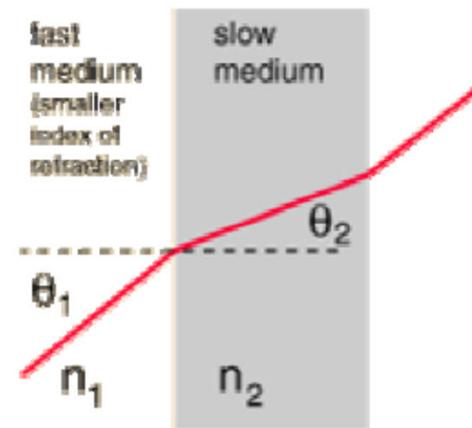
- Refraction is the bending of a wave when it enters a medium where its speed is different.
- The refraction of light when it passes from a fast medium to a slow medium bends the light ray toward the normal to the boundary between the two media.
- The amount of bending depends on the indices of refraction of the two media and is described quantitatively by Snell's Law.



# Snell's Law

Snell's Law

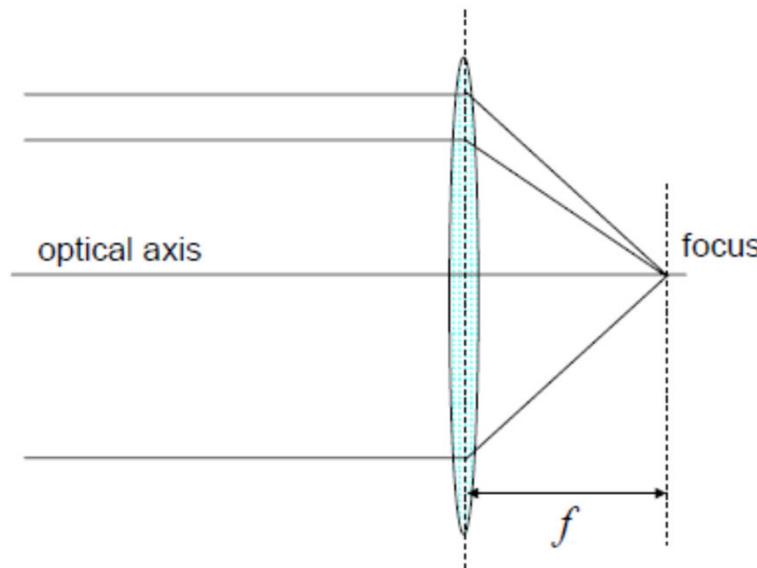
$$\frac{n_1}{n_2} = \frac{\sin \theta_2}{\sin \theta_1}$$



$n_i$  -- is the index of refraction which is defined as the speed of light in vacuum divided by the speed of light in the medium.



# Thin Lens



**Gathers more light  
Image plane needs to  
Placed at focal plane  
For optimal sharpness**

Spherical lens surface: Parallel rays are refracted to single point



# Cameras and Computer Vision

- Computer vision works best with clear images
- We encounter issues due to
  - Low light grain
  - Motion blur due to long exposure
  - Deformation due to rolling shutter in CMOS devices
  - Codec issues JPEG and H265 artefacts
  - Depth of field and focussing issues
  - Low resolution of small objects
- Some of these problems are best solved by getting a better camera



# Some Image Capture Issues

**Motion Blur**



**Low Light Grain**



Grainy Image

**Rolling Shutter Distortion**





## H264 CCTV Artefacts



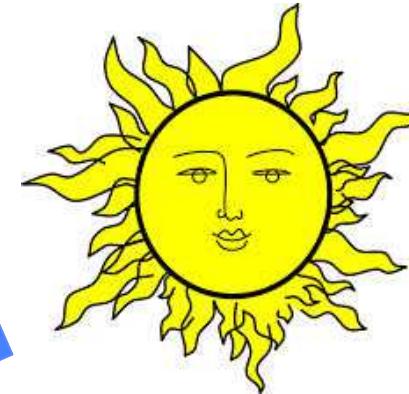
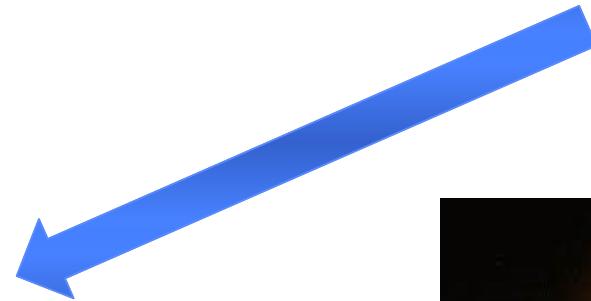
7/7/2005 London Tube Bombers



Fence in front of Subject!!



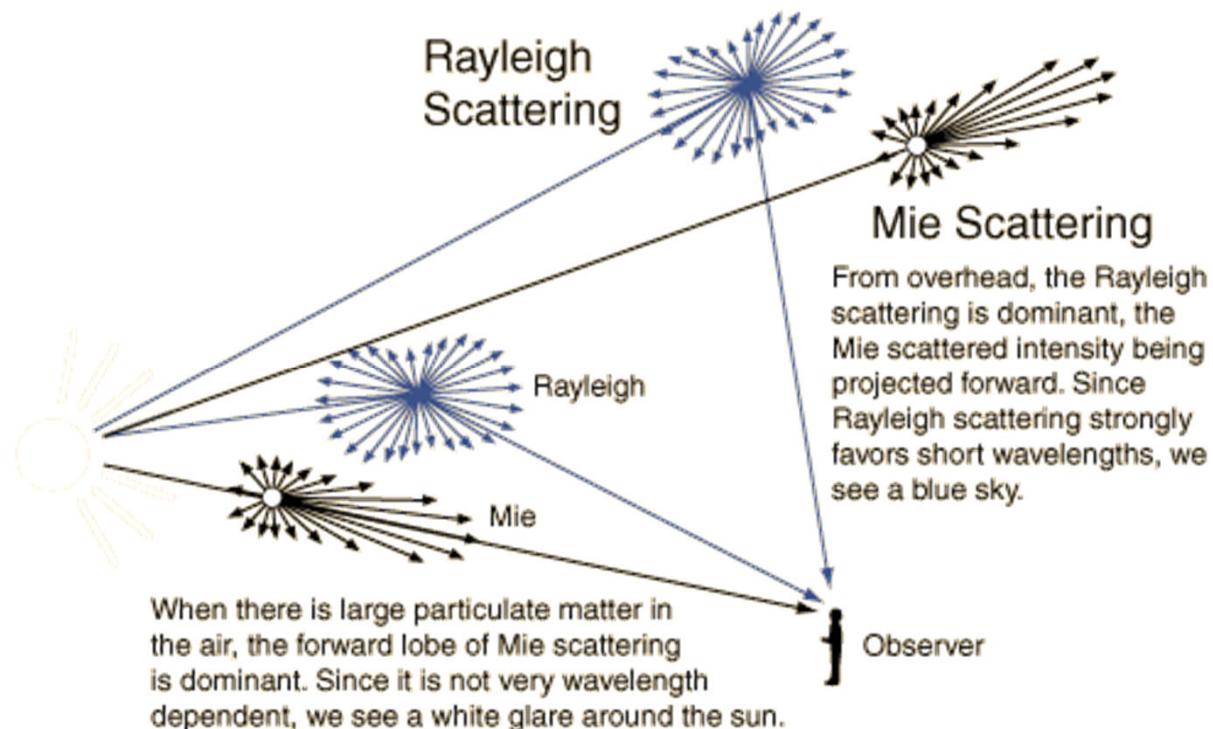
## Why is the Sky Blue?



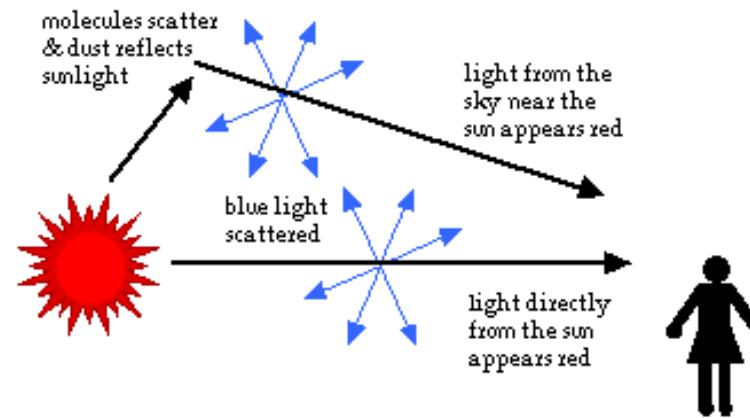
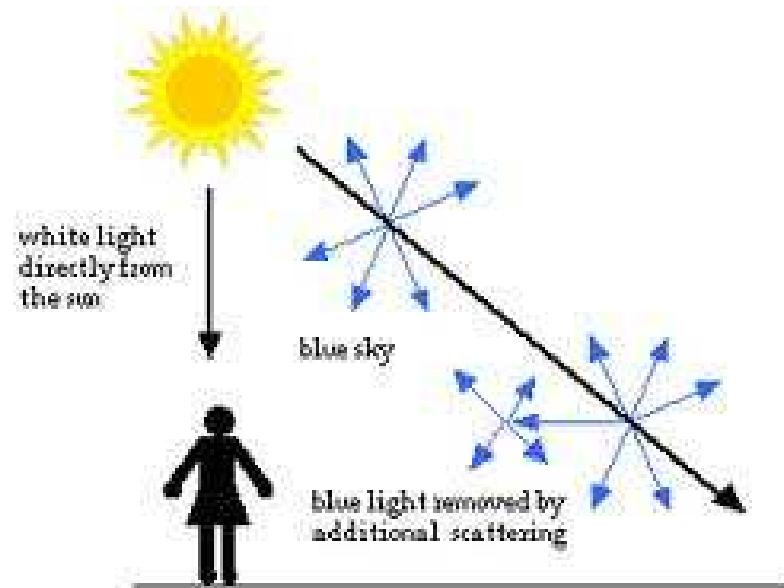
**From space it looks like this**



## Rayleigh Scattering from Atmosphere



<http://hyperphysics.phy-astr.gsu.edu/hbase/atmos/blusky.html>



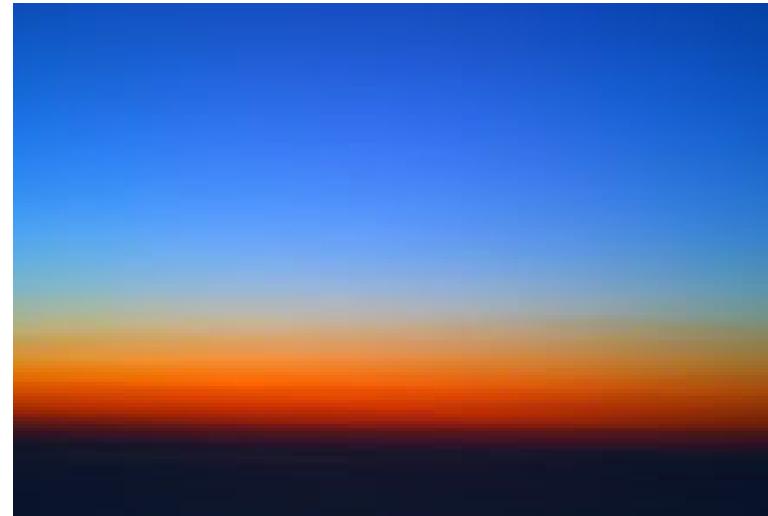
<http://macaulay.cuny.edu/eportfolios/sciencefordessert/2010/09/29/why-is-the-sky-blue-and-the-sunset-red/>



## Evening Sky



Before Sunset



After Sunset



## Light Field Cameras or Plenoptic Cameras



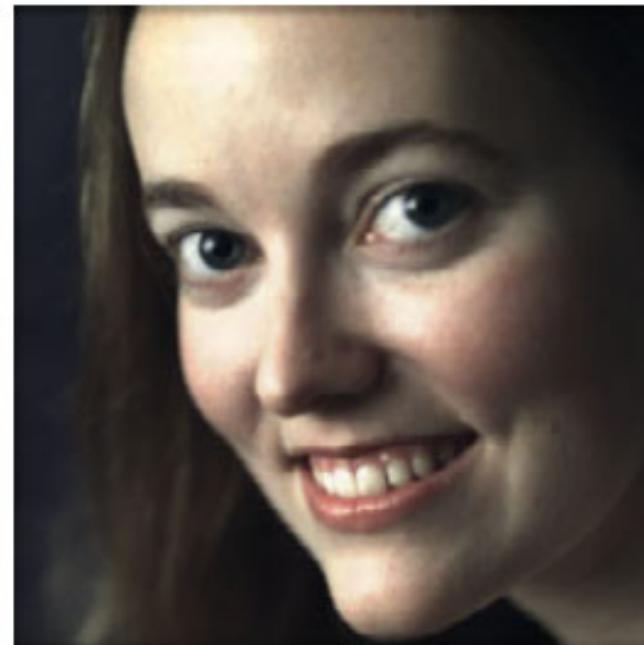
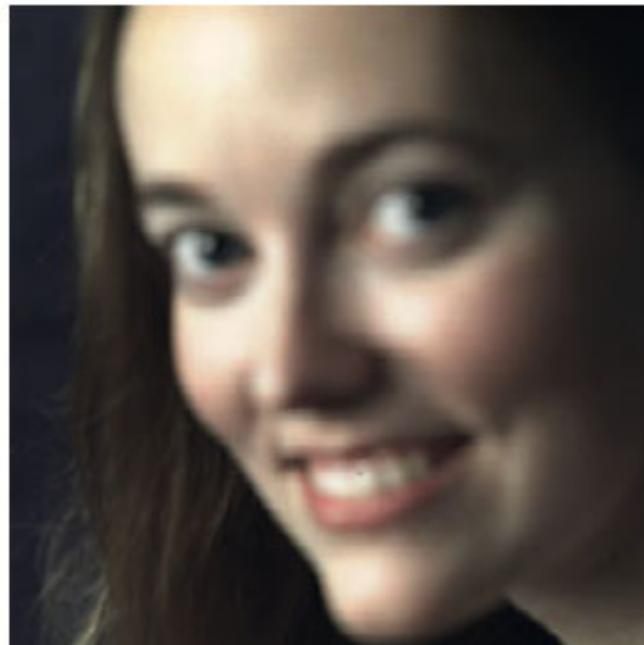
R. Ng, [M. Levoy](#), M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan. [Light Field Photography with a Hand-Held Plenoptic Camera](#). Stanford University Computer Science Tech Report CSTR 2005-02, April 2005

Lytro was founded in 2006 by Executive Chairman Ren Ng, whose Ph.D. research on light field photography won Stanford University's prize for best thesis in computer science in 2006 as well as the internationally recognized ACM Dissertation award.

In 2012, Lytro released the world's first consumer Light Field Camera which offers photographic capabilities never before possible, such as focusing a picture after it's taken, changing the perspective in the picture and creating interactive living pictures that can be endlessly refocused and enjoyed by friends and family online.



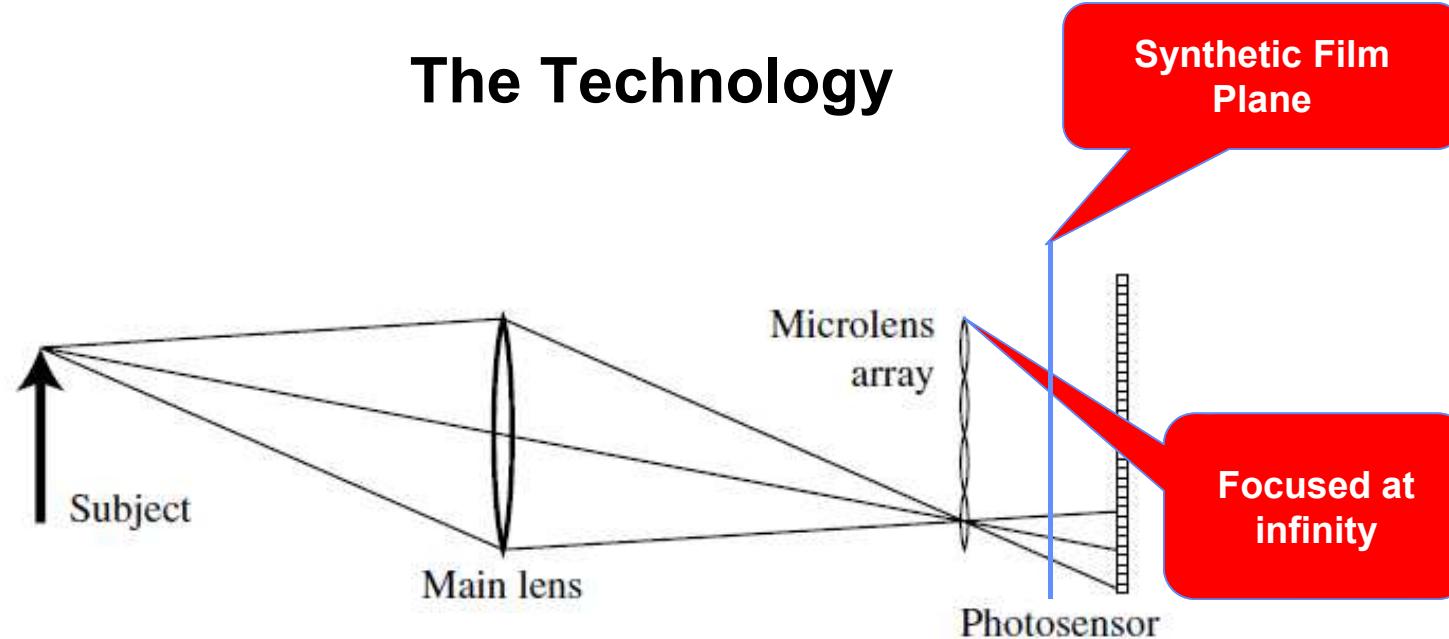
## Refocus after Photo Taken



**Figure 16:** Refocusing of a portrait. Left shows what the conventional photo would have looked like (autofocus mis-focused by only 10 cm on the girl's hair). Right shows the refocused photograph.



## The Technology

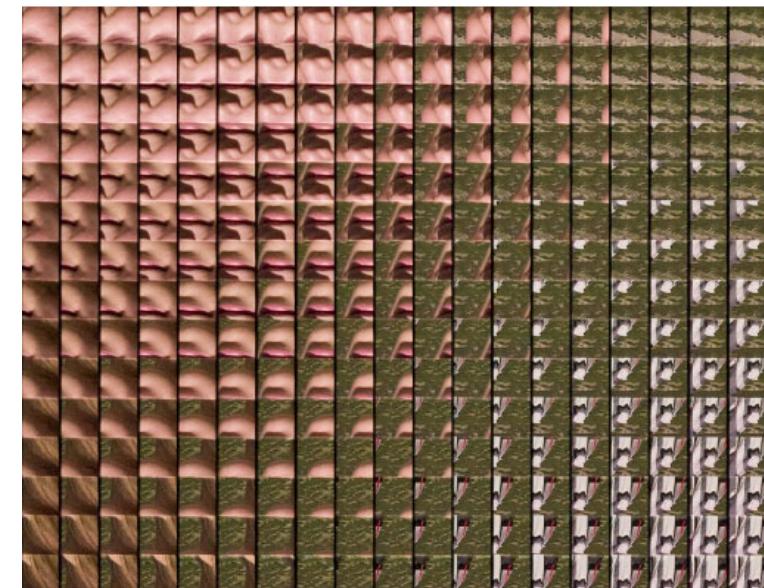
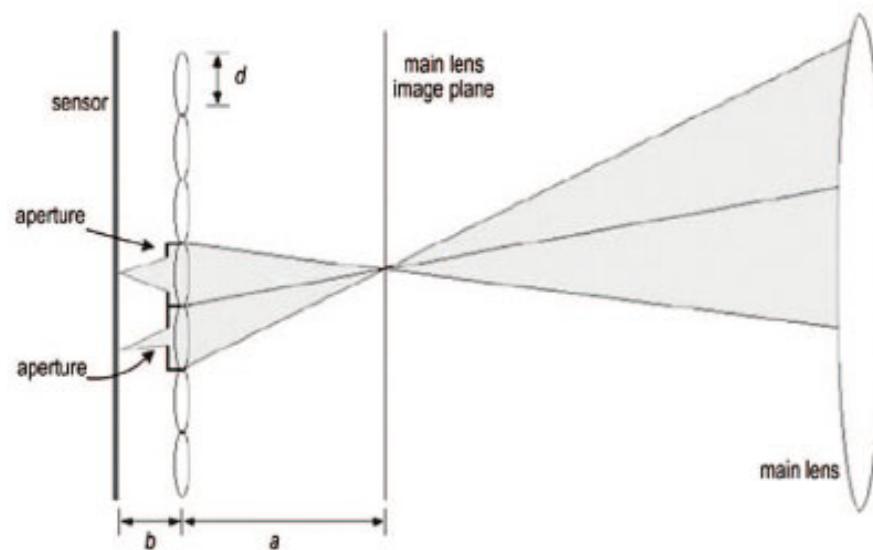


**Figure 1:** Conceptual schematic (not drawn to scale) of our camera, which is composed of a main lens, microlens array and a photosensor. The main lens focuses the subject onto the microlens array. The microlens array separates the converging rays into an image on the photosensor behind it.



# Refocussing

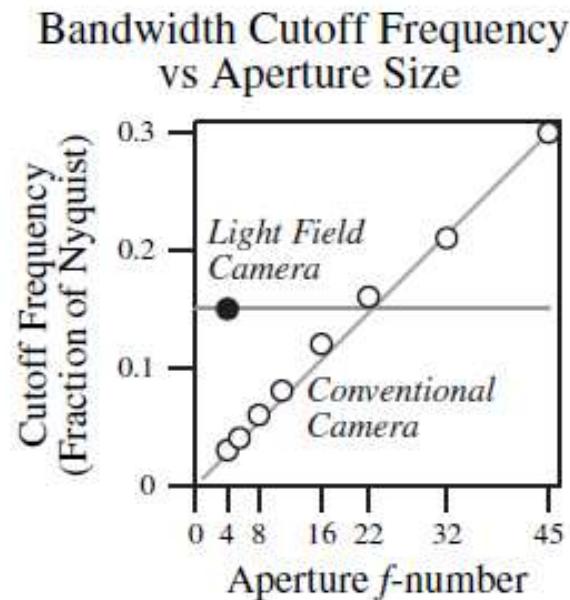
- Synthesize a film plane by shifting and adding subaperture windows
- This also reduces sensor noise

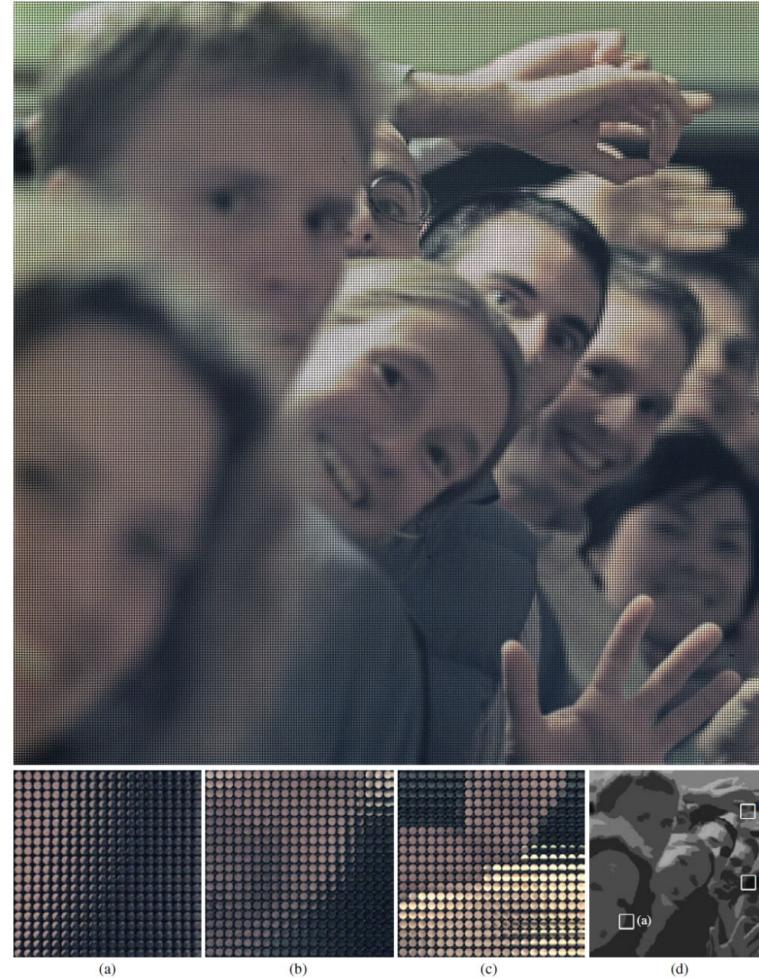




## Comparison to Standard Camera

**Figure 11:** Comparison of bandwidth cutoff frequencies for light field camera with refocusing versus conventional cameras with stopped down apertures. We choose the cutoff frequency as the minimum that contains 75% of the transfer function energy. Note that with this criterion the light field camera most closely matches the  $f/22$  conventional camera.

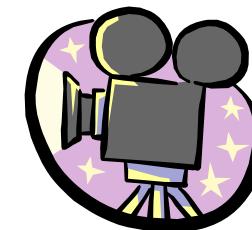






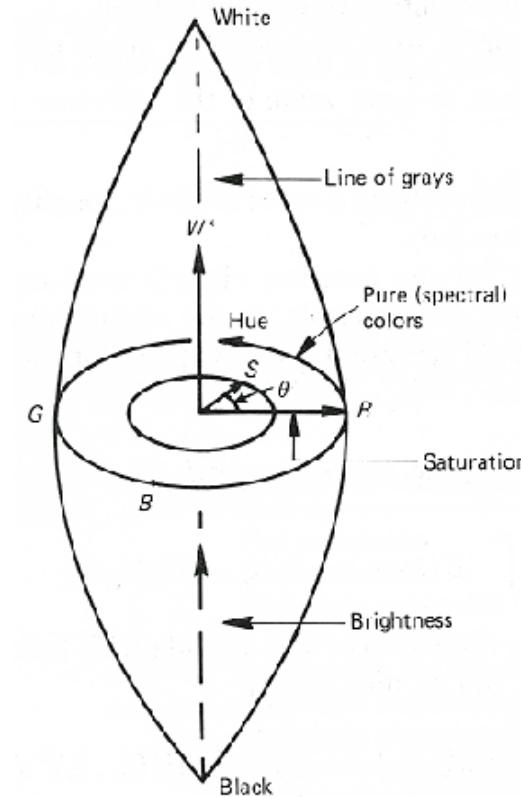
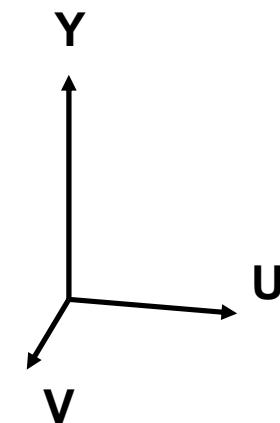
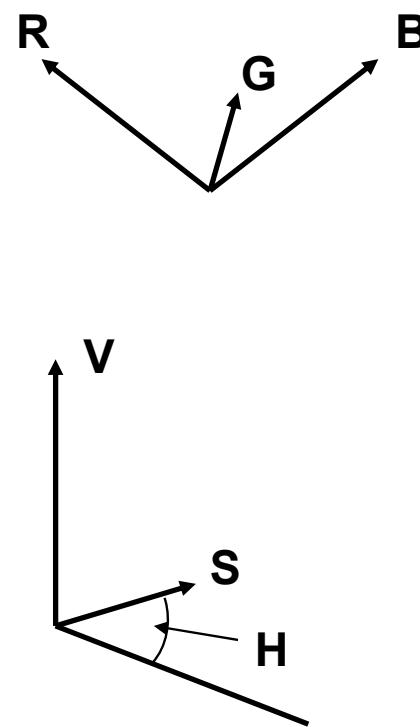
## Colour Coordinate Systems

- RGB
  - component colours, used in computer monitors and cathode ray tubes
  - Cartesian coordinate system
- YIQ or YUV
  - composite colour system used in PAL/NTSC colour TV transmission
  - Y denotes intensity (B&W TV signal), UV are Cartesian coordinates specifying hue and saturation
- HSV or HSI
  - similar to YUV except expressed in polar coordinates
  - H denotes hue, S denotes saturation, V (I) denotes value (intensity)
- CYMK
  - subtractive primaries for printing plus black(K)





# Colour Space



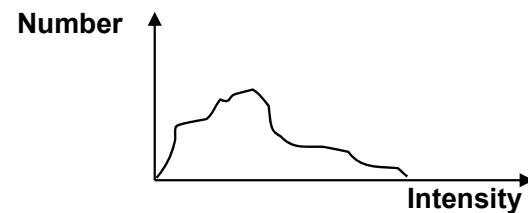


# Image Enhancement

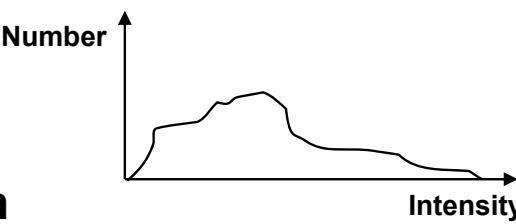
- Operations such as image adjustment for under and overexposure are best done in, say, HSV space rather than RGB space



**RGB Equalization**



**HSV  
Histogram  
Equalization**





## The Colour Constancy Problem

Colour constancy is an example of subjective constancy and a feature of the human color perception system which ensures that the perceived color of objects remains relatively constant under varying illumination conditions. A green apple for instance looks green to us at midday, when the main illumination is white sunlight, and also at sunset, when the main illumination is red. This helps us identify objects.

Cameras use white balance and other mechanisms to make pictures look right.





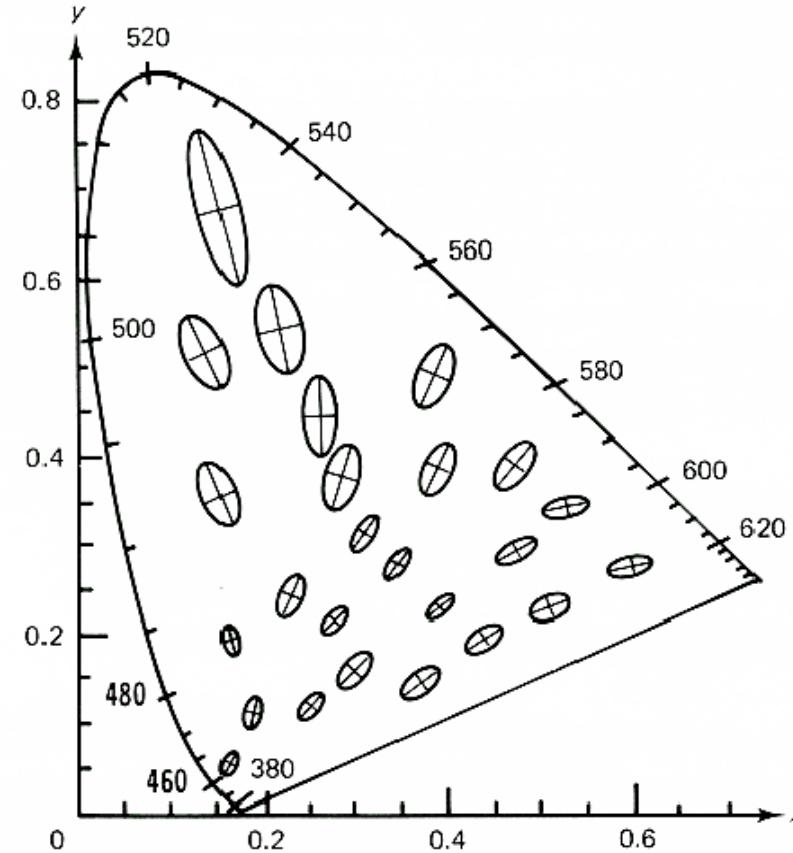
## Colour Matching

- If we wish to use these colour coordinates to compare colours we have a problem.
  - For some colours, very small changes in RGB space, say, can yield very noticeable differences
  - For other colours, large changes in RGB values cause very little colour change
  - In other words, these colour coordinate systems are perceptually non-linear
  - What we need is a warping of these spaces so that Euclidean distances correspond to colour differences
  - Use *CIE perceptually linear colour space*, sometimes called the *uniform chromaticity scale* (UCS)



## Just Noticeable Differences

This nonlinearity can be shown by plotting JND (just noticeable difference) ellipses on the chromaticity diagram.





## Conversion of YUV to UCS

Colour Distance →  $(\Delta s)^2 = (\Delta L^*)^2 + (\Delta u^*)^2 + (\Delta v^*)^2$

$$L^* = 25 \left( \frac{100Y}{Y_0} \right)^{\frac{1}{3}} - 16$$

$$u^* = 13L^*(u' - u_0); v^* = 13L^*(v' - v_0)$$

$$u' = u; v' = 1.5v$$

**$u_0, v_0, Y_0$  corresponds to reference white   CIE 1976**



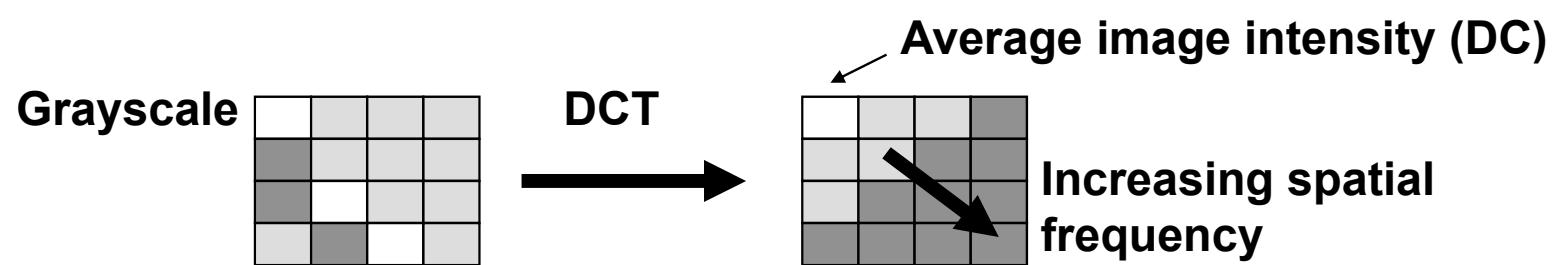
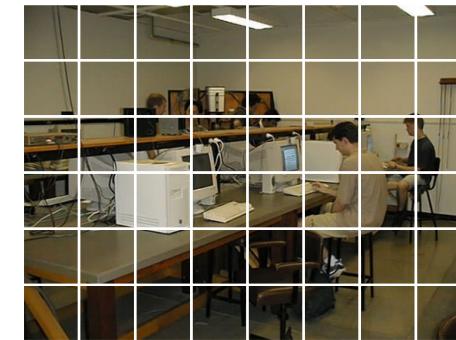
## Some File Formats

- PPM
  - portable pixel map, 8/24 bit colour, uncompacted (big files)
- BMP
  - windows bitmap, uncompacted
- Compuserve GIF (Generic Interchange Format)
  - 256 (8 bit) colours only, compacted (lossless compression)
- TIFF
  - 8/24 bit colour, compacted
- **JPEG (Joint Photographic Experts Group)**
  - 8/24 bit colour, usually compressed (lossy)
  - Usually best for photographs, very small files



## JPEG Encoding

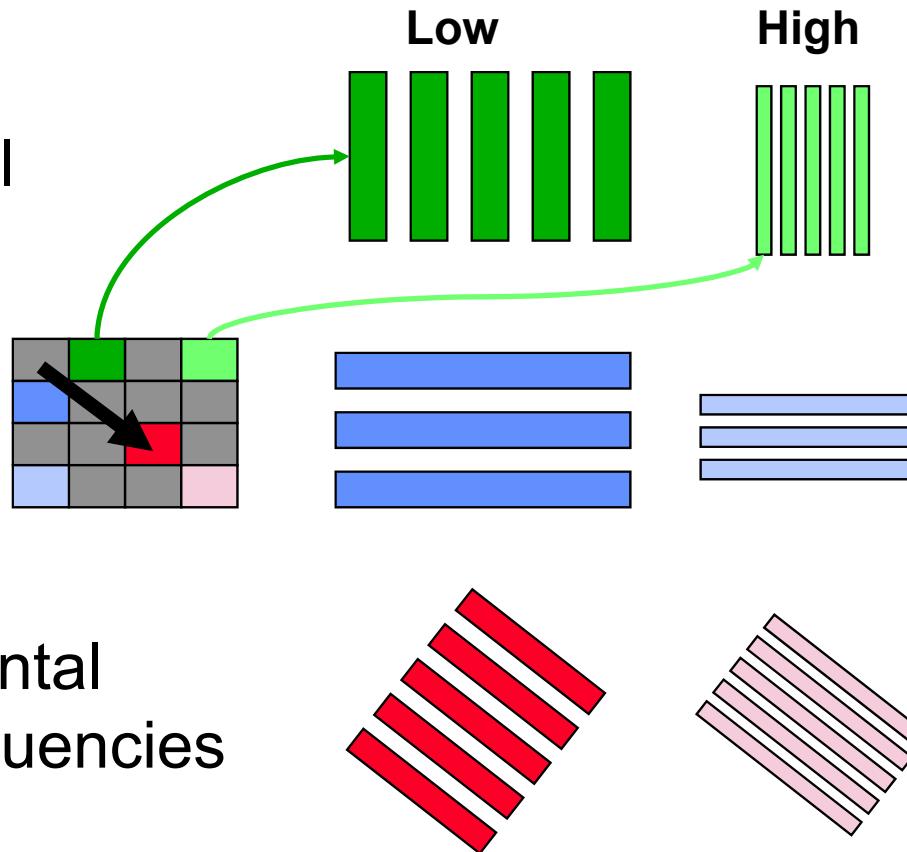
- Split Image up into RGB monochrome components
- Split monochrome components into 8x8 grayscale pixel blocks
- Perform Discrete Cosine Transform (DCT) on each block to get 8x8 spatial frequency bins (all real)





## Spatial Frequencies

- Horizontal Spatial Frequency
- Vertical Spatial Frequency
- Mixture of Horizontal and Vertical Frequencies





## Purpose of DCT

- Coefficient at index (0,0) corresponds to DC or average intensity and is always positive
- The other 63 coefficients correspond to AC components and can be either positive or negative
- Because pixel values typically vary slowly from point to point across an image, the DCT achieves data compression by concentrating most of the signal in the lower spatial frequencies
- For a typical 8x8 block most spatial frequencies will be zero or near zero and need not be encoded

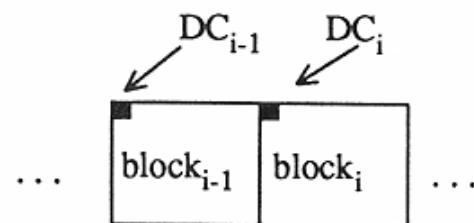


## Quantization

- After the DCT each coefficient is quantized non-uniformly into 8 bits. The quantization levels were carefully determined from psychovisual experiments (error level below perceptual threshold)
- After quantization, the DC coefficient is handled separately.
  - This is worth while, because it always contains a significant fraction of the image energy
  - It is encoded as the difference from the last DC coefficient encoded
- AC coefficients are scanned in a zigzag pattern and then compacted with either Huffman or Arithmetic coding
  - The zigzag ordering helps the compaction by placing low-frequency coefficients (which are more likely to be non-zero) before high frequency coefficients.

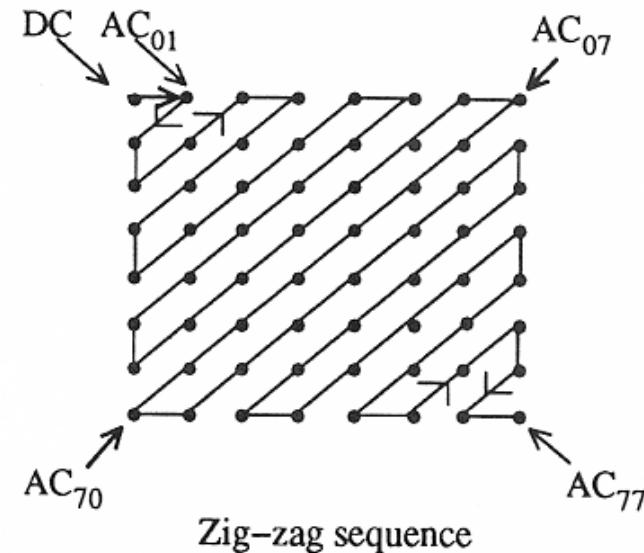


## Zigzag Encoding



$$\text{DIFF} = \text{DC}_i - \text{DC}_{i-1}$$

Differential DC encoding





## JPEG Processing Chain

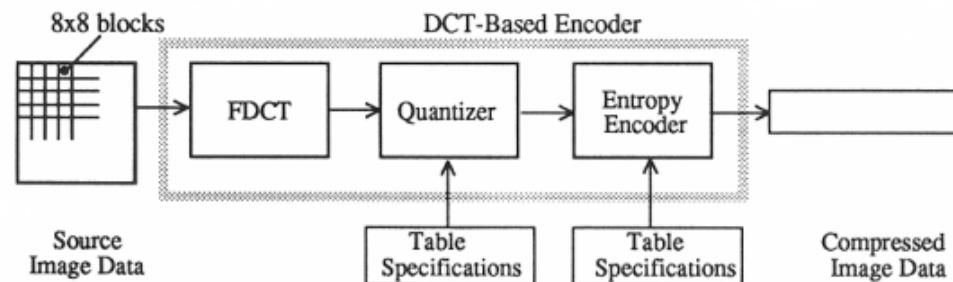


Figure 1. DCT-Based Encoder Processing Steps

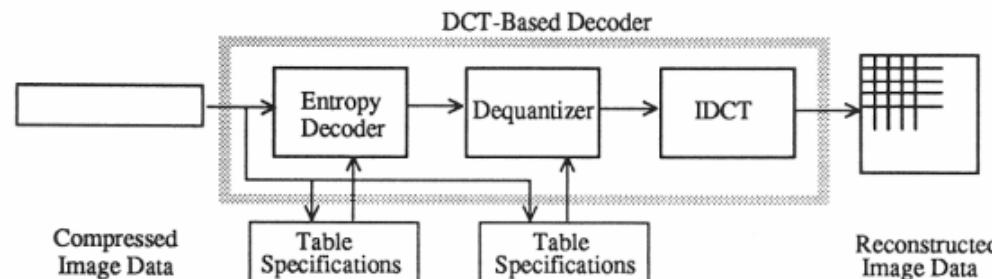


Figure 2. DCT-Based Decoder Processing Steps

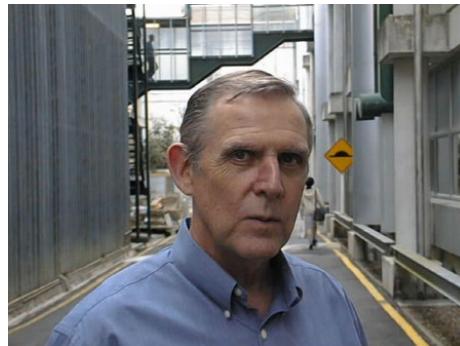


## Overall Performance

- 0.25-0.5 bits/pixel: moderate to good quality
- 0.5-0.75 bits/pixel: good to very good quality
- 0.75-1.5 bits/pixel: excellent quality, sufficient for most applications

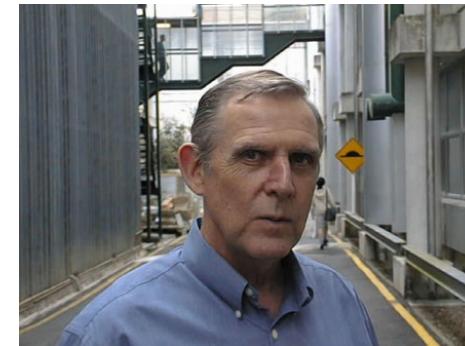
Note: JPEG can be lossless  
but rarely implemented

Original BMP file 1:1  
448x336 pixels



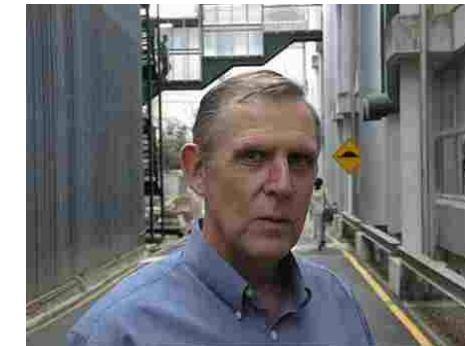
442 kB

Good JPEG 14:1



32 kB

Acceptable JPEG 63:1



7 kB

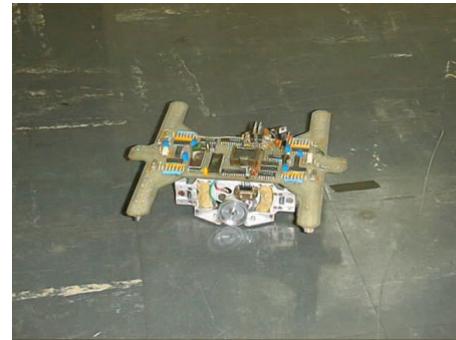


## Common Image Processing Tasks

- Thresholding (2 level quantization)
- Requantizing (changing number of quantization levels)
  - sometimes called posterizing in application packages
  - necessary to convert 24 bit colour to 8 bit colour models
  - may require dither to prevent “false contours”
- Blurring
  - convolving with a 2D low pass FIR filter (point spread function)
- Sharpening
  - convolving with a 2D high boost FIR filter
  - note: do not use high pass as you will lose DC component and resultant image will have negative intensity values

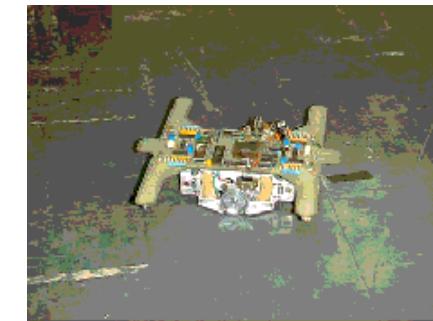


Original Image



## Examples

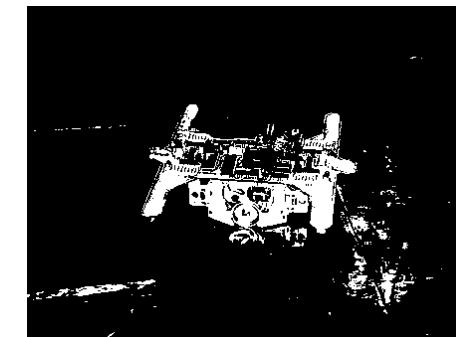
Posterizing:  
reduced to  
8 colours only



↓ Grayscale  
conversion



Thresholding





## Examples

### 3x3 2D Filter

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



lowpass

Blurred



2D CONVOLUTION

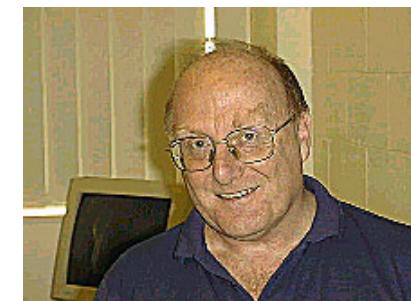
### 3x3 2D Filter

$$\frac{1}{10} * \begin{bmatrix} 1 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.1 & -0.1 & 0.1 \\ -0.1 & 1 & -0.1 \\ 0.1 & -0.1 & 0.1 \end{bmatrix}$$

highboost

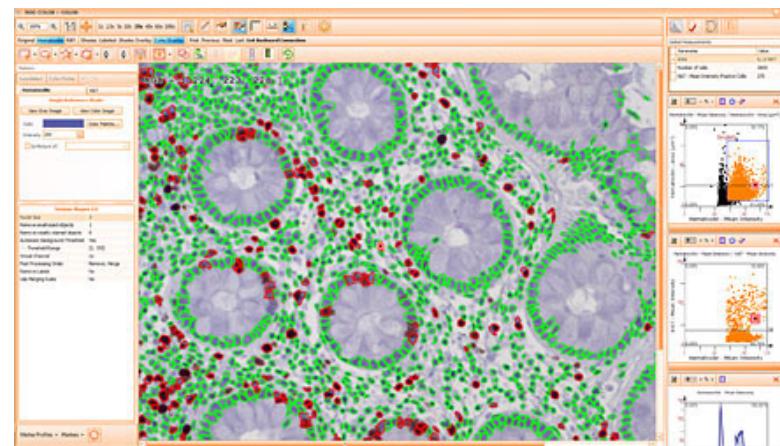
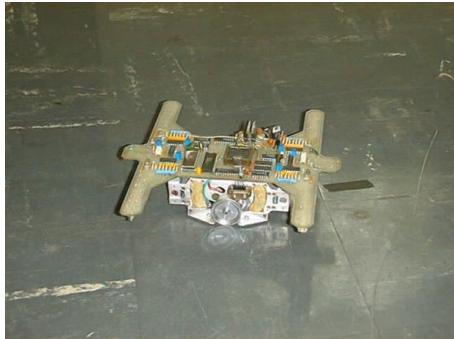
2D CONVOLUTION

Sharpened





# Image Analysis



Extracting Information From Images



# Image Analysis

- The first step in image analysis is generally to segment the image.
- The level of segmentation depends on the problem to hand.
- For example, in an autonomous air-to-ground missile system we may segment the road from the image and then segment the road into objects that may correspond to vehicles. There is no point in segmenting below this scale or in looking at objects outside the boundary of the road.



# Image Segmentation

- In general, autonomous segmentation is one of the most difficult tasks in image processing.
- Unfortunately, the performance of this difficult first step generally determines the overall success of the whole system.
- In fact, effective segmentation rarely fails to lead to a successful system.
- In industrial inspection applications, we often have some control over the environment which can greatly simplify the task of segmentation.



# Vehicle Segmentation Example



**Hand drawn. How do we do this with a computer?**



# Controlling the Environment

- If we can control some aspects of the imaging environment, we can often greatly simplify the task of segmentation. An experienced image analyst will never miss an opportunity to exploit this.
- Examples.
  - Control of lighting intensity, position of lights, colour of light.
  - Control of background field.
  - Control of camera type, focal length, position and aspect/orientation of object.
  - Use of segmentation aids such as staining of cell images.
  - Use of prior knowledge of expected objects under analysis.
  - Use of additional sensors (microwave).
  - Use of motion cues – temporal filtering.

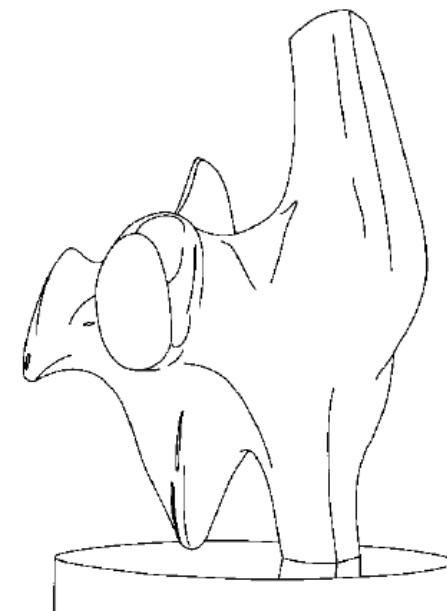
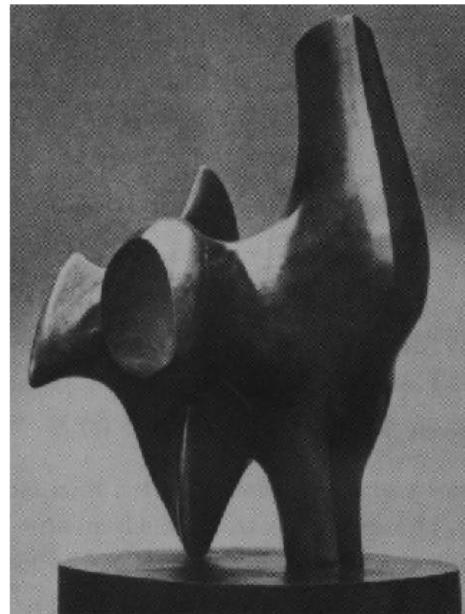


# Segmentation Techniques

- Segmentation techniques for monochrome images generally are based on two basic properties of grayscale values
  - Discontinuity and Similarity
- Discontinuity based methods
  - detection of points, lines, and edges in an image
- Similarity Based Methods
  - Thresholding, region growing, region splitting and merging
- If we are examining dynamic (time-varying) images, then motion can give powerful cues to improve segmentation

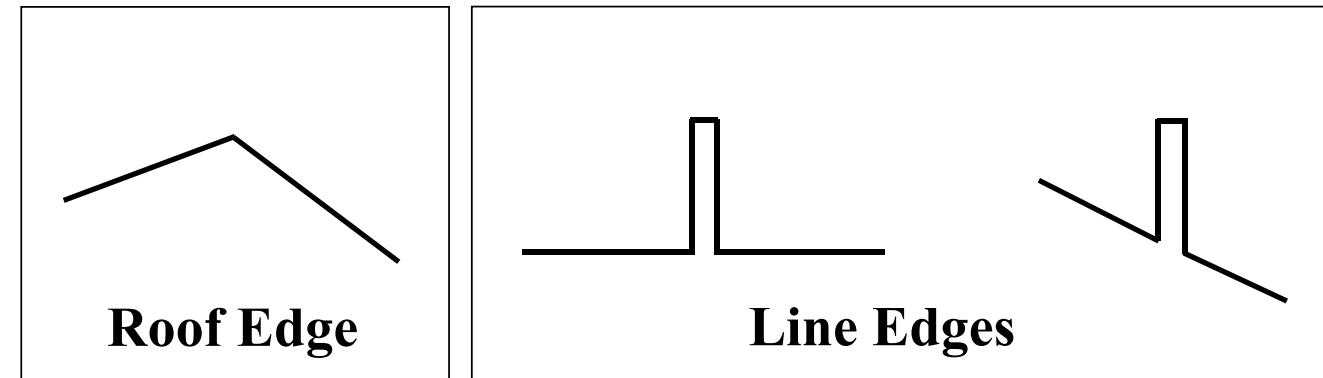
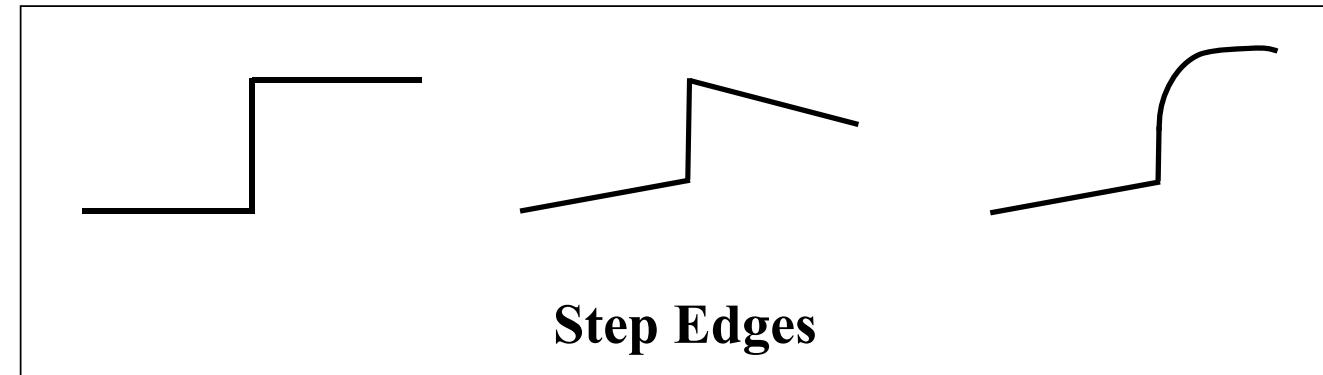


# How can you tell that a pixel is on an edge?



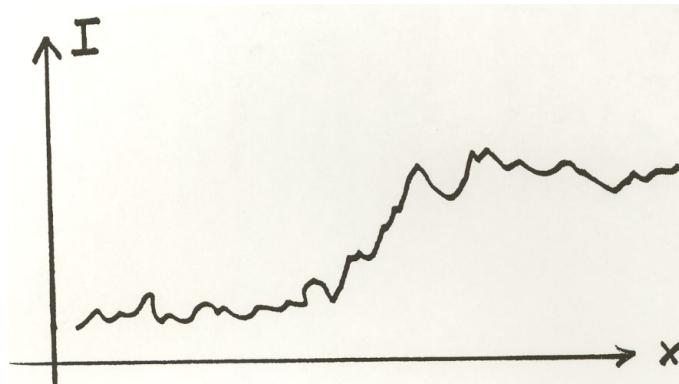


# Edge Types





# Real Edges



**Noisy and Discrete!**

We want an Edge Operator that produces:

- Edge **Magnitude**
- Edge **Orientation**
- High **Detection Rate** and Good **Localization**



# Gradient

- Gradient equation:  $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$
- Represents direction of most rapid change in intensity

$$\nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right]$$

$$\nabla f = \left[ 0, \frac{\partial f}{\partial y} \right]$$

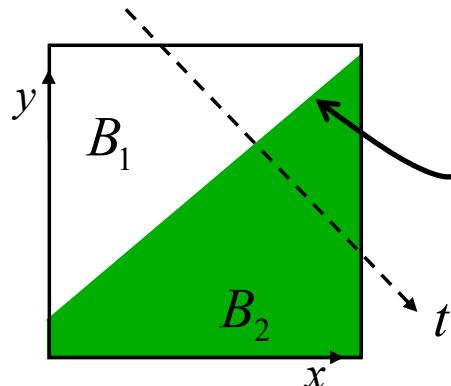
$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- **Gradient direction:**  $\theta = \tan^{-1} \left( \frac{\partial f / \partial y}{\partial f / \partial x} \right)$
- **The edge strength is given by the gradient magnitude**

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



# Theory of Edge Detection



**Ideal edge**

$$L(x, y) = x \sin \theta - y \cos \theta + \rho = 0$$

$$B_1 : L(x, y) < 0$$

$$B_2 : L(x, y) > 0$$

**Unit step function:**

$$u(t) = \begin{cases} 1 & \text{for } t > 0 \\ \frac{1}{2} & \text{for } t = 0 \\ 0 & \text{for } t < 0 \end{cases} \quad u(t) = \int_{-\infty}^t \delta(s) ds$$

**Image intensity (brightness):**

$$I(x, y) = B_1 + (B_2 - B_1)u(x \sin \theta - y \cos \theta + \rho)$$



# Theory of Edge Detection

- **Image intensity (brightness):**

$$I(x, y) = B_1 + (B_2 - B_1)u(x \sin \theta - y \cos \theta + \rho)$$

- **Partial derivatives (gradients):**

$$\frac{\partial I}{\partial x} = +\sin \theta (B_2 - B_1) \delta(x \sin \theta - y \cos \theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos \theta (B_2 - B_1) \delta(x \sin \theta - y \cos \theta + \rho)$$

- **Squared gradient:**

$$s(x, y) = \left( \frac{\partial I}{\partial x} \right)^2 + \left( \frac{\partial I}{\partial y} \right)^2 = [(B_2 - B_1) \delta(x \sin \theta - y \cos \theta + \rho)]^2$$

**Edge Magnitude:**  $\sqrt{s(x, y)}$

**Rotationally symmetric, non-linear operator**

**Edge Orientation:**  $\arctan \left( \frac{\partial I}{\partial y} / \frac{\partial I}{\partial x} \right)$  (normal of the edge)



- **Image intensity (brightness):**

$$I(x, y) = B_1 + (B_2 - B_1)u(x \sin \theta - y \cos \theta + \rho)$$

- **Partial derivatives (gradients):**

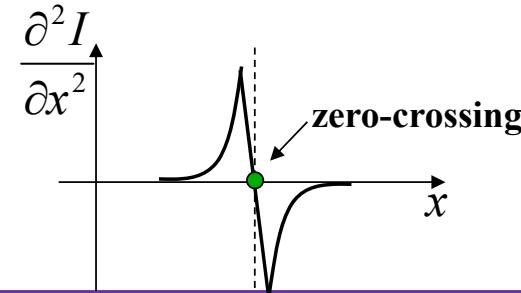
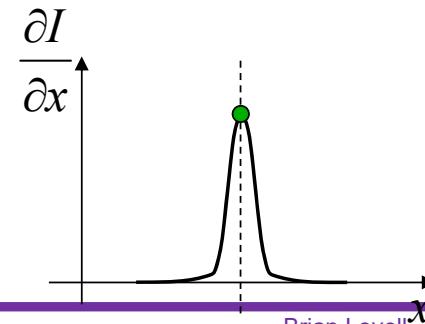
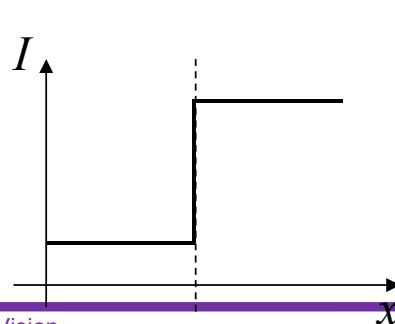
$$\frac{\partial I}{\partial x} = +\sin \theta (B_2 - B_1) \delta(x \sin \theta - y \cos \theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos \theta (B_2 - B_1) \delta(x \sin \theta - y \cos \theta + \rho)$$

- **Laplacian:**

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = (B_2 - B_1) \delta'(x \sin \theta - y \cos \theta + \rho)$$

Rotationally symmetric, linear operator





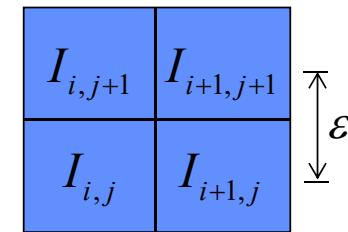
# Discrete Edge Operators

- How can we differentiate a *discrete* image?

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon} ((I_{i+1,j+1} - I_{i,j+1}) + (I_{i+1,j} - I_{i,j}))$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon} ((I_{i+1,j+1} - I_{i+1,j}) + (I_{i,j+1} - I_{i,j}))$$



Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|} \hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array}$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|} \hline 1 & 1 \\ \hline -1 & -1 \\ \hline \end{array}$$



# Discrete Edge Operators

- Second order partial derivatives:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2} (I_{i-1,j} - 2I_{i,j} + I_{i+1,j})$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2} (I_{i,j-1} - 2I_{i,j} + I_{i,j+1})$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

- Laplacian :

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

Convolution masks :

$$\nabla^2 I \approx \frac{1}{\varepsilon^2} \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

or  $\frac{1}{6\varepsilon^2}$

$$\begin{array}{|c|c|c|} \hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline \end{array}$$

(more accurate)



# The Sobel Operators

- Better approximations of the gradients exist
  - The *Sobel* operators below are commonly used

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$s_x$

$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$s_y$



**Gradient:**

$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

**Good Localization  
Noise Sensitive  
Poor Detection**

**Roberts (2 x 2):**

0	1
-1	0

1	0
0	-1

**Sobel (3 x 3):**

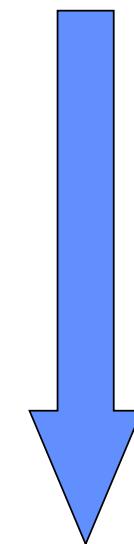
-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	1

**Sobel (5 x 5):**

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

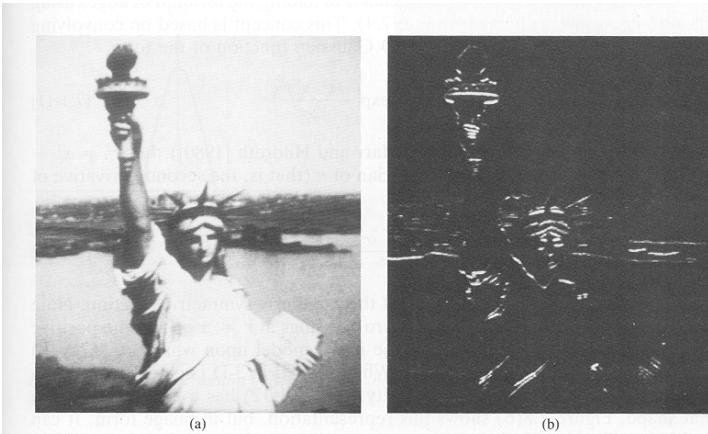


**Poor Localization  
Less Noise Sensitive  
Good Detection**



## Sobel Edge Detection

Original



$G_x$

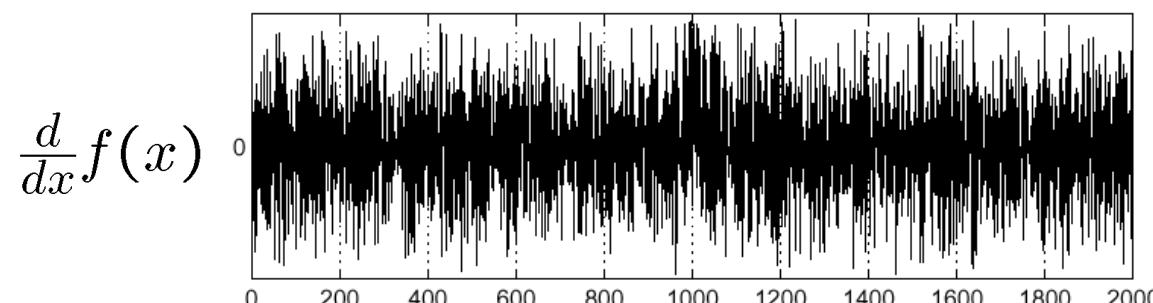
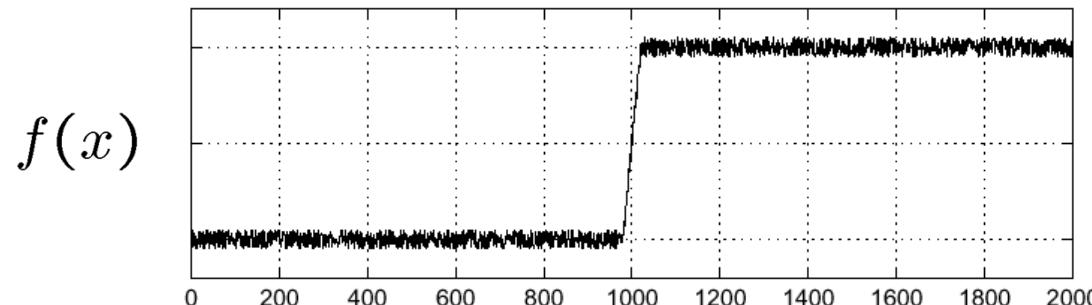
$G_y$

Combined



# Effects of Noise

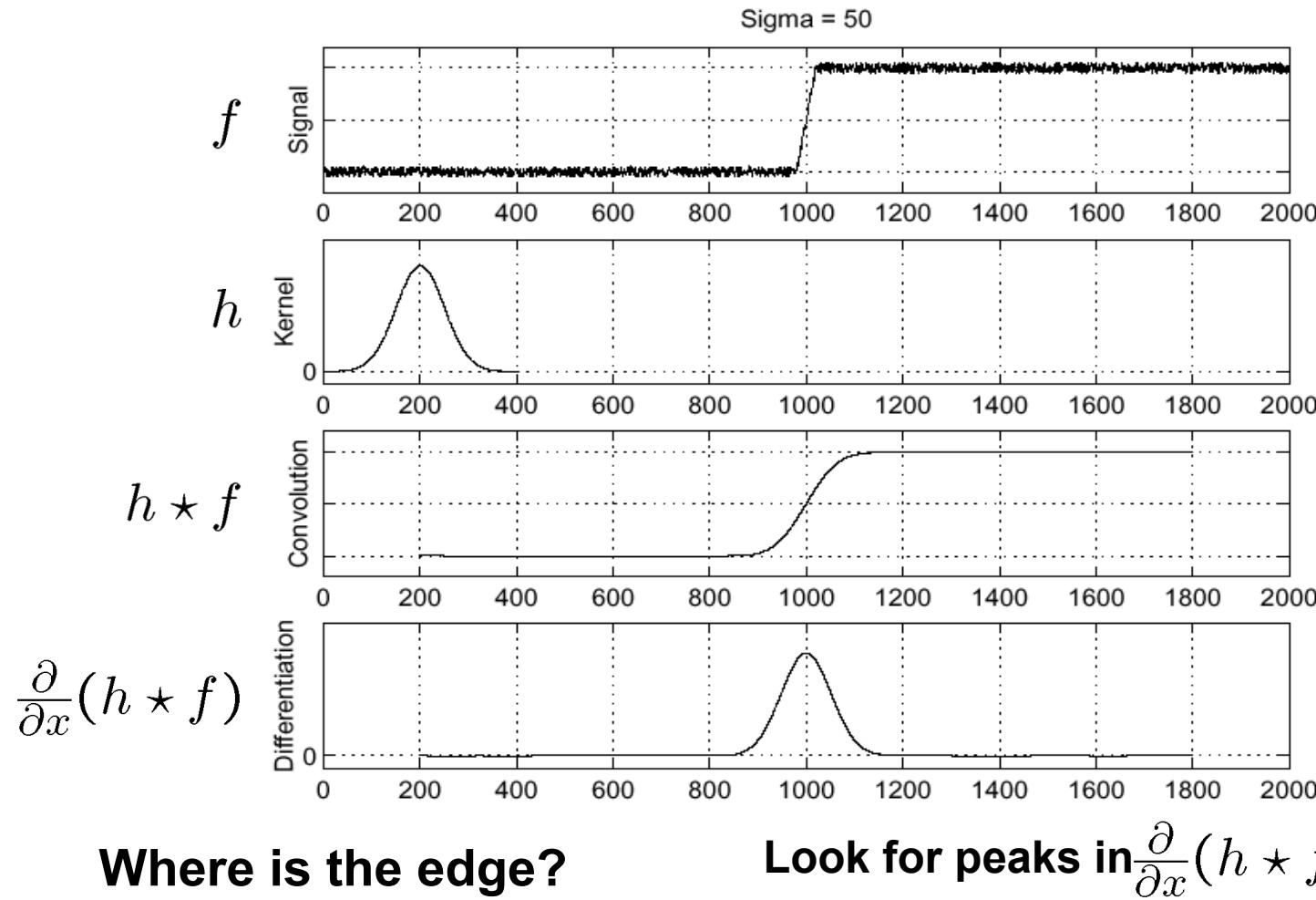
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



**Where is the edge??**



# Solution: Smooth First



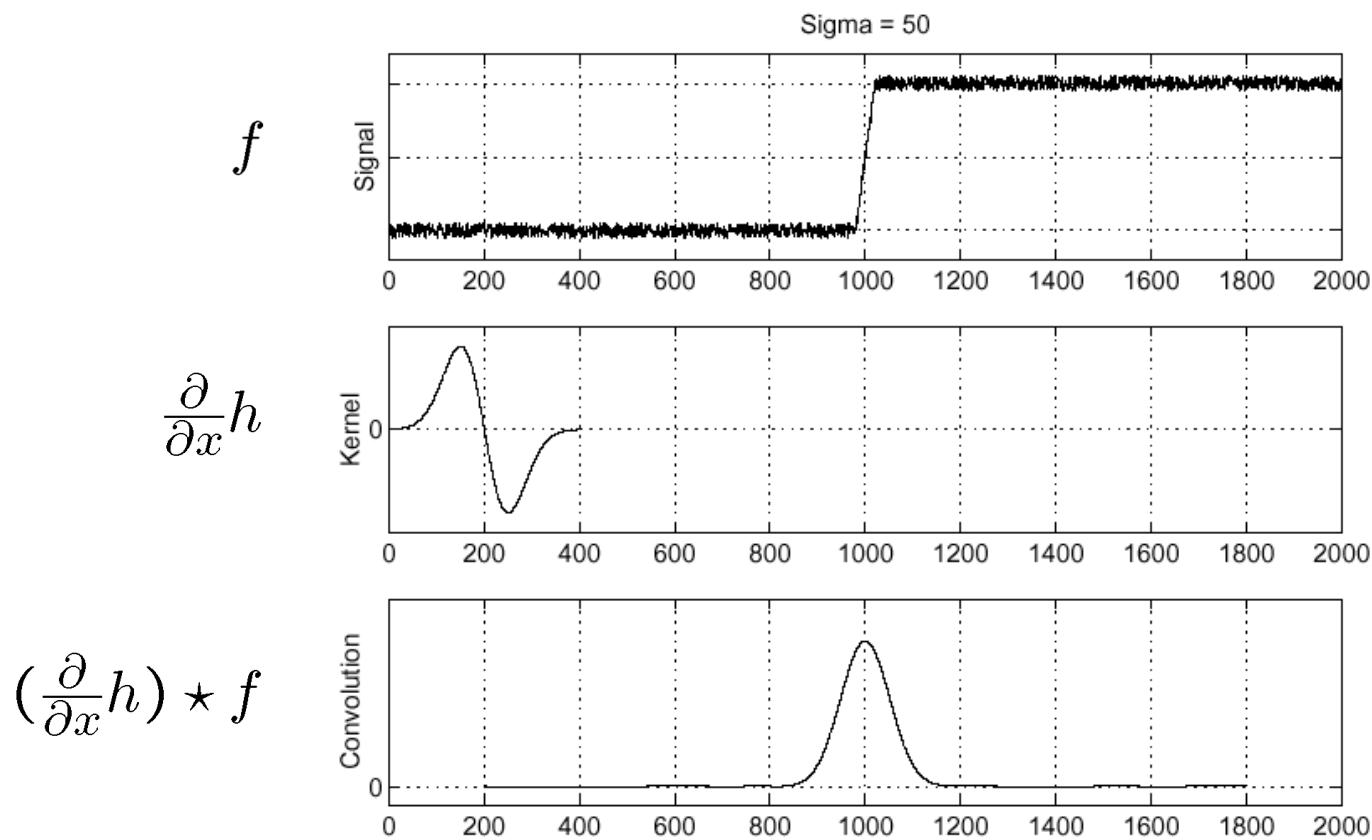
Where is the edge?

Look for peaks in  $\frac{\partial}{\partial x}(h \star f)$



# Derivative Theorem of Convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f \quad \dots \text{saves us one operation.}$$

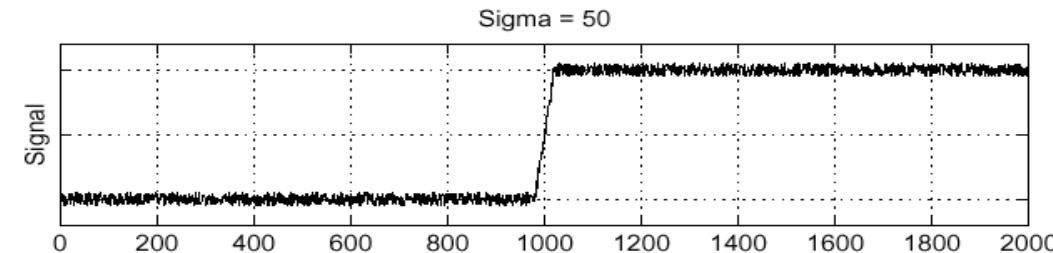
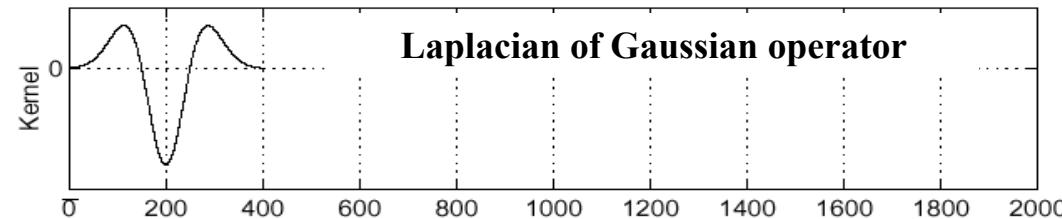
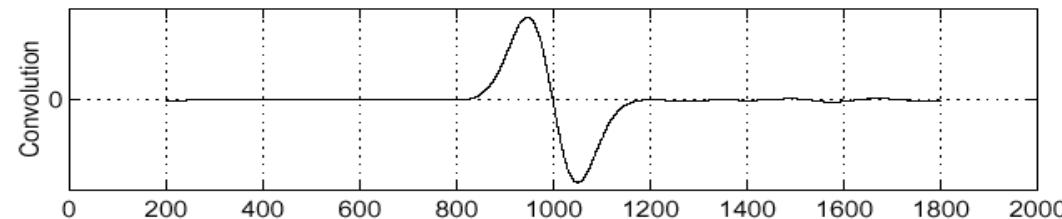




# Laplacian of Gaussian (LoG)

$$\frac{\partial^2}{\partial x^2}(h * f) = \left( \frac{\partial^2}{\partial x^2} h \right) * f$$

Laplacian of Gaussian

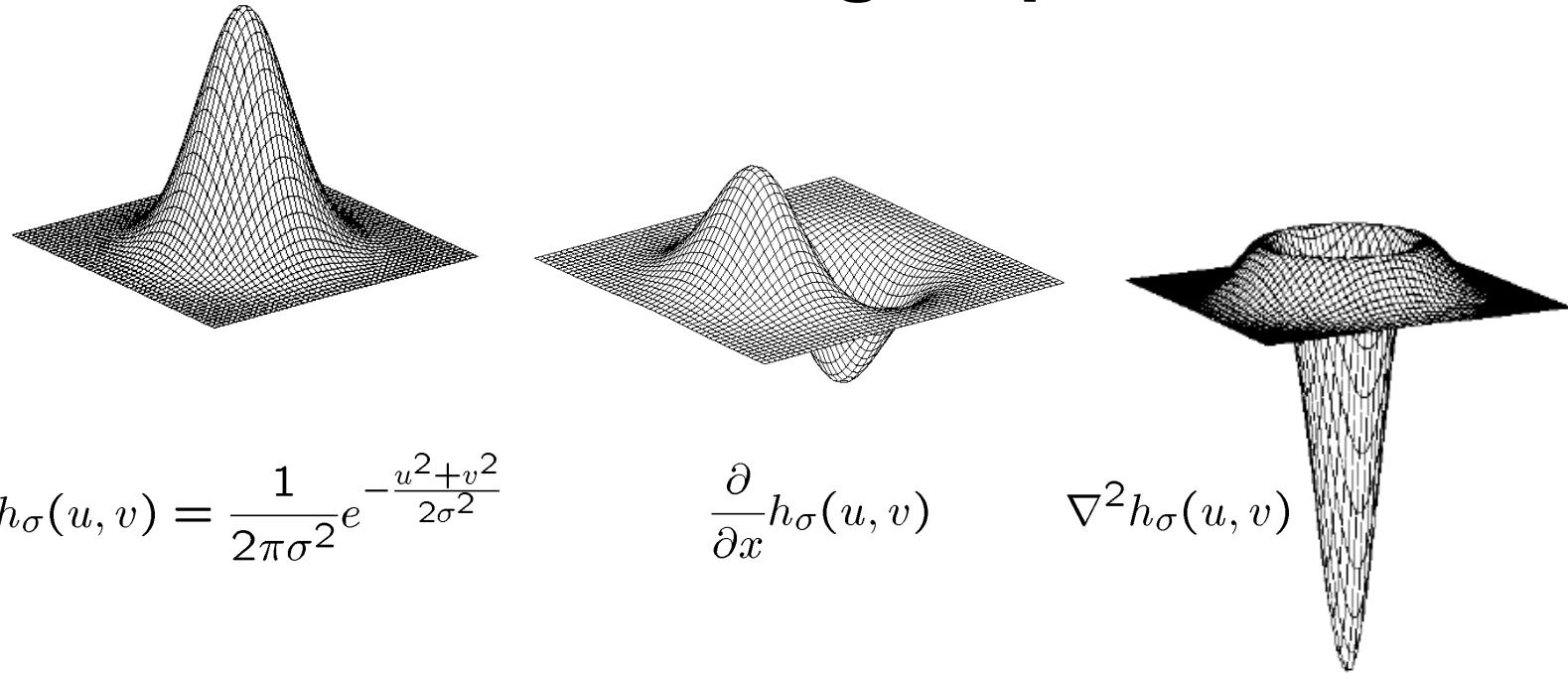
 $f$  $\frac{\partial^2}{\partial x^2} h$  $(\frac{\partial^2}{\partial x^2} h) * f$ 

Where is the edge?

Zero-crossings of bottom graph !



# 2D Gaussian Edge Operators



$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$

$$\nabla^2 h_\sigma(u, v)$$

**Gaussian**

**Derivative of Gaussian (DoG)**

**Laplacian of Gaussian  
Mexican Hat (Sombrero)**

- $\nabla^2$  is the **Laplacian operator**:

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



# Canny Edge Operator

- Smooth image  $I$  with 2D Gaussian:  $G * I$
- Find local edge normal directions for each pixel

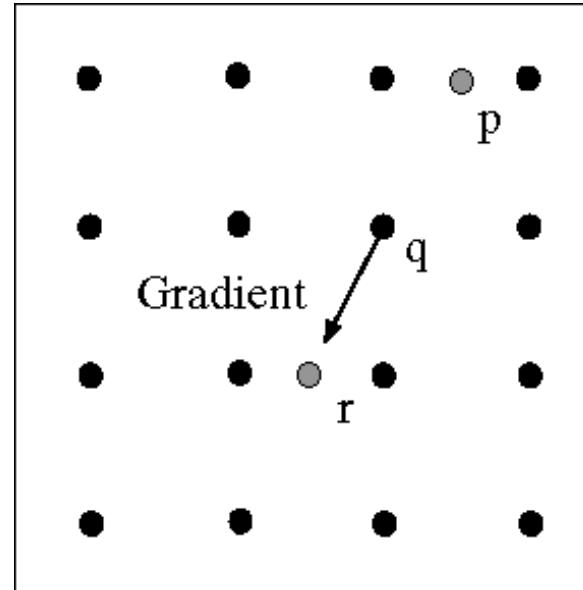
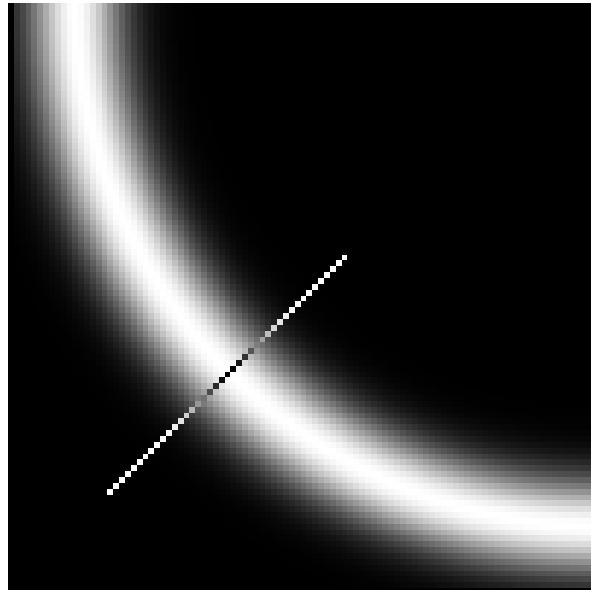
$$\bar{\mathbf{n}} = \frac{\nabla(G * I)}{|\nabla(G * I)|}$$

- Compute edge magnitudes  $|\nabla(G * I)|$
- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2(G * I)}{\partial \bar{\mathbf{n}}^2} = 0$$



# Non-maximum Suppression



- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r
  - Also uses hysteresis thresholding to follow weak edges below primary threshold



# The Canny Edge Detector



original image (Lena)



# The Canny Edge Detector



**magnitude of the gradient**



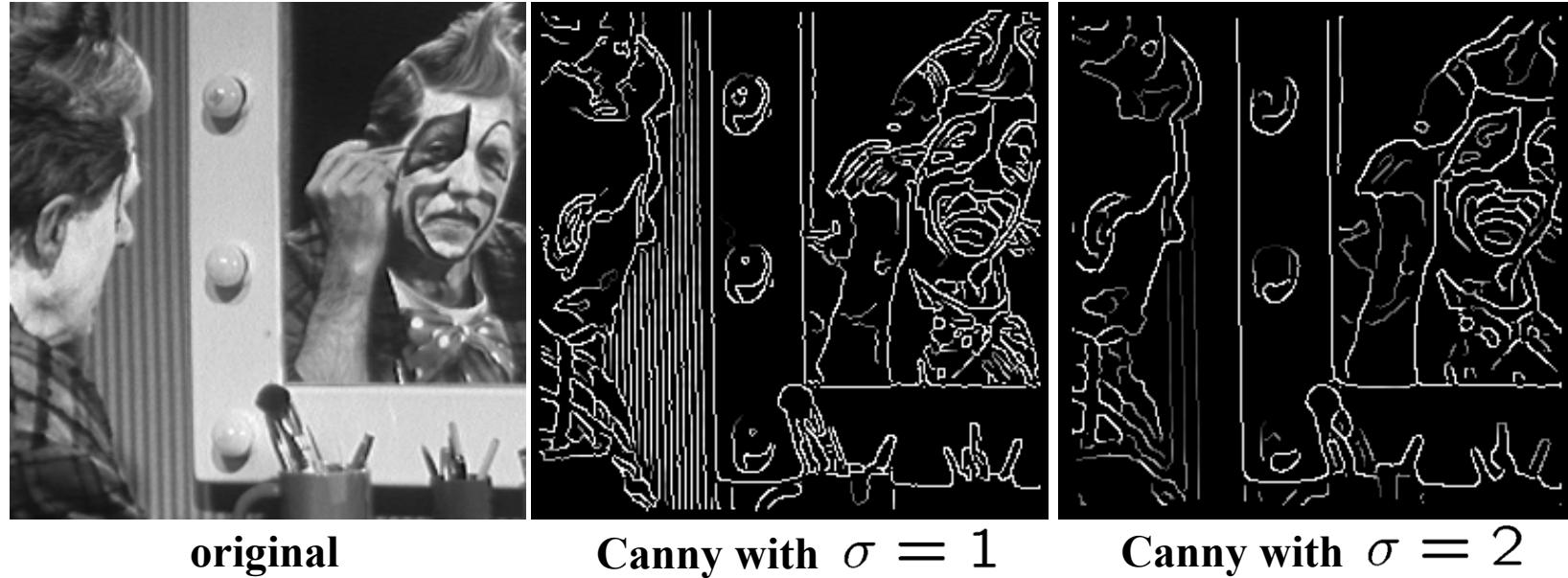
# The Canny Edge Detector



**After non-maximum suppression**



# Canny Edge Operator



- **The choice of  $\sigma$  depends on desired behavior**
  - large  $\sigma$  detects large scale edges
  - small  $\sigma$  detects fine features



# Canny Edge Detector (and Linker)

*Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.*

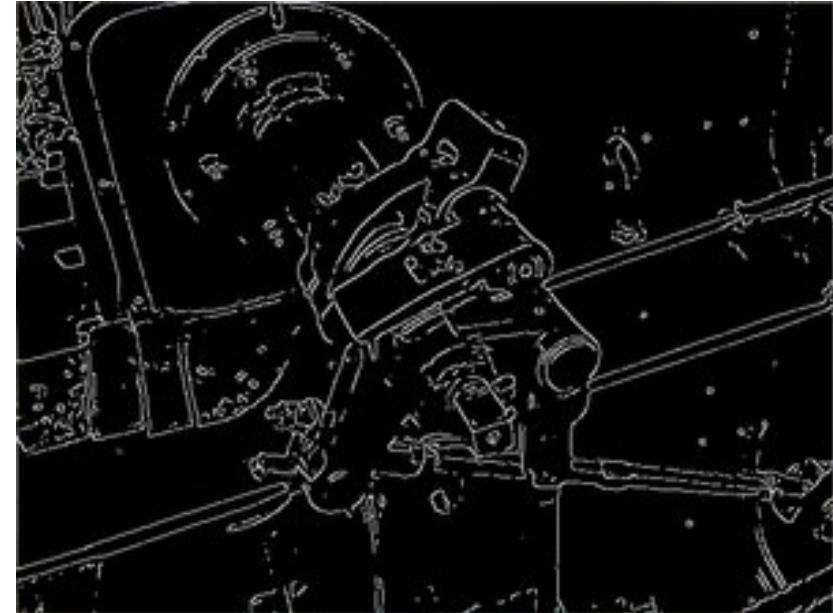
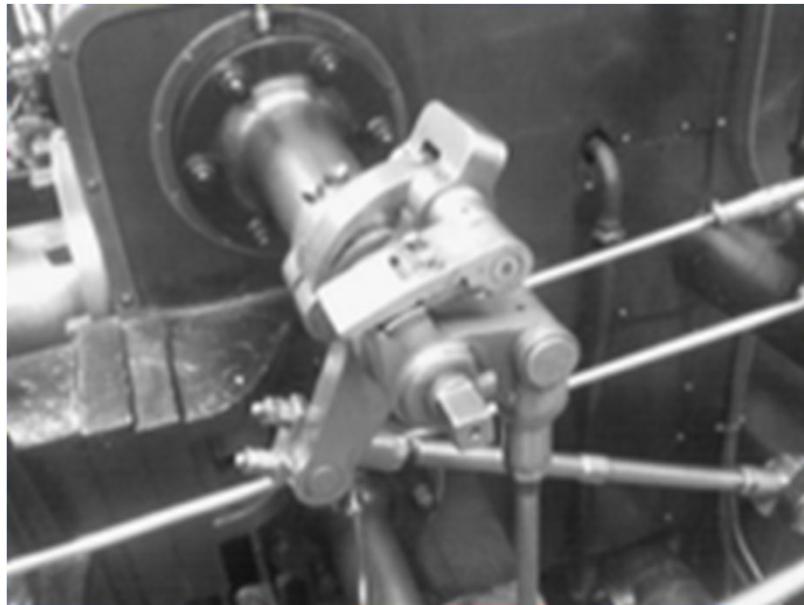
## Processing Steps

- Apply Gaussian filter to smooth the image in order to remove the noise
- Find the intensity gradients of the image
- Apply non-maximum suppression to get rid of spurious response to edge detection (ridge following)
- Apply double threshold to determine potential edges
- Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.



# Canny Edge Detector

The preferred method for many edge detection tasks





# Image Resampling and Pyramids



# Text Aliasing

gv: kpathsea.dvi

File State Page Portrait 3.000 BBox

Fixed Size Open Print All Print Marked Save All Save Marked

<> Redisplay

Chapter 1: Introduction

## 1 Introduction

This manual corresponds to version 3.2 of the Kpathsea library.

The library's fundamental purpose is to return a filename for a user, similar to what shells do when looking up program names.

The following software, all of which we maintain, uses the library:

- Dviljk (see the ‘dviljk’ man page)
- Dvipsk (see section “Introduction” in Dvips: A DVI driver)
- GNU font utilities (see section “Introduction” in Gnu-Font: A collection of GNU font utilities)
- Web2c (see section “Introduction” in Web2c: A TeX implementation)
- Xdvik (see the ‘xdvik’ man page)

gv: kpathsea.dvi

File State Page Portrait 3.000 BBox

Fixed Size Open Print All Print Marked Save All Save Marked

<> Redisplay

Chapter 1: Introduction

## 1 Introduction

This manual corresponds to version 3.2 of the Kpathsea library.

The library's fundamental purpose is to return a filename for a user, similar to what shells do when looking up program names.

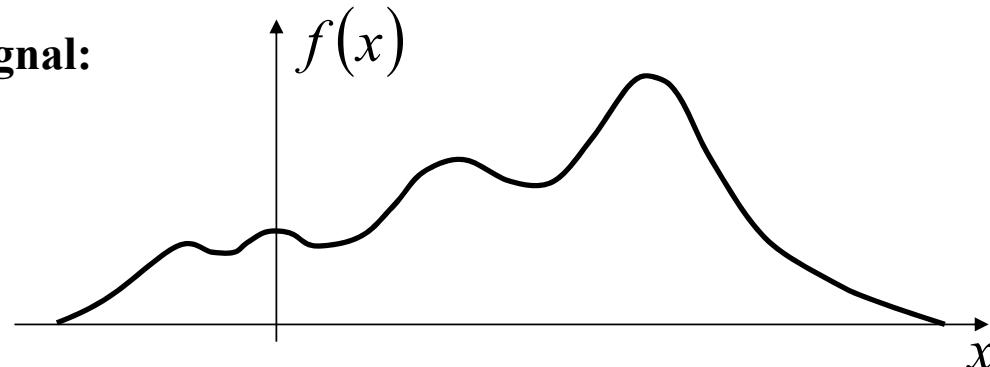
The following software, all of which we maintain, uses the library:

- Dviljk (see the ‘dviljk’ man page)
- Dvipsk (see section “Introduction” in Dvips: A DVI driver)
- GNU font utilities (see section “Introduction” in Gnu-Font: A collection of GNU font utilities)
- Web2c (see section “Introduction” in Web2c: A TeX implementation)
- Xdvik (see the ‘xdvik’ man page)

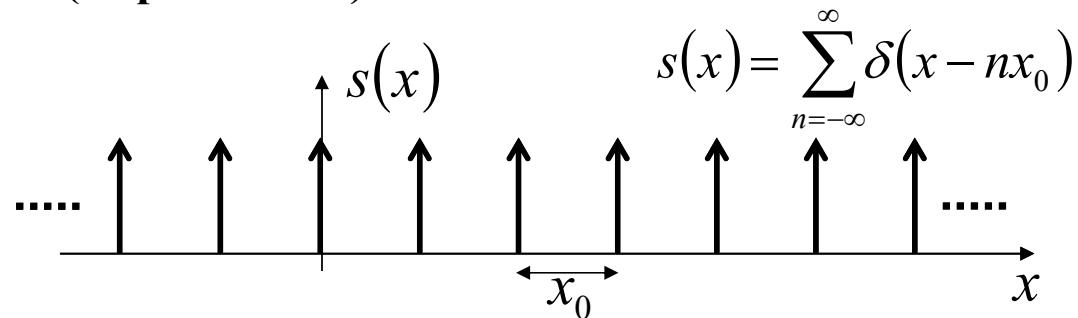


# RECAP - Sampling Theorem

Continuous signal:



Shah function (Impulse train):



Sampled function:

$$f_s(x) = f(x)s(x) = f(x) \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$



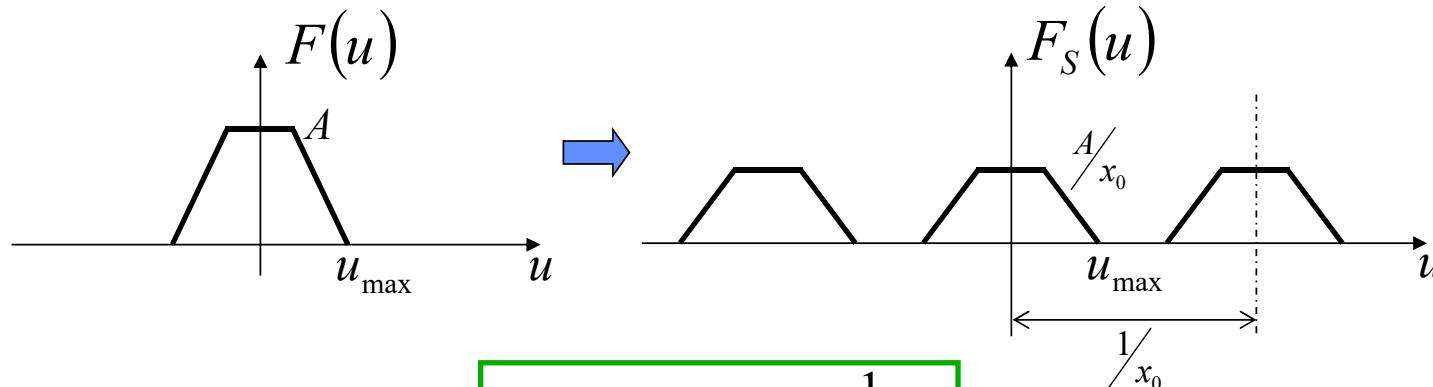
# RECAP - Sampling Theorem

Sampled function:

$$f_s(x) = f(x)s(x) = f(x) \sum_{n=-\infty}^{\infty} \delta(x - nx_0)$$

Sampling frequency  $\frac{1}{x_0}$

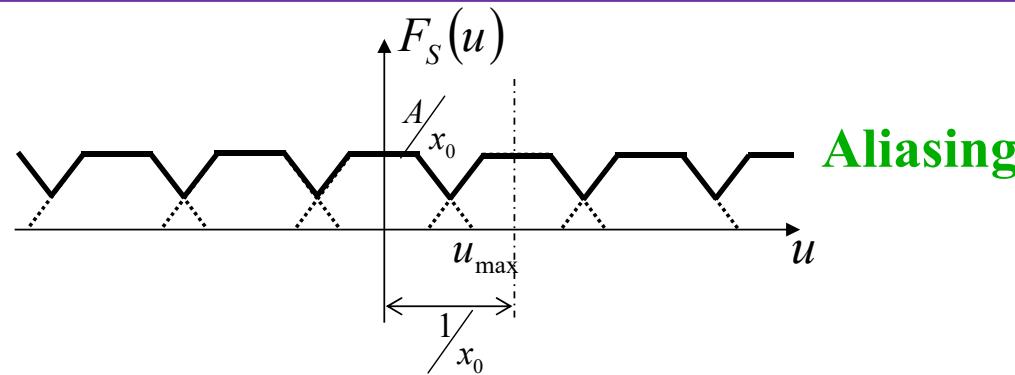
$$F_S(u) = F(u) * S(u) = F(u) * \frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{x_0}\right)$$



Only if  $u_{max} \leq \frac{1}{2x_0}$



If  $u_{\max} > \frac{1}{2x_0}$



When can we recover  $F(u)$  from  $F_S(u)$ ?

Only if  $u_{\max} \leq \frac{1}{2x_0}$  (Nyquist Frequency)

We can use

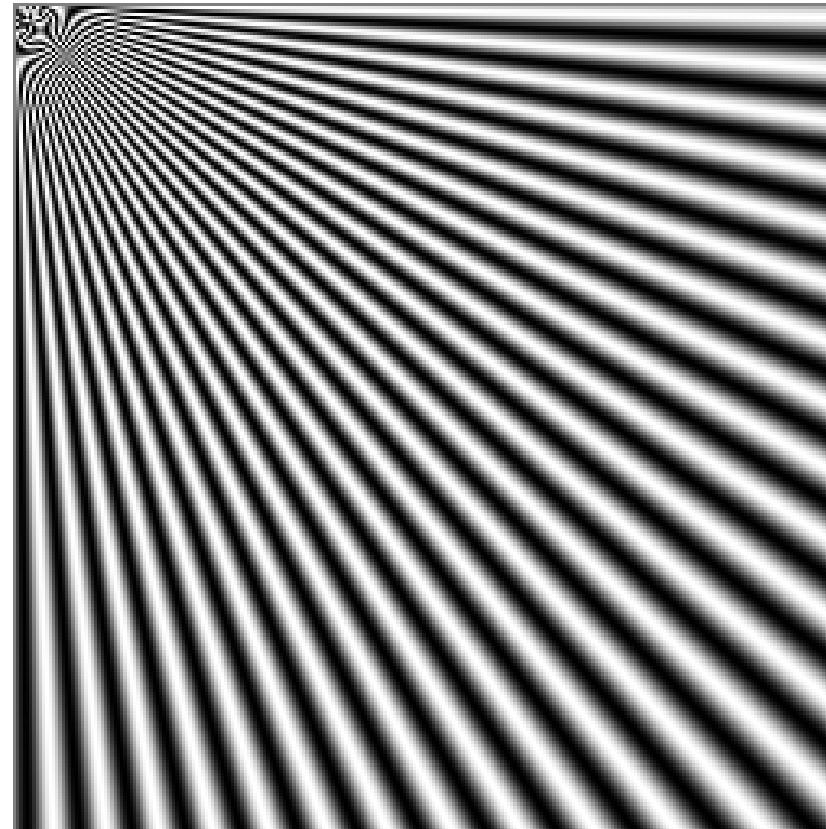
$$C(u) = \begin{cases} x_0 & |u| < \frac{1}{2x_0} \\ 0 & \text{otherwise} \end{cases}$$

Then  $F(u) = F_S(u)C(u)$  and  $f(x) = \text{IFT}[F(u)]$

Sampling frequency must be greater than  $2u_{\max}$



# Aliasing

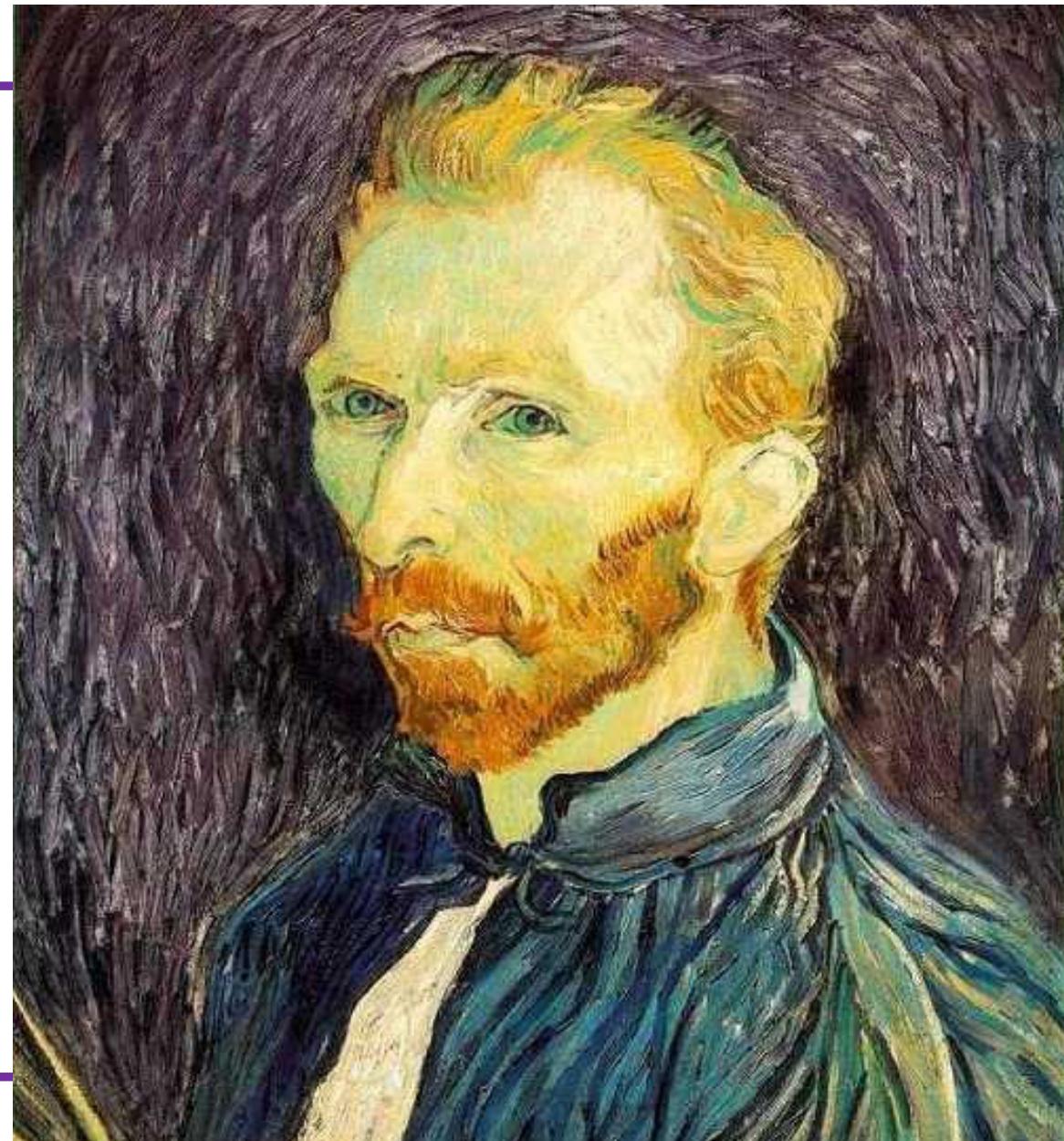




# Image Scaling

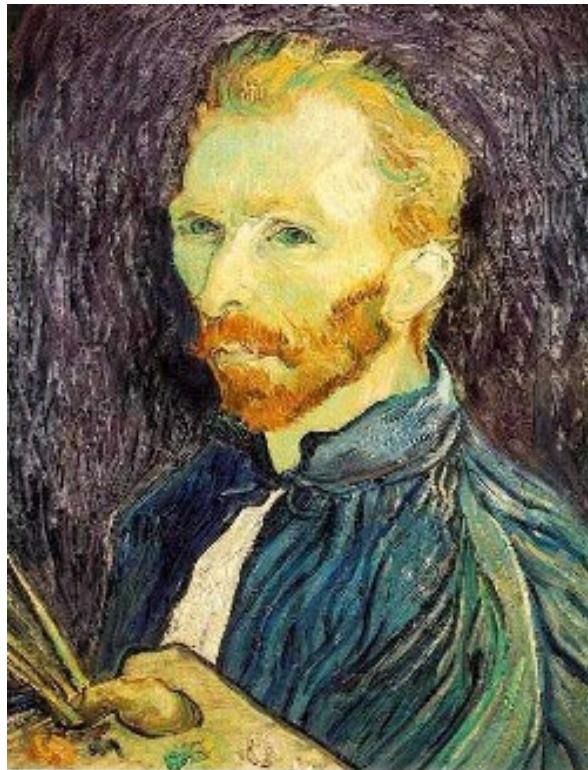
**This image is too big to fit on the screen. How can we reduce it?**

**How to generate a half-sized version?**





# Image Sub-Sampling



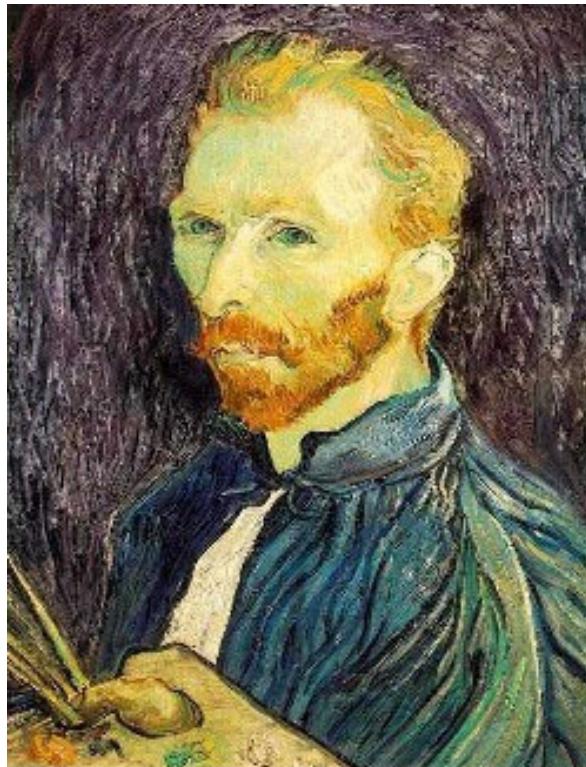
1/4

1/8

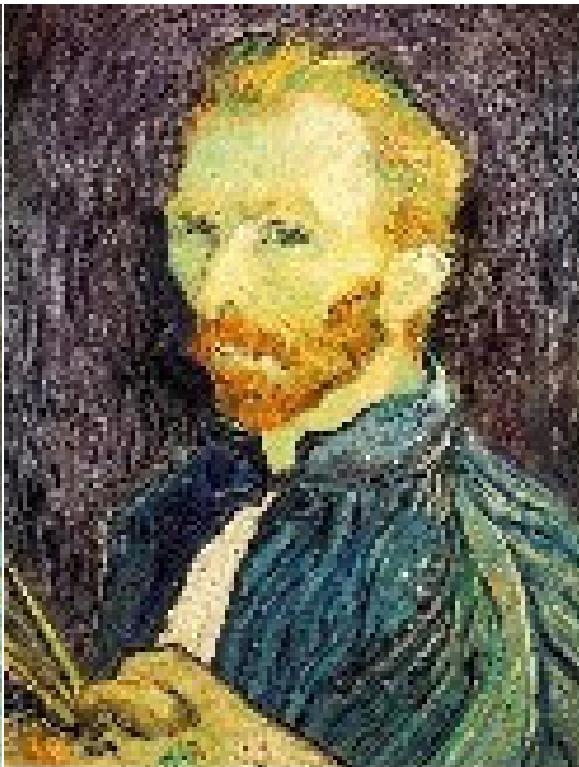
**Throw away every other row and  
column to create a 1/2 size image  
- called *image sub-sampling***



# Image Sub-Sampling



**1/2**



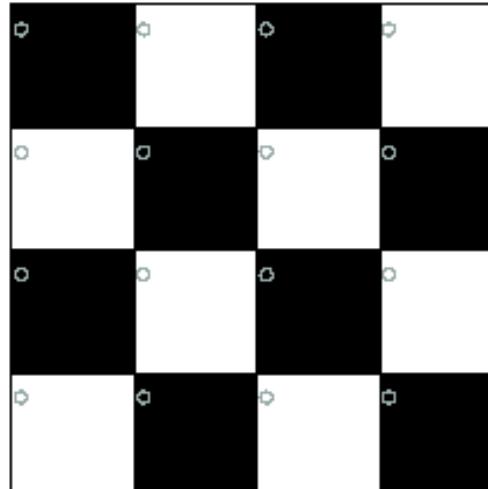
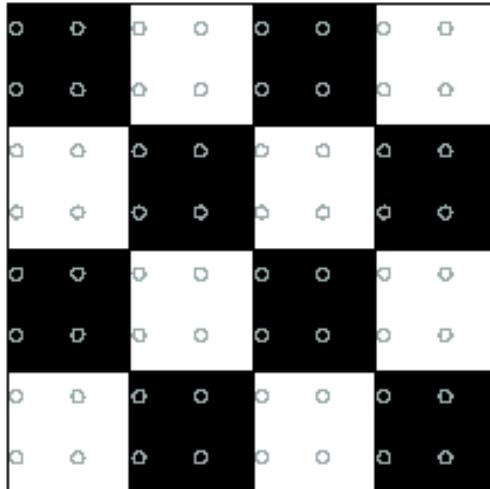
**1/4 (2x zoom)**



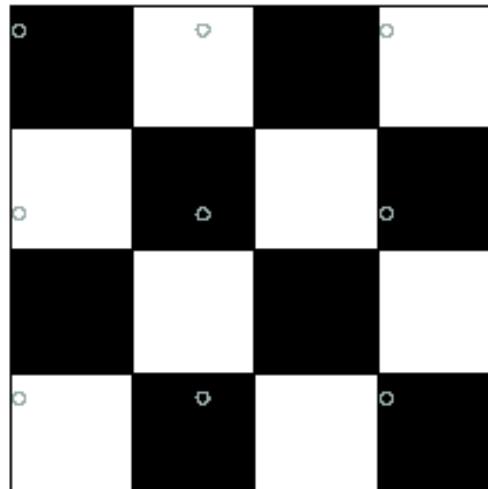
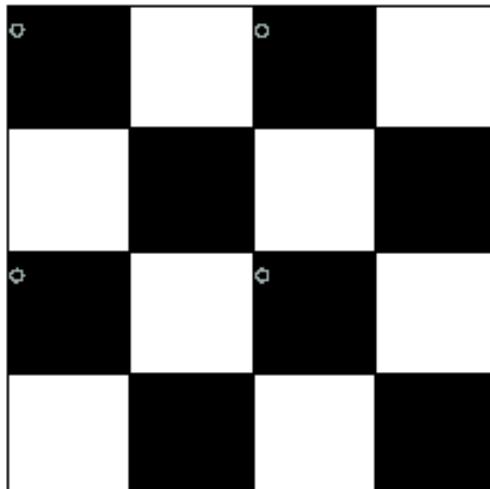
**1/8 (4x zoom)**



# Good and Bad Sampling



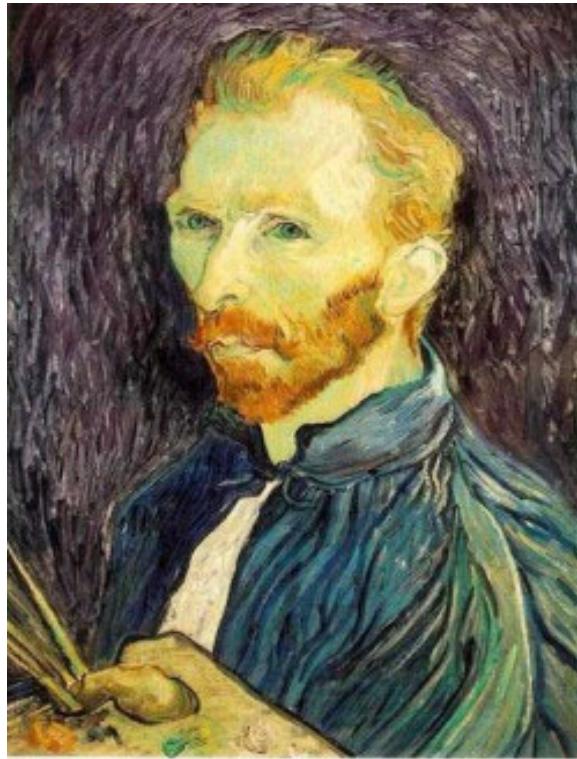
**Good sampling:**  
•Sample often or,  
•Sample wisely



**Bad sampling:**  
•see aliasing in action!



# Sub-Sampling with Gaussian Pre-Filtering



G 1/4



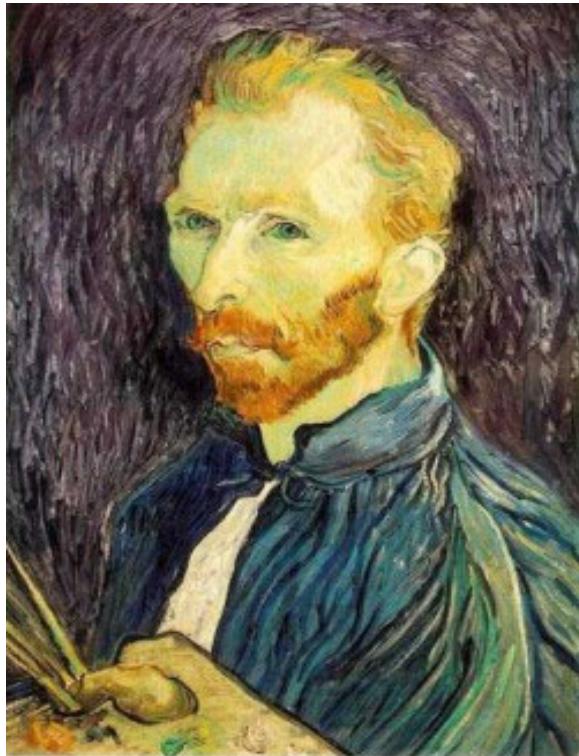
G 1/8

**Gaussian 1/2**

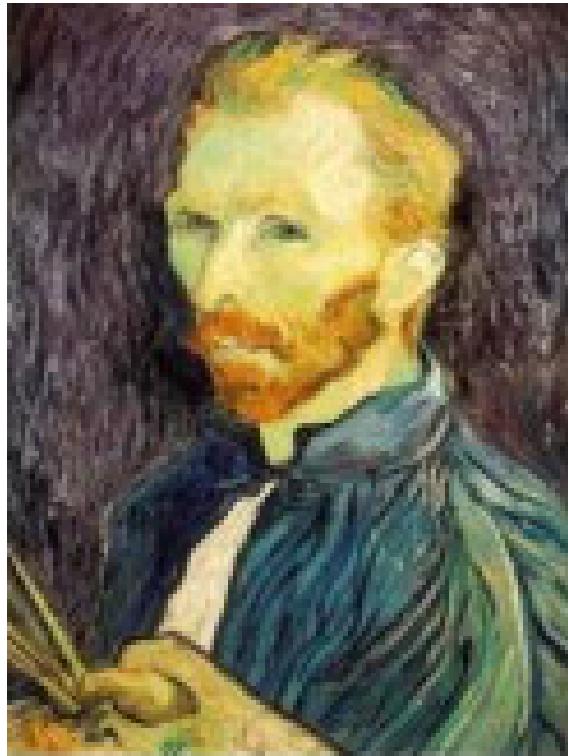
- **Solution: filter the image, *then* subsample**
  - Filter size should double for each  $\frac{1}{2}$  size reduction. Why?



## Sub-Sampling with Gaussian Pre-Filtering



**Gaussian 1/2**



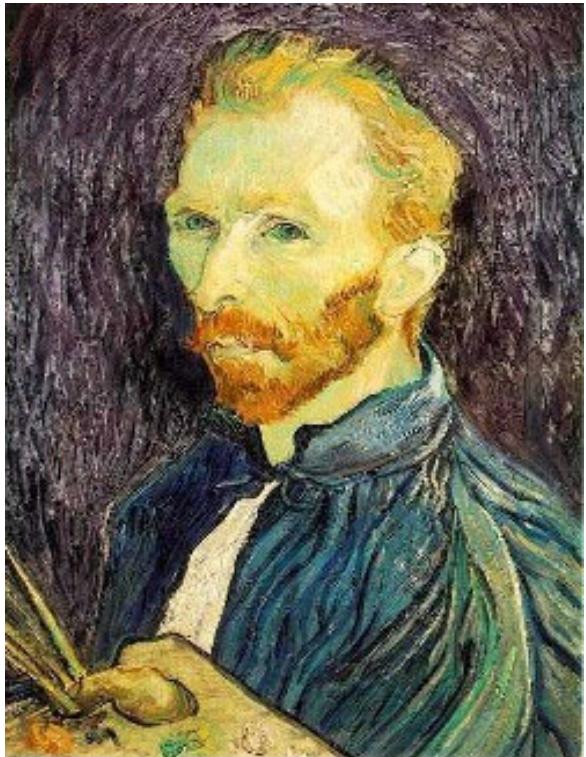
**G 1/4**



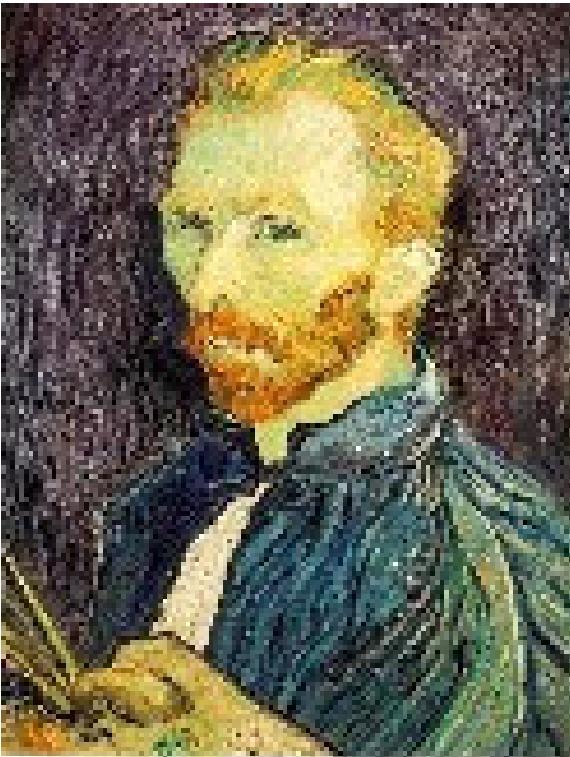
**G 1/8**



## Compare with...



**1/2**



**1/4 (2x zoom)**

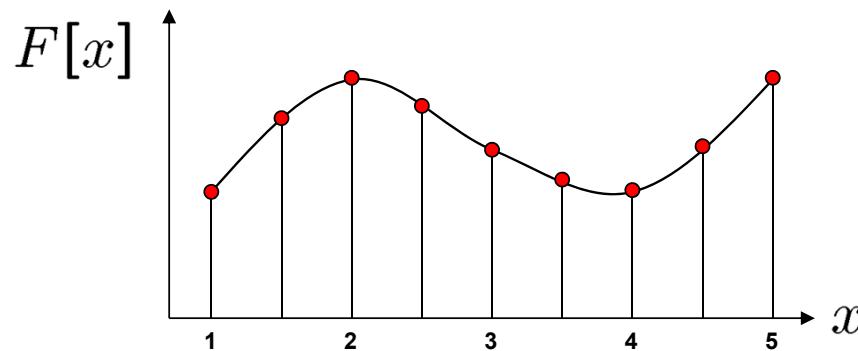


**1/8 (4x zoom)**



# Image Resampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?



- **Recall how a digital image is formed**

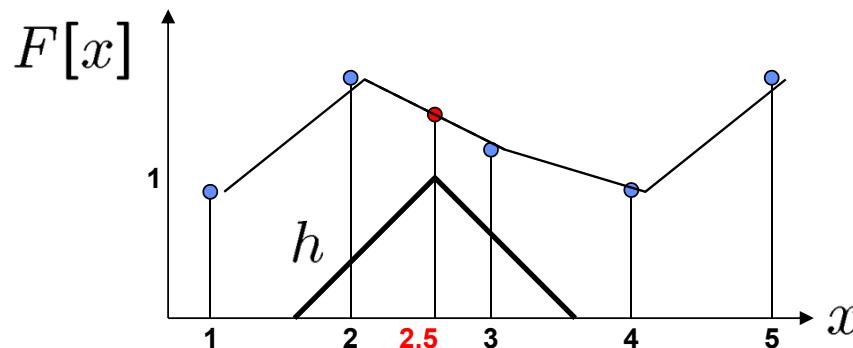
$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- **It is a discrete point-sampling of a continuous function**
- **If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale**



# Image Resampling

- So what to do if we don't know  $f$ 
  - Answer: guess an approximation  $\tilde{f}$
  - Can be done in a principled way: filtering

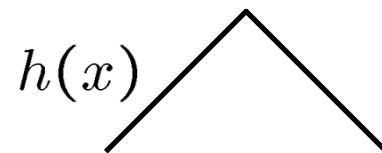


- Image reconstruction
  - Convert  $F$  to a continuous function
$$f_F(x) = F\left(\frac{x}{d}\right) \text{ when } \frac{x}{d} \text{ is an integer, 0 otherwise}$$
  - Reconstruct by convolution:
$$\tilde{f} = h \otimes f_F$$

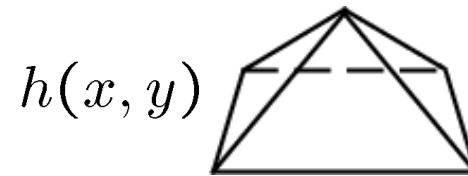


# Resampling filters

- What does the 2D version of this hat function look like?



performs  
linear interpolation



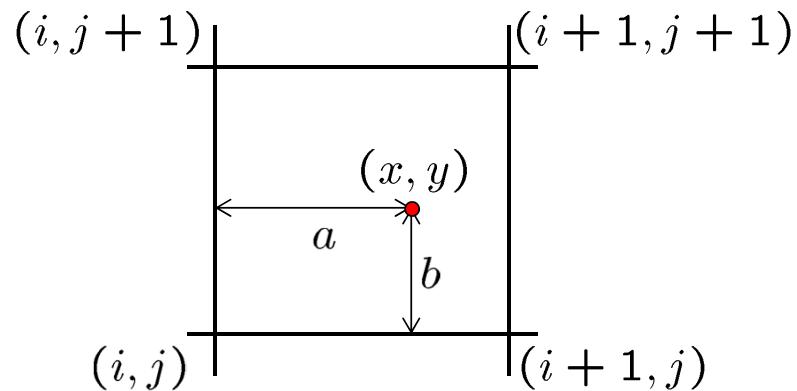
performs  
bilinear interpolation

- **Better filters give better resampled images**
  - Bicubic is common choice



# Bilinear interpolation

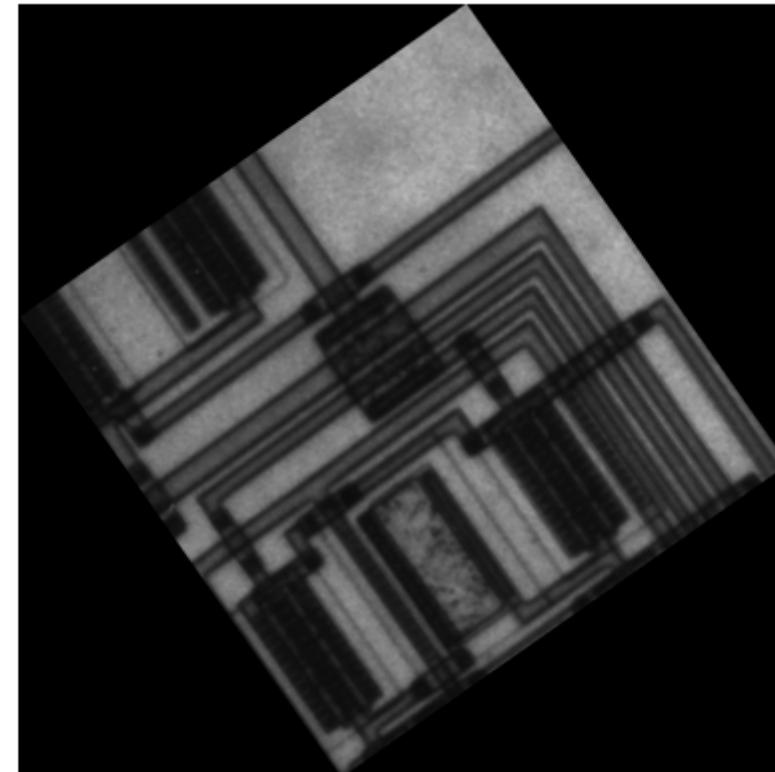
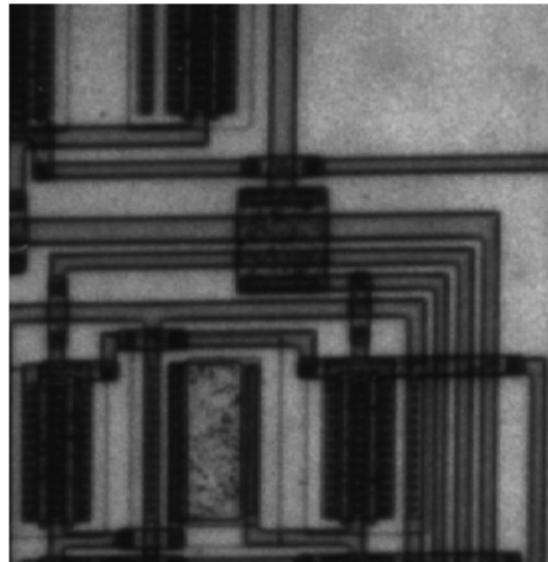
- A common method for resampling images



$$\begin{aligned} F(x, y) = & (1 - a)(1 - b) F(i, j) \\ & + a(1 - b) F(i + 1, j) \\ & + ab F(i + 1, j + 1) \\ & + (1 - a)b F(i, j + 1) \end{aligned}$$



# Image Rotation





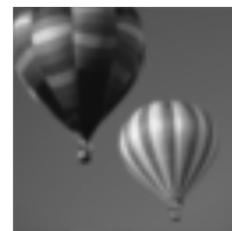
## Multi-Resolution Image Representation

- Fourier domain tells us “what” (frequencies, sharpness, texture properties), but not “where”.
- Spatial domain tells us “where” (pixel location) but not “what”.
- We want a image representation that gives a local description of image “events” – what is happening where.
- Naturally, think about representing images across varying scales.



# Multi-resolution Image Pyramids

**Low resolution**

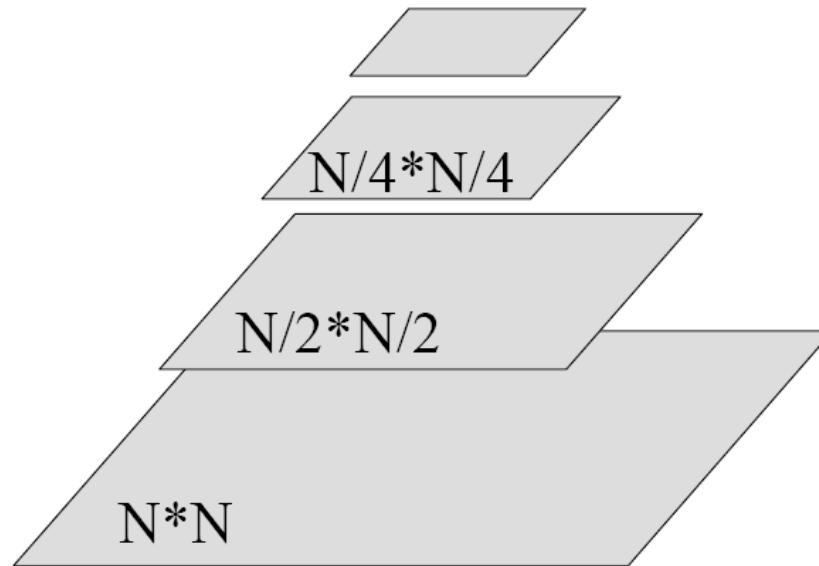


↑

**High resolution**



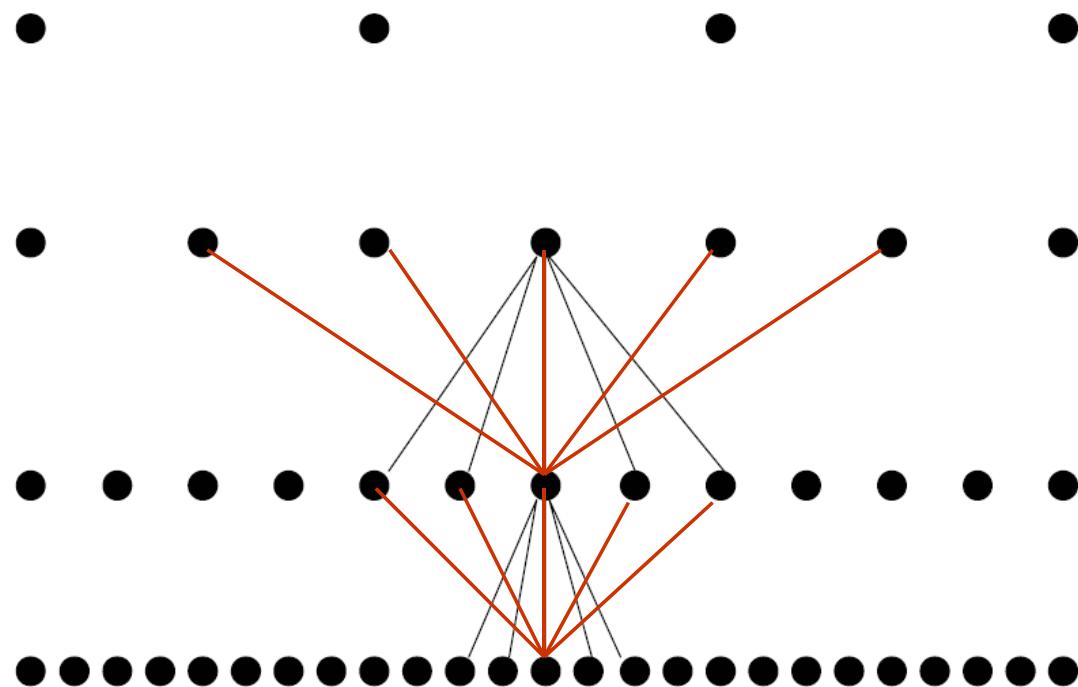
# Space Required for Pyramids

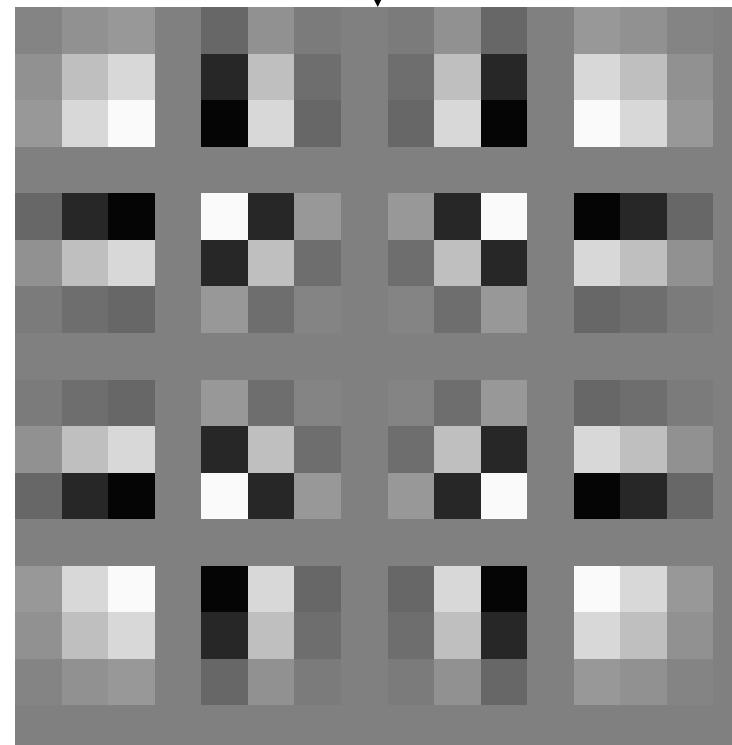
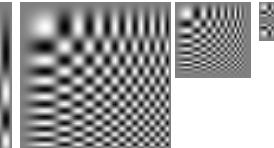
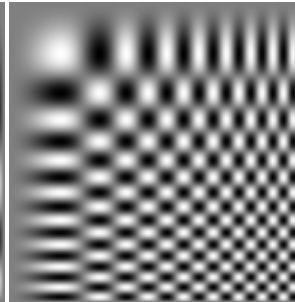
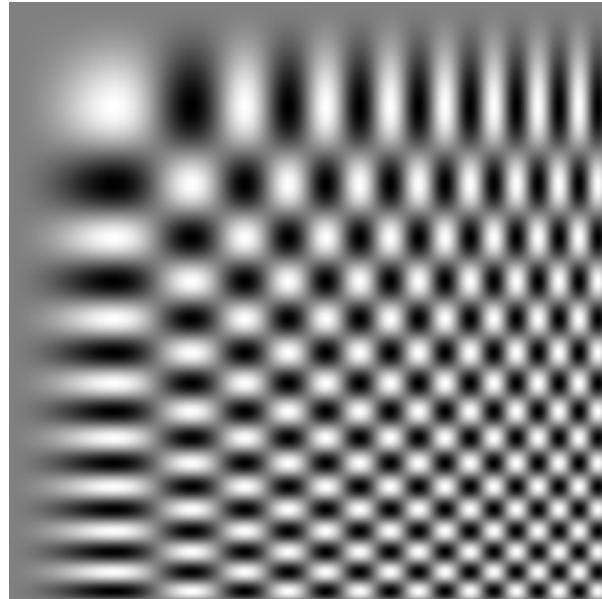


$$N^2 + \frac{1}{4}N^2 + \frac{1}{16}N^2 + N^2 + \dots = 1\frac{1}{3}N^2$$



# Pyramid Construction





**Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer**



## Even worse for synthetic images



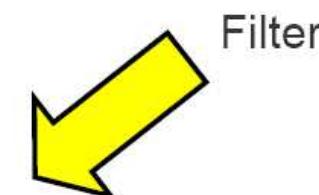


# Downsampling

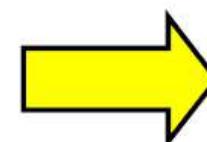


\*

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1



Subsample

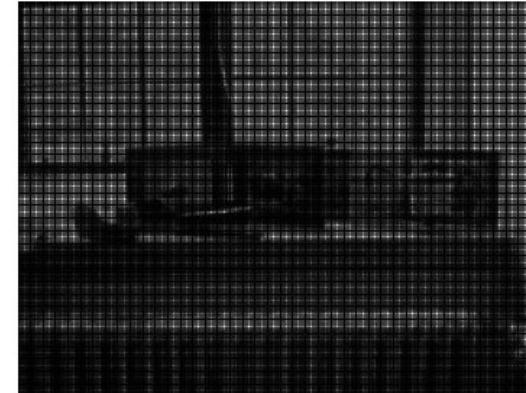




# Expansion



Expand  
Image



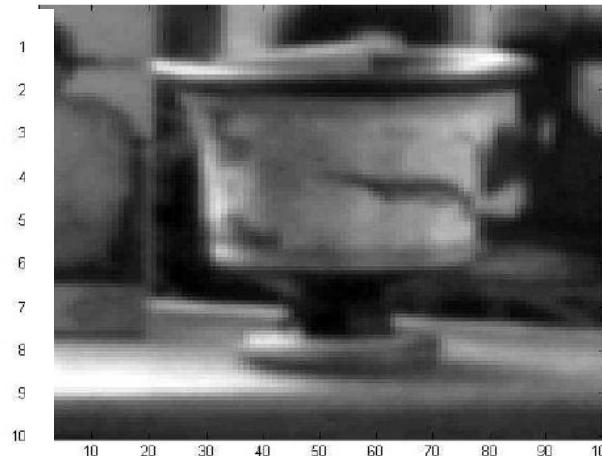
Interpolation



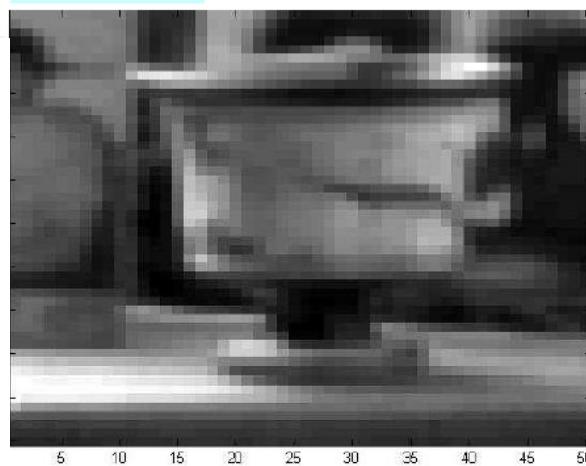


# Interpolation Results

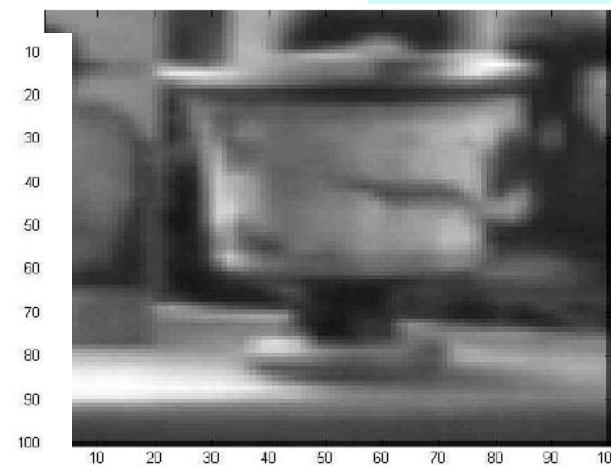
Original  
Image



Nearest  
Neighbor



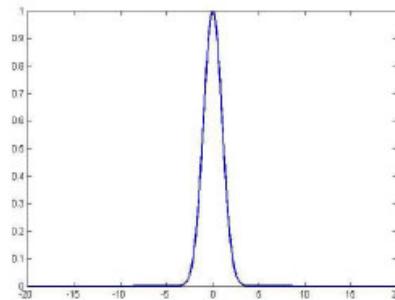
Bilinear  
Interpolation



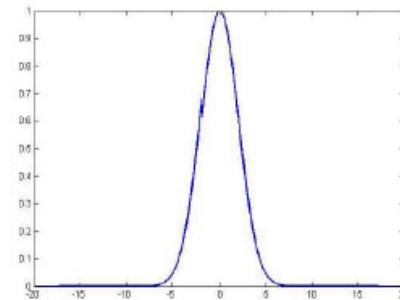


# The Gaussian Pyramid

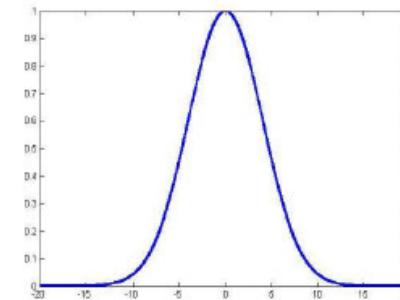
- Smooth with Gaussians because
  - a Gaussian\*Gaussian=another Gaussian
- Synthesis
  - smooth and downsample
- Gaussians are low pass filters, so repetition is redundant
- Kernel width doubles with each level



Level 1



Level 2



Level 3

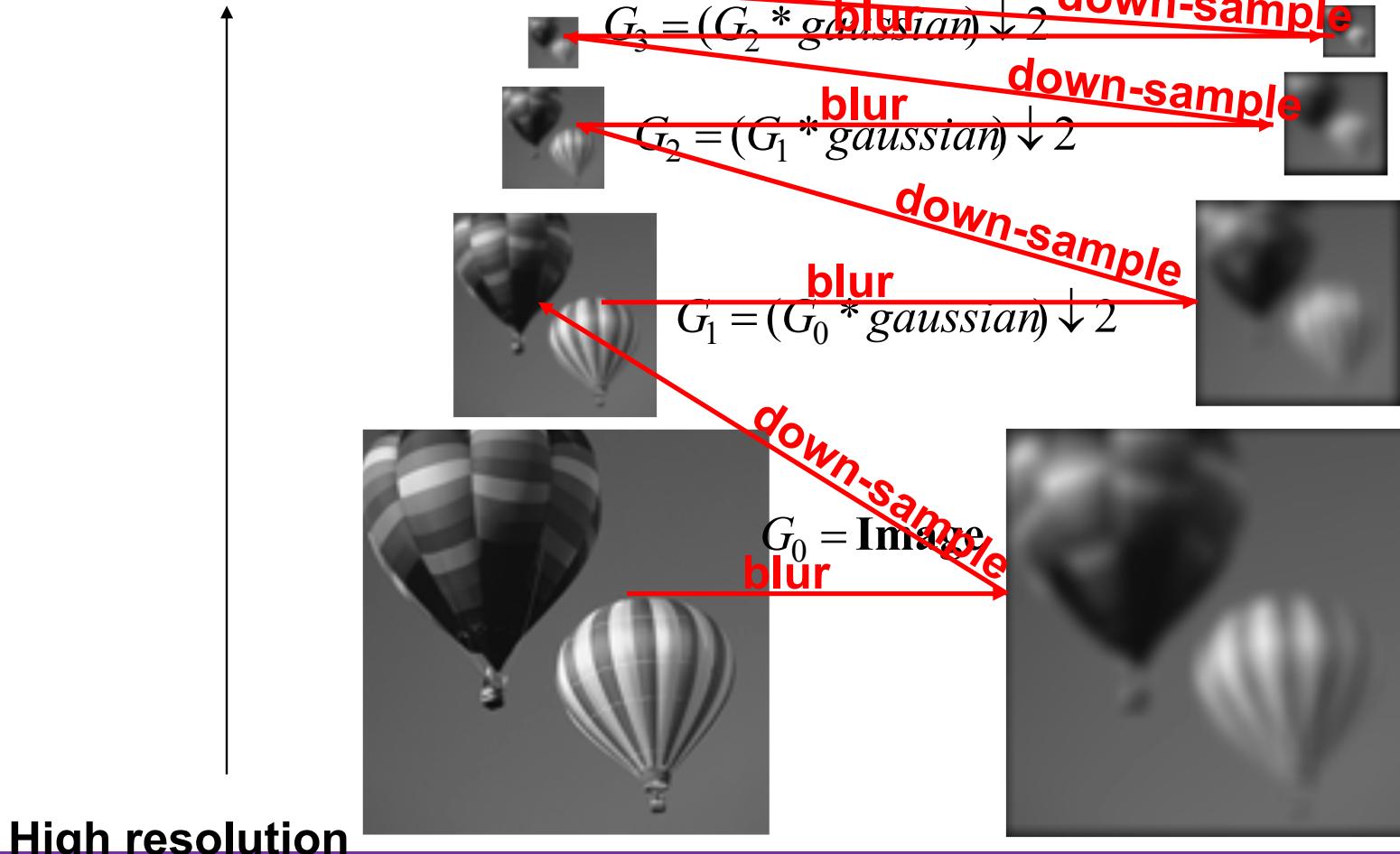


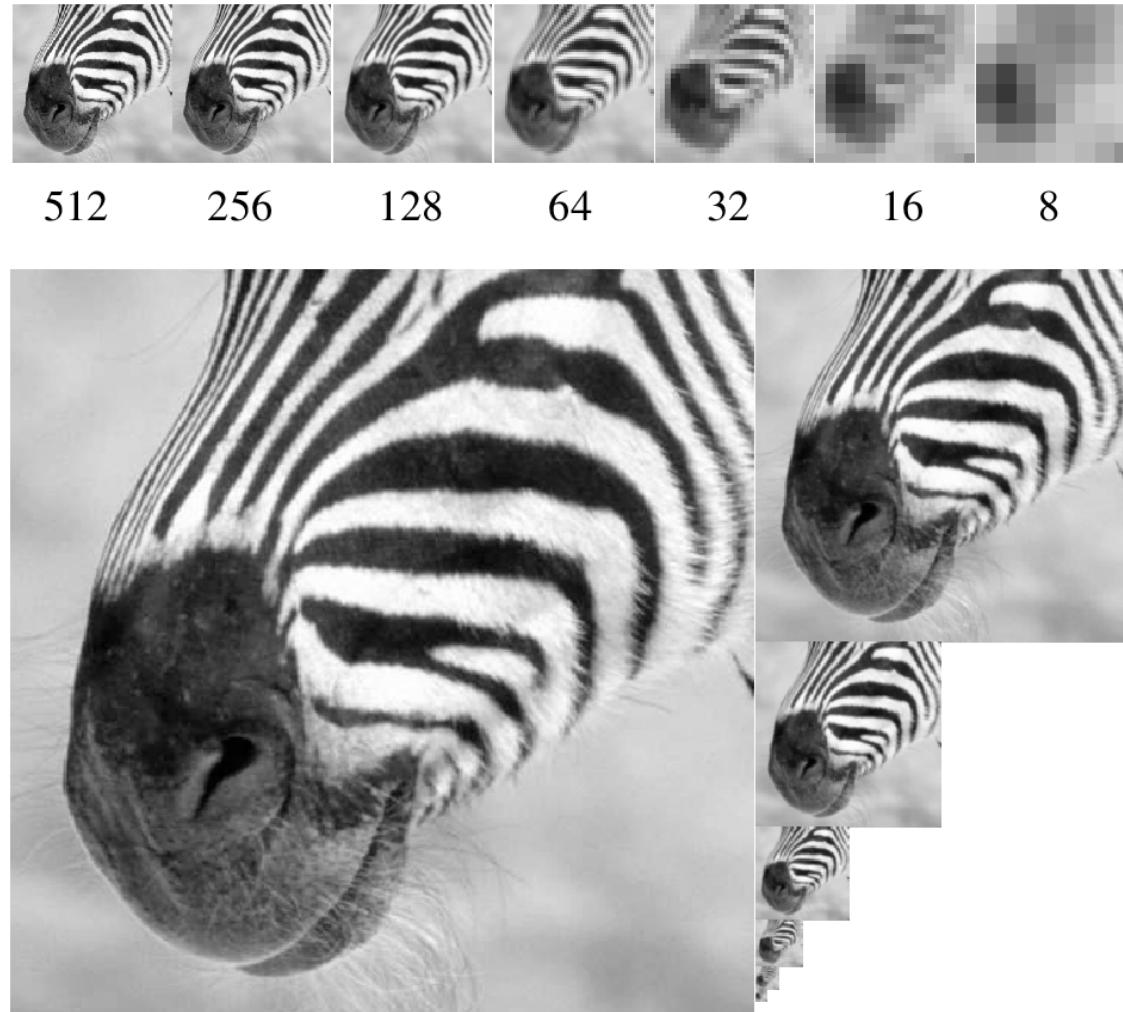
## Smoothing as low-pass filtering

- High frequencies lead to trouble with sampling.
- Suppress high frequencies before sampling !
  - truncate high frequencies in FT
  - or convolve with a low-pass filter
- Common solution: use a Gaussian
  - multiplying FT by Gaussian is equivalent to convolving image with Gaussian.



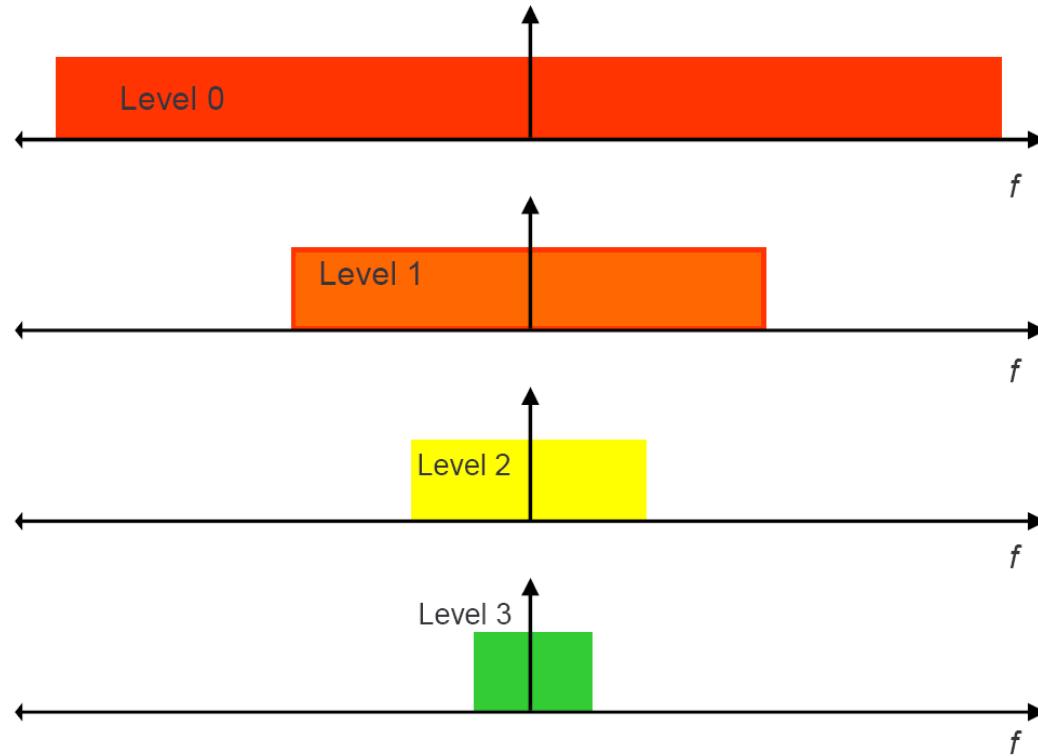
**Low resolution**







## Gaussian Pyramid Frequency Composition





# Pyramids at Same Resolution





## Difference of Gaussians (DoG)

- Laplacian of Gaussian can be approximated by the difference between two different Gaussians

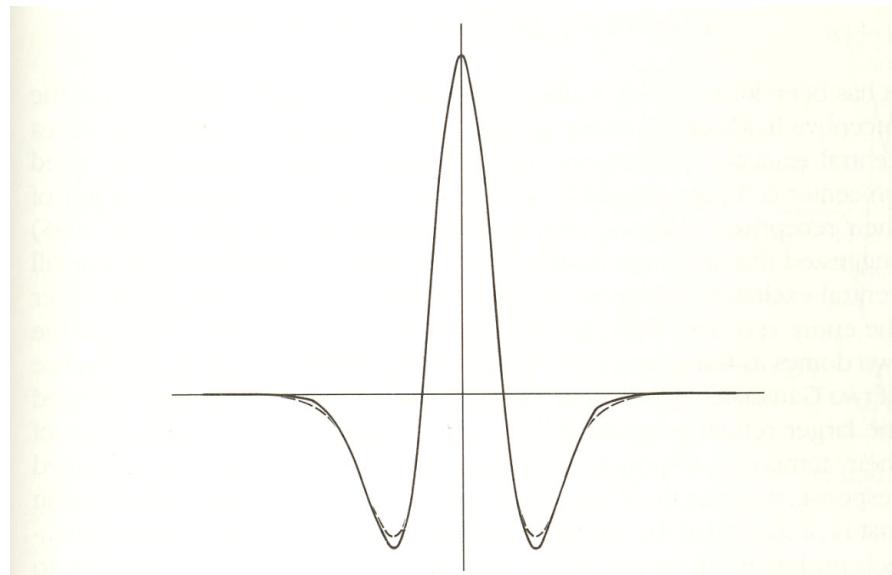
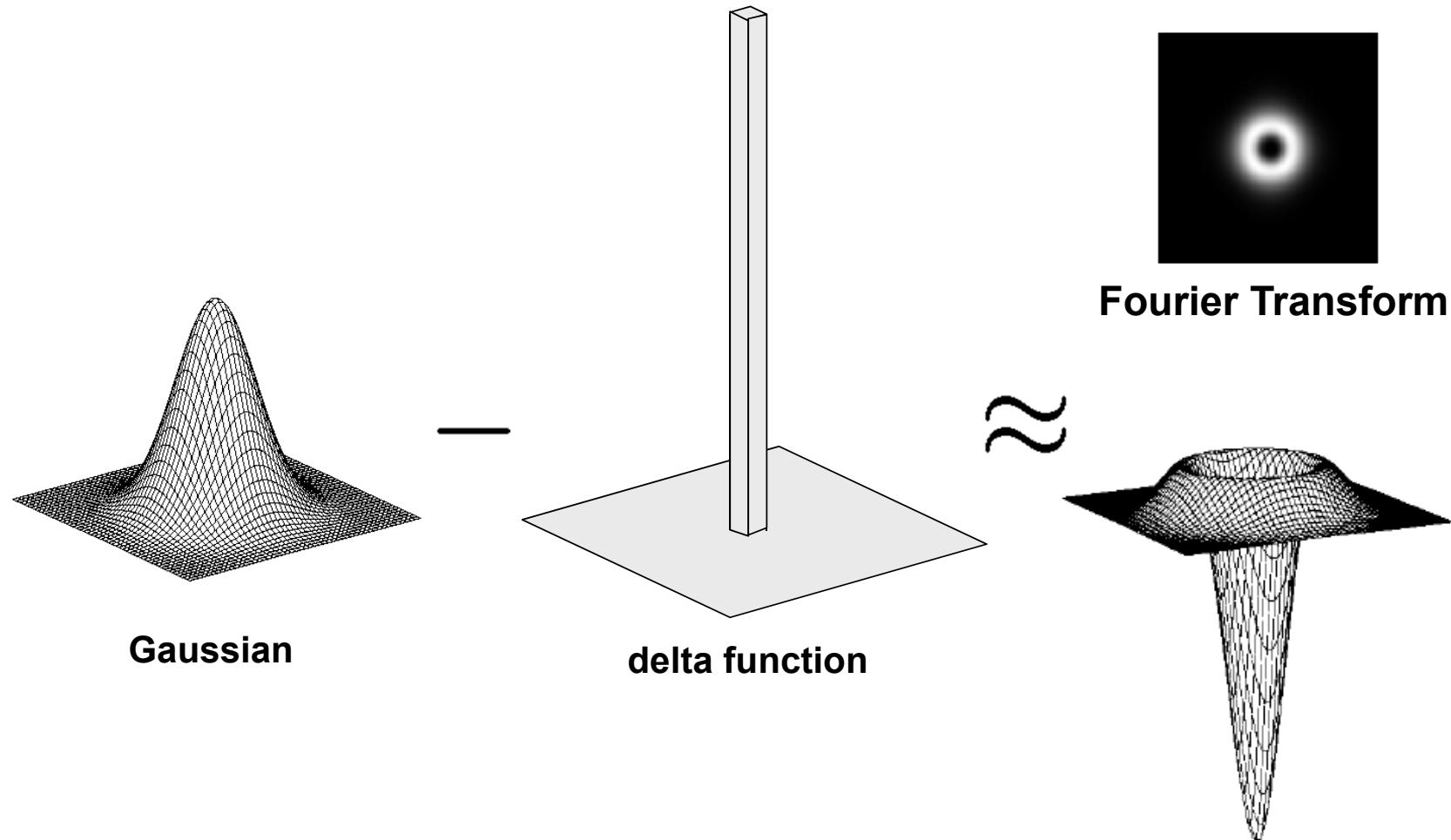
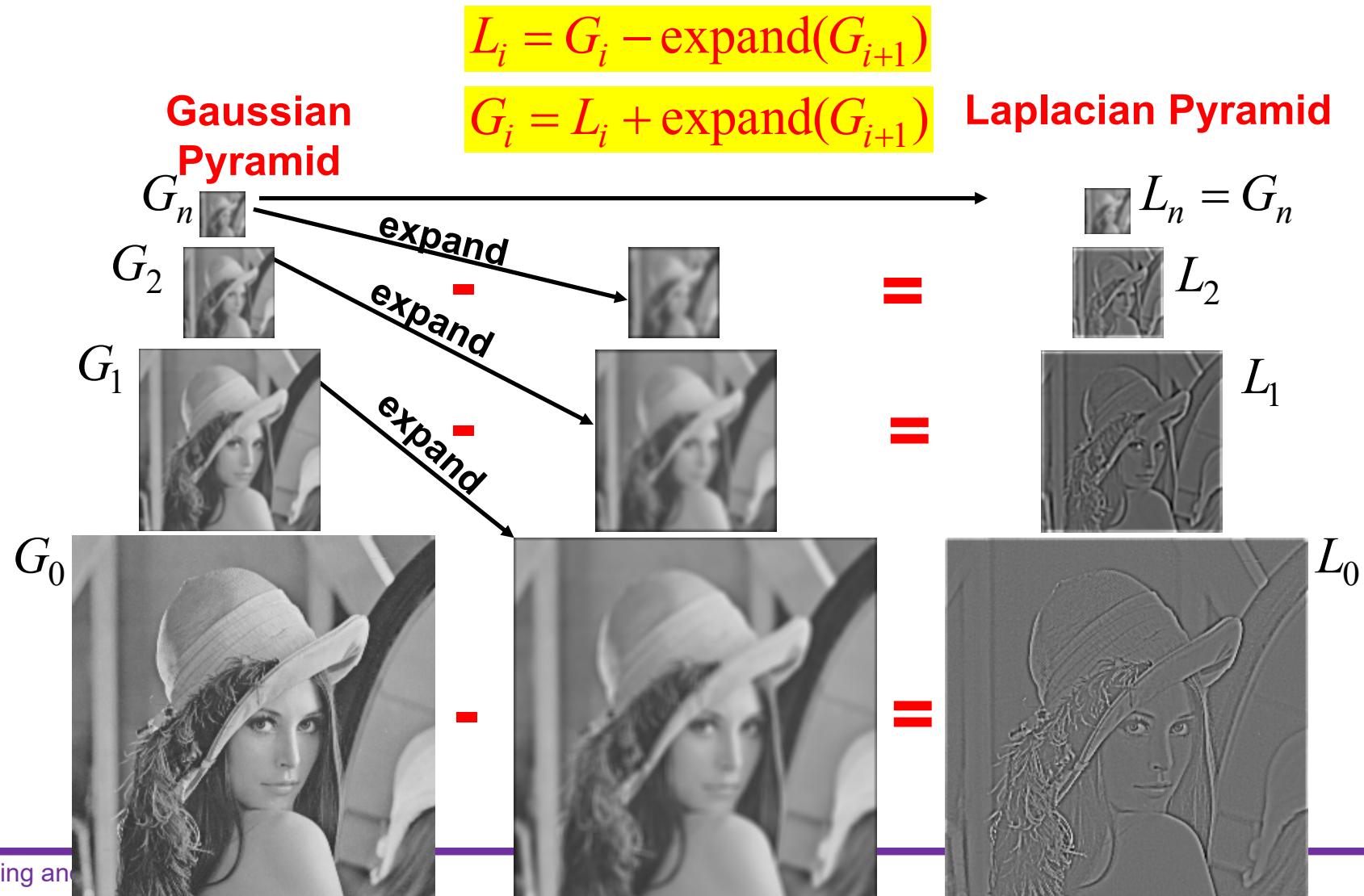


Figure 2–16. The best engineering approximation to  $\nabla^2 G$  (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar. (Reprinted by permission from D. Marr and E. Hildreth, “Theory of edge detection,” *Proc. R. Soc. Lond. B* 204, pp. 301–328.)



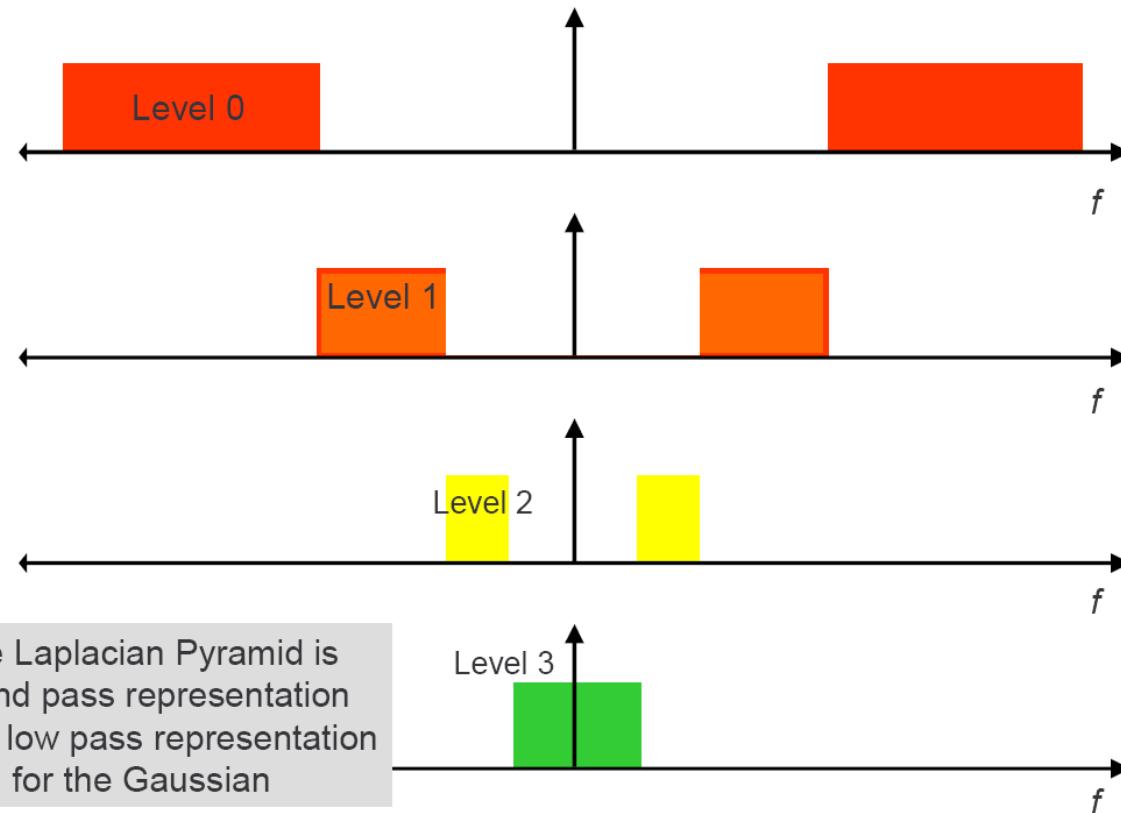
# Gaussian – Image filter







## Laplacian Pyramid Frequency Composition





# Applications of Image Pyramids

- Coarse-to-Fine strategies for computational efficiency.
- Search for correspondence
  - look at coarse scales, then refine with finer scales
- Edge tracking
  - a “good” edge at a fine scale has parents at a coarser scale
- Control of detail and computational cost in matching
  - e.g. finding stripes
  - very important in texture representation
- Image Blending and Mosaicing
- Data compression



# Edge Detection using Pyramids

**Coarse-to-fine strategy:**

- Do edge detection at higher level.
- Consider edges of finer scales only near the edges of higher scales.

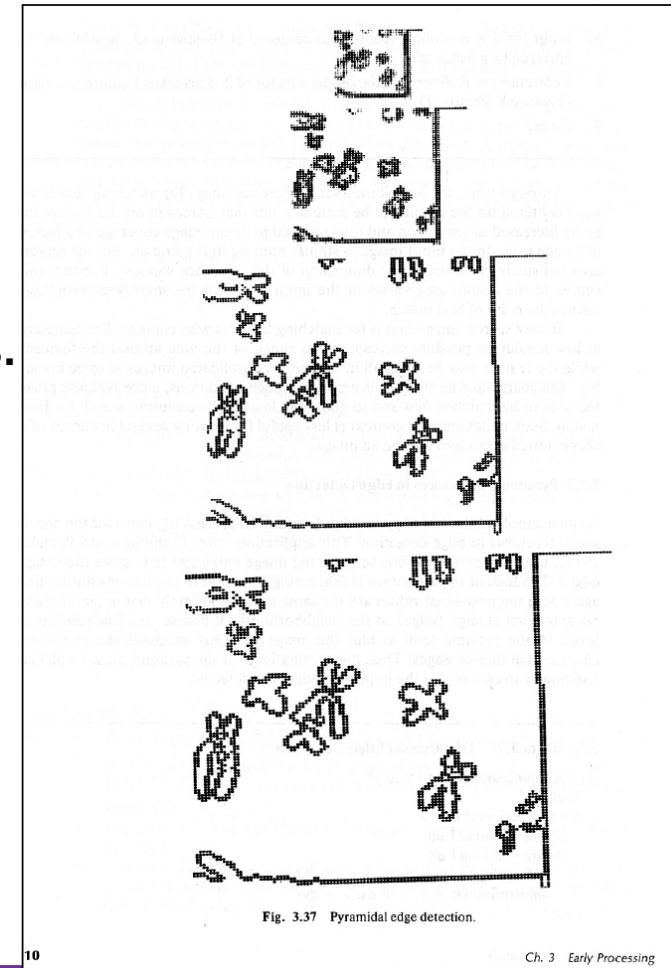


Fig. 3.37 Pyramidal edge detection.

10

Ch. 3 Early Processing

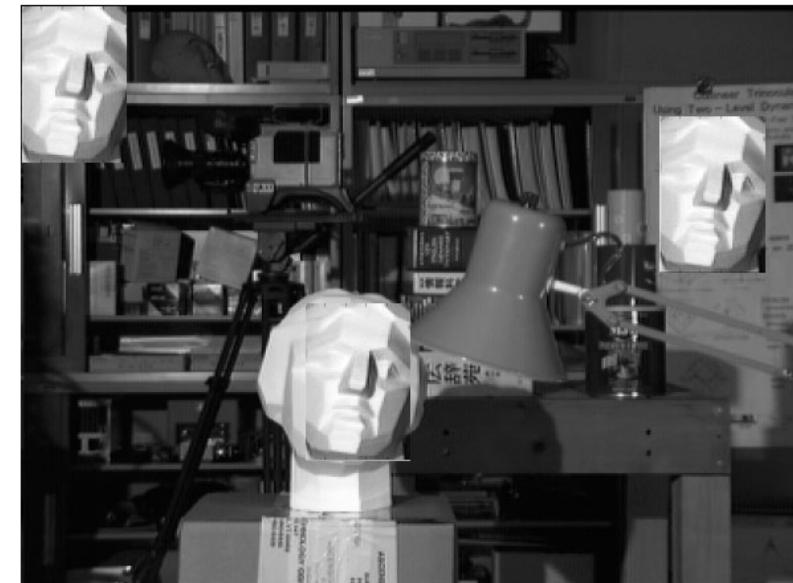


# Fast Template Matching

*Template*



*Search Region*



- For an  $m \times n$  image...
- For a  $p \times q$  template...
- The complexity of the 2D pattern recognition task is  $O(mnpq)$  😞
- This gets even worse for a family of templates (e.g., to address scale and/or rotational effects)



# Fast Template Matching

*Template*



*Search Region*

Original Image



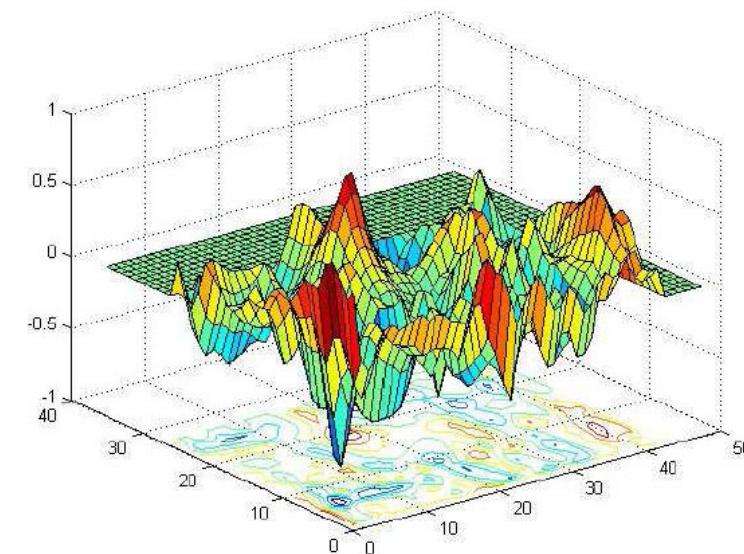
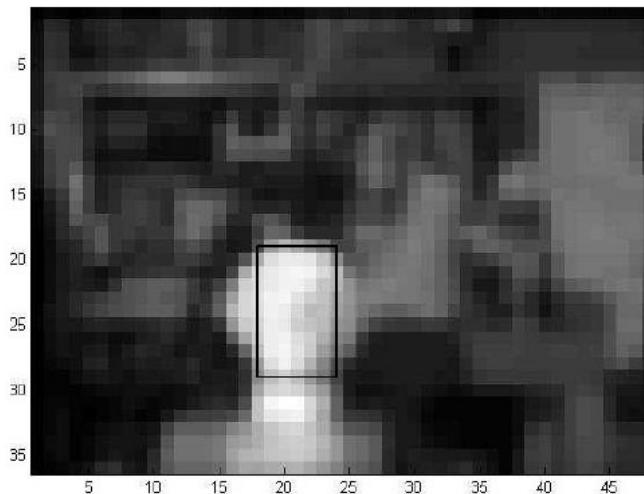


- Multi-resolution template matching
  - reduce resolution of both template and image by creating an **image pyramid**
  - match small template against small image
  - identify locations of strong matches
  - expand the image and template, and match higher resolution template selectively to higher resolution image
  - iterate on higher and higher resolution images
- Issue:
  - how to choose detection thresholds at each level
    - too low will lead to too much cost
    - too high will miss match



## Level 3 Search

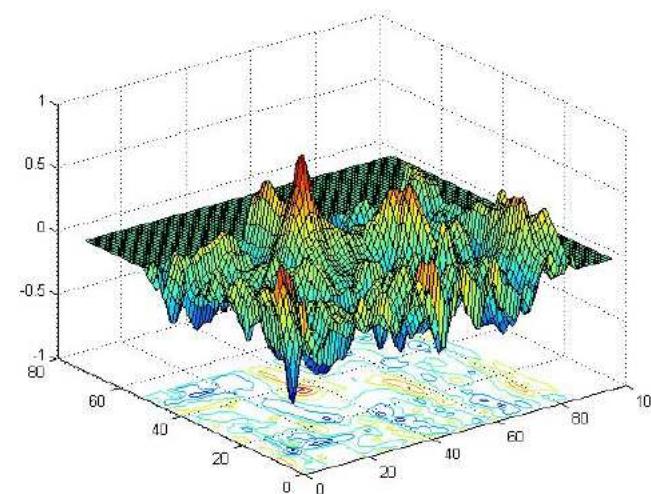
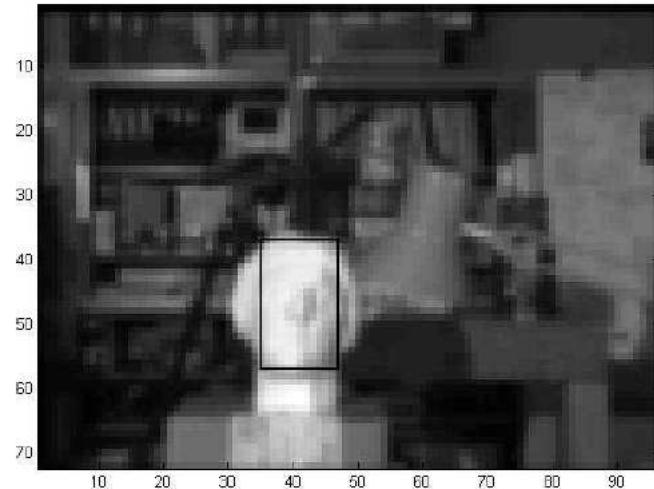
- At the lowest pyramid level, we search the entire image with the correlation template





## Level 2 Search

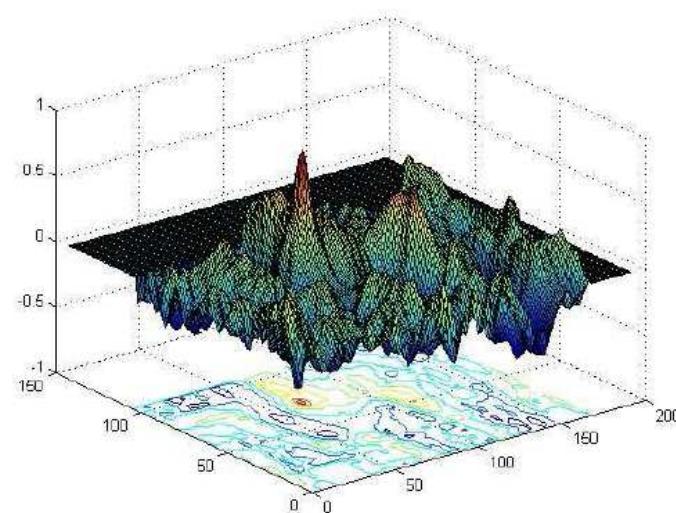
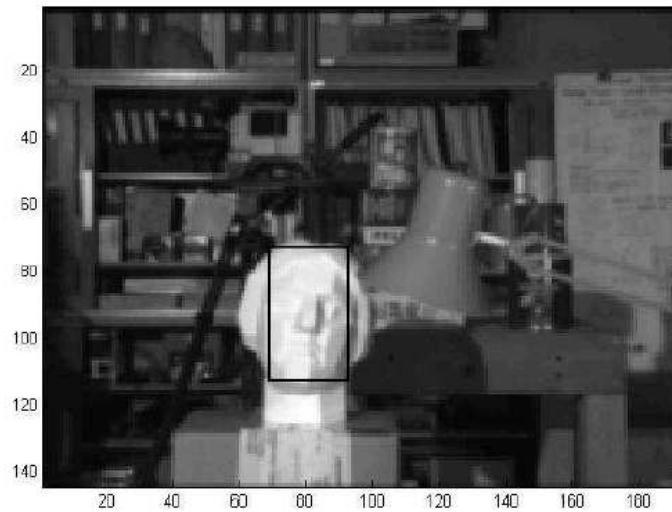
- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions





## Level 1 Search

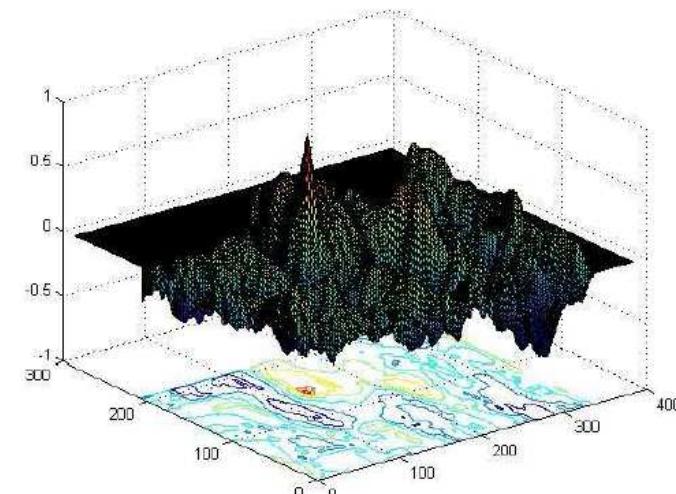
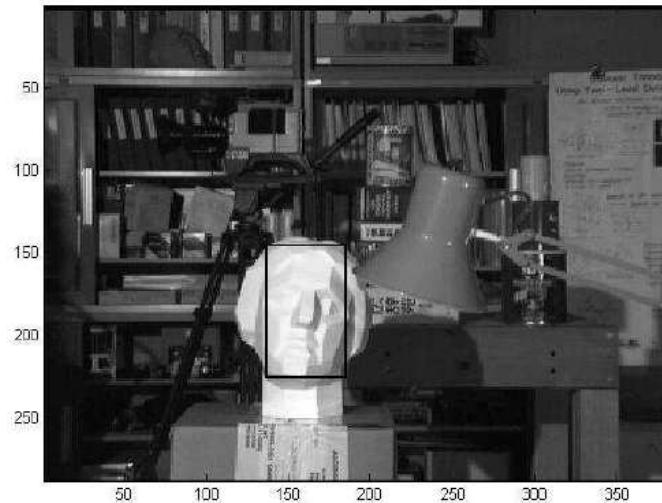
- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions





## Level 0 Search

- In the end, the total time (in Matlab) was reduced from  $\approx 31$  seconds to  $\approx 0.5$  seconds while obtaining the same template match





# Image Blending and Mosaicing

- Given two images  $A$  and  $B$
- Construct Laplacian Pyramid  $L_a$  and  $L_b$
- Create a third Laplacian Pyramid  $L_c$  where for each level  $l$

$$L_c(i, j) = \begin{cases} L_a(i, j) & \text{if } i < \text{width}/2 \\ (L_a(i, j) + L_b(i, j))/2 & \text{if } i = \text{width}/2 \\ L_b(i, j) & \text{if } i > \text{width}/2 \end{cases}$$

- Sum all levels  $L_c$  in to get the blended image

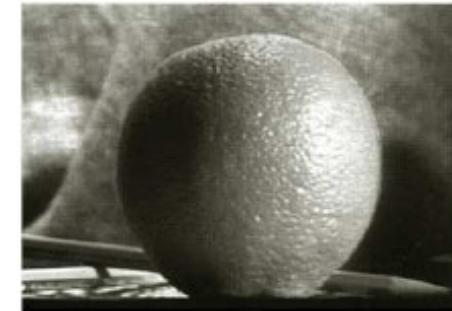
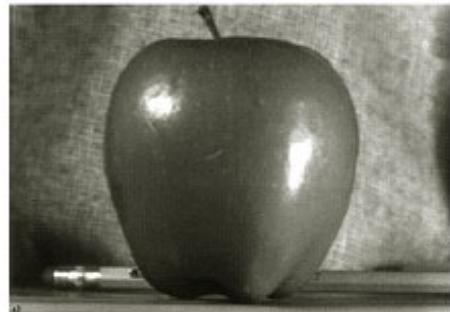
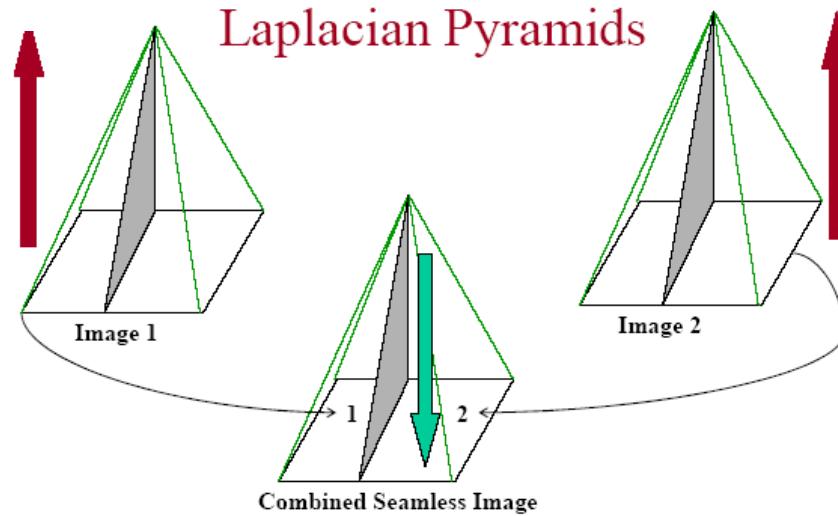
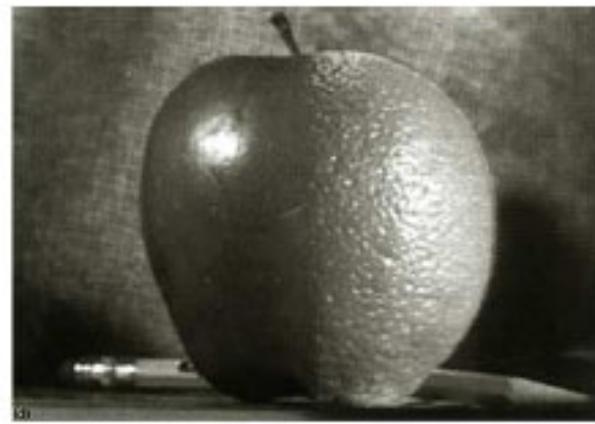
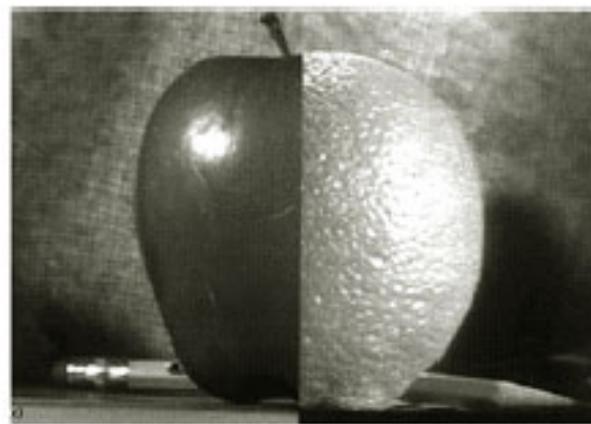
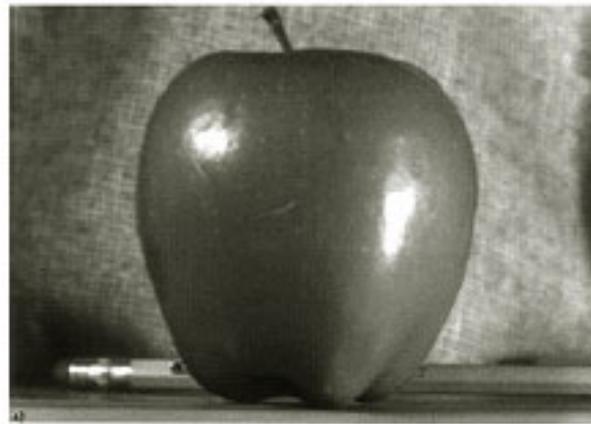


Image Merging with  
Laplacian Pyramids





# Blending Apples and Oranges



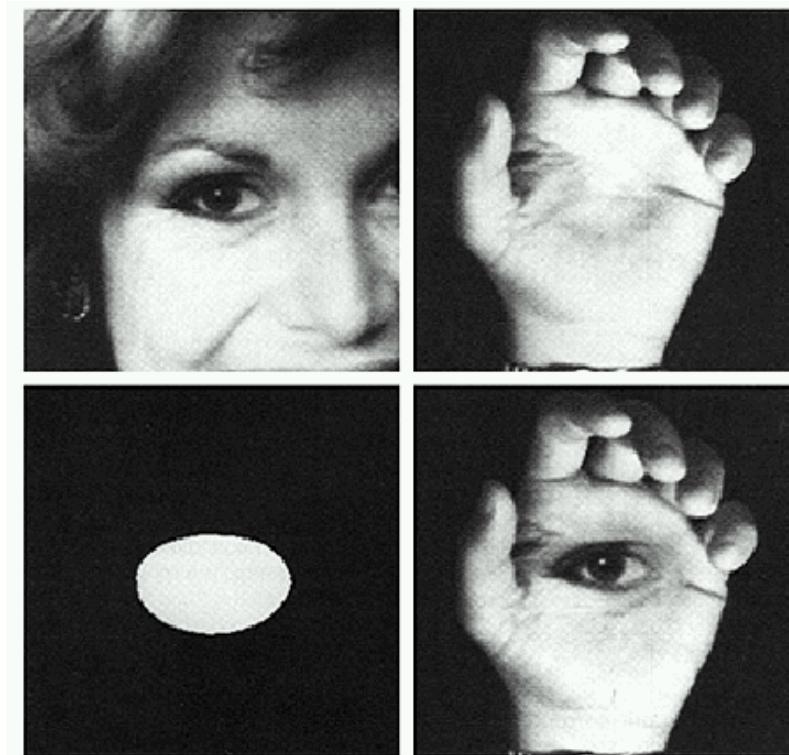


# Pyramid blending of Regions

- Given two images  $A$  and  $B$ , and a mask  $M$
- Construct Laplacian Pyramids  $L_a$  and  $L_b$
- Construct a Gaussian Pyramid  $G_m$
- Create a third Laplacian Pyramid  $L_c$  where for each level  $l$

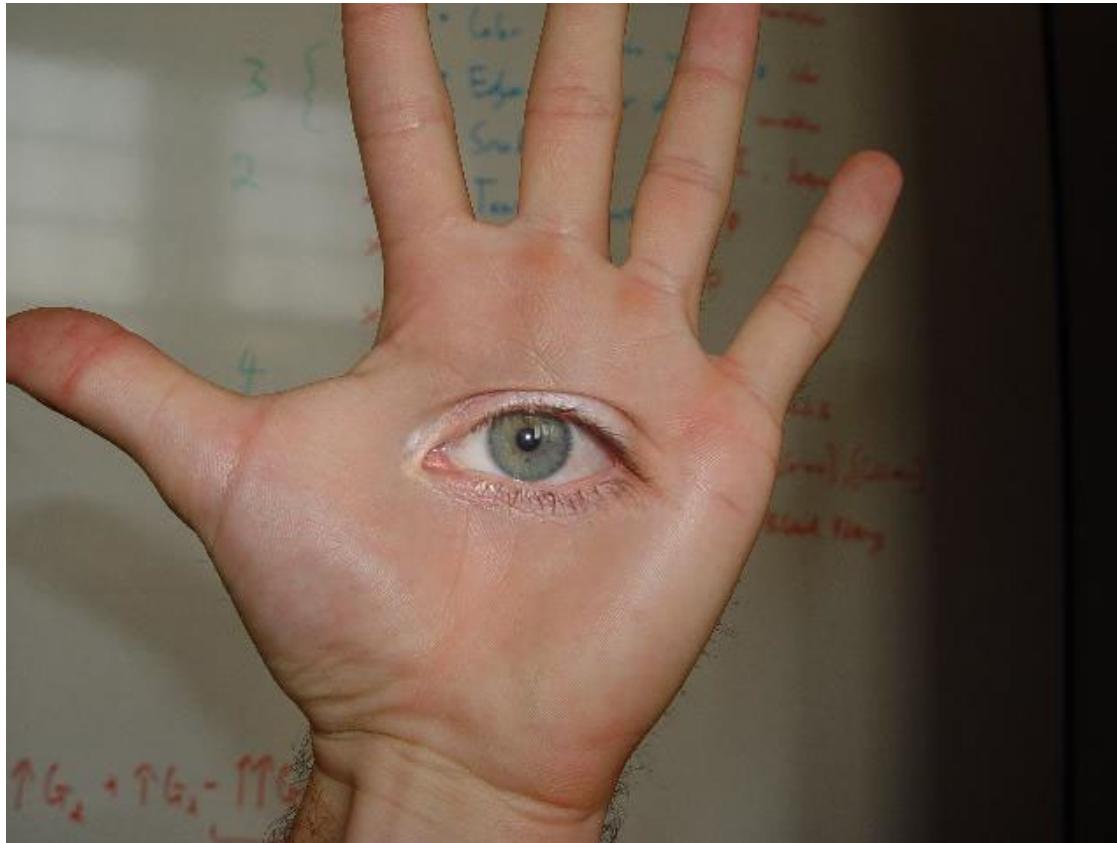
$$L_c(i, j) = G_m(i, j)L_a(i, j) + (1 - G_m(i, j))L_b(i, j)$$

- Sum all levels  $L_c$  in to get the blended image



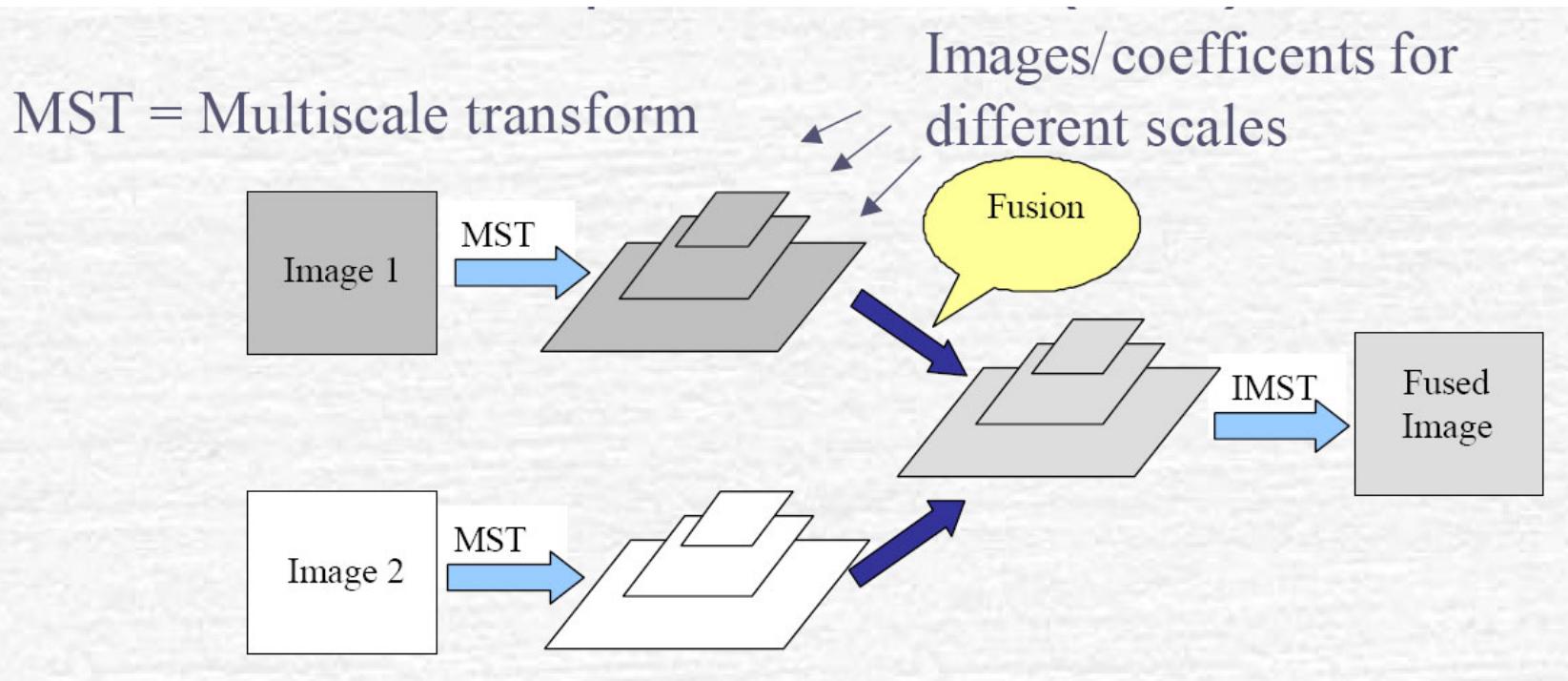


# Horror Photo





# Image Fusion



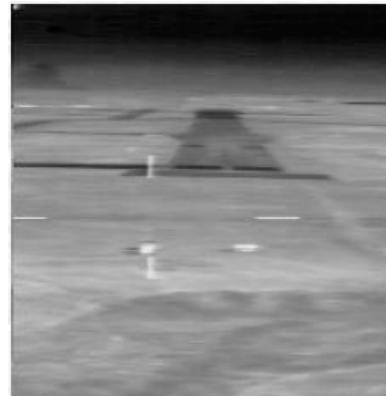
**Multi-scale Transform (MST)**

= Obtain Pyramid from Image

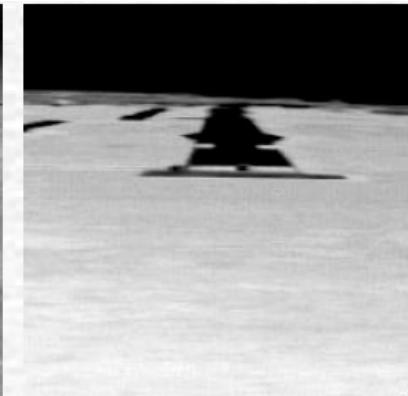
**Inverse Multi-scale Transform (IMST)** = Obtain Image from Pyramid



# Multi-Sensor Fusion

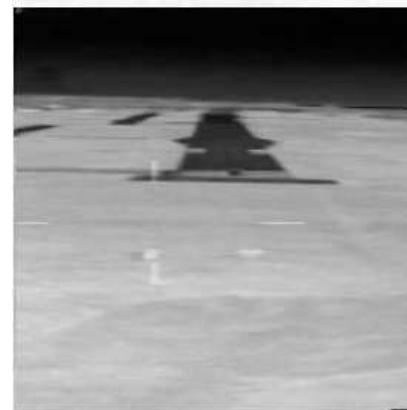


Long Wave



Medium Wave

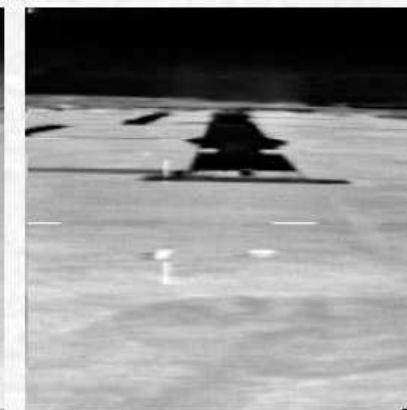
**Matlab  
Impyramid()**



Averaging



Selecting Maximum

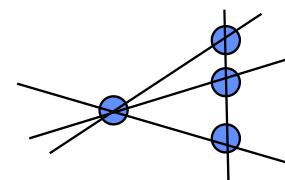


Laplacian Fusion



## Global Processing via the Hough Transform

- Here we consider linking points by determining whether they lie on a specified curve or line
- This processing considers global relationships between pixels
- Suppose for  $n$  points in an image, we want to find subsets of these points that lie on straight lines
- One possible solution is to first find all lines determined by every pair of points and then find subsets of points associated with each line
- Involves  $n(n-1)/2 \sim n^2$  lines and  $(n)(n(n-1))/2 \sim n^3$  comparisons
- prohibitively expensive in all but trivial situations!



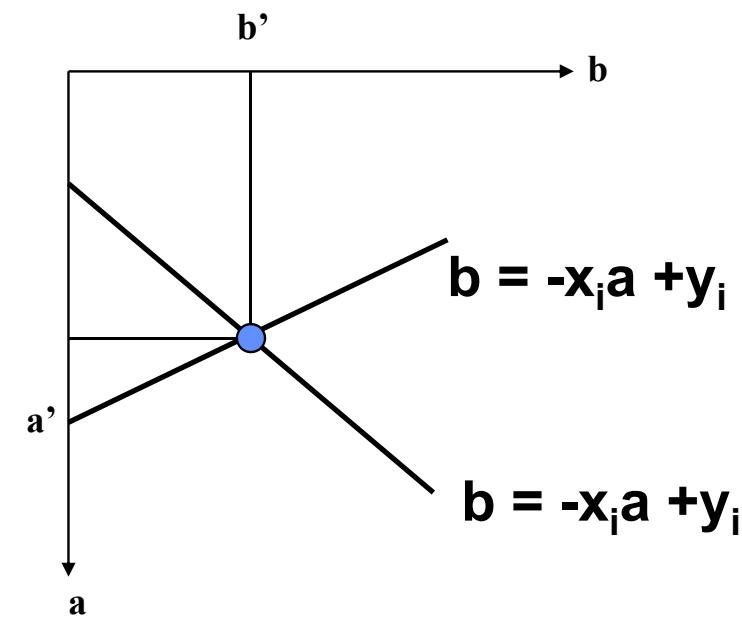
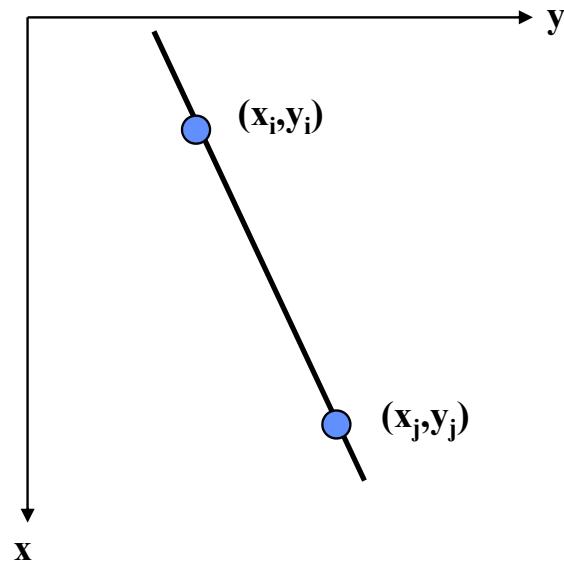


## Hough Transform

- Hough (1962) proposed an alternative referred to as the *Hough transform*
- Consider a point  $(x_i, y_i)$  and the general equation of a straight line  $y_i = a x_i + b$
- Infinitely many lines pass through  $(x_i, y_i)$  but they all satisfy  $y_i = a x_i + b$  for varying values of  $a$  and  $b$
- However, expressing this equation as  $b = -x_i a + y_i$  and considering the  $(a, b)$  parameter plane yields the equation of a single line for a fixed pair  $(x_i, y_i)$
- Further a second point  $(x_j, y_j)$  also has a line associated with it and the two lines intersect at  $(a', b')$



## XY and Parameter Planes





## Accumulator Cells

- Subdivide a and b axes in parameter space into bins and set bin value to zero
- a and b ranges can be limited to cover only expected slopes of lines in image
- For each (x,y) point in edge image, increment the corresponding line of bins
- Peaks of Hough transform indicate groups of collinear pixels
- Reverse process to find edge pixels on straight lines

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



## Comments

- One problem with using ab space is that the slope and the intercept approach infinity as the line approaches the vertical
- A solution is to use the following representation for a line

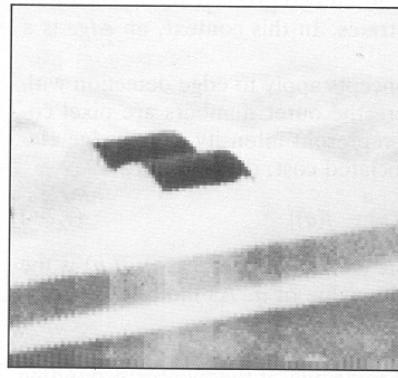
$$x \cos \theta + y \sin \theta = \rho$$

- Now points yield sinusoidal curves in parameter space
- Method can be extended to detect circles and other shapes rather than lines.



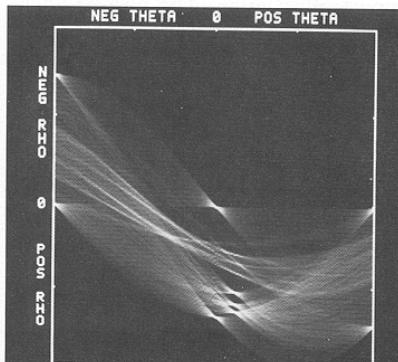
## Hough Transform Example

Original  
Infrared  
Image



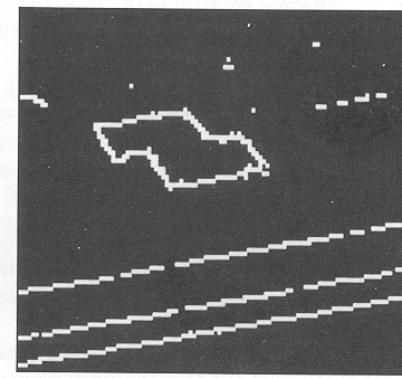
(a)

Hough  
Transform



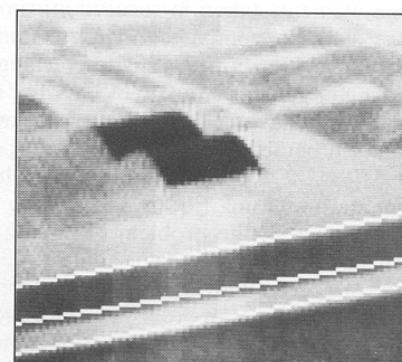
(c)

Edge Detected



(b)

Projected  
Back onto  
Image



(d)

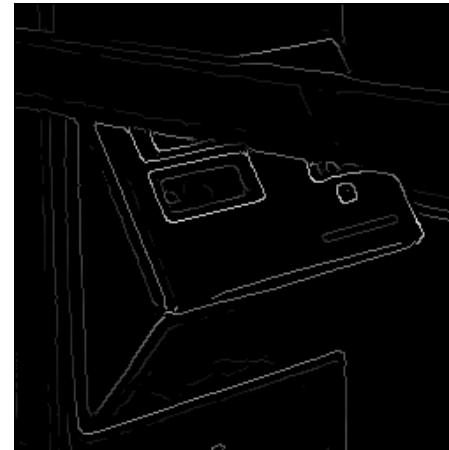
*Figure 7.19 (a) Infrared image; (b) gradient image; (c) Hough transform; (d) linked pixels.  
(Courtesy of D. R. Cate, Texas Instruments, Inc.)*



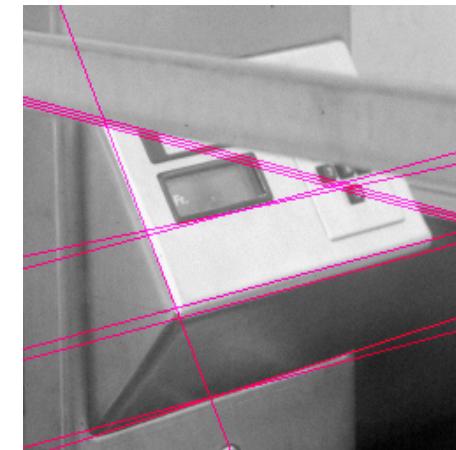
## Another Example



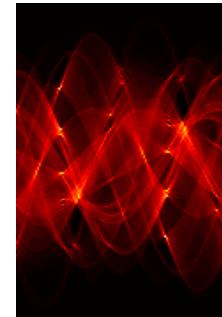
Original



Edge  
Detection



Found  
Lines



Parameter  
Space



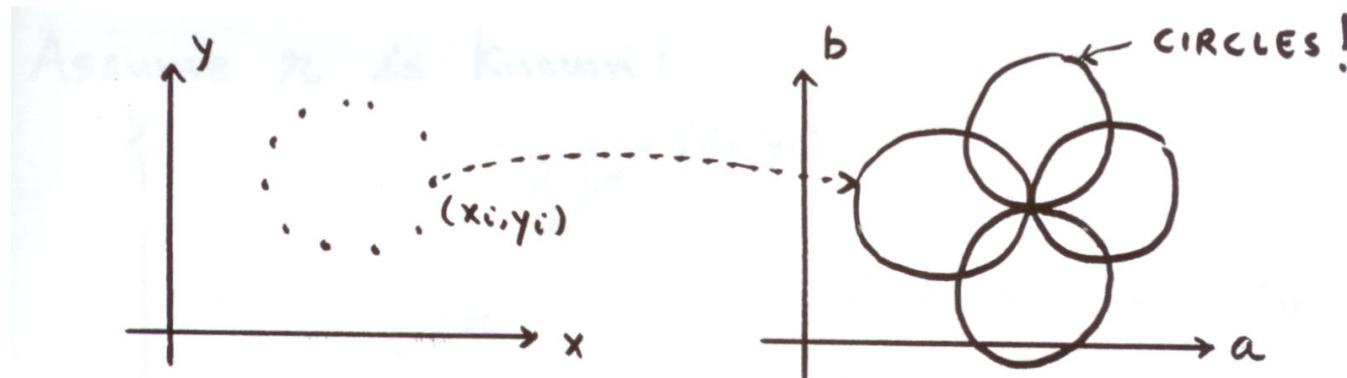
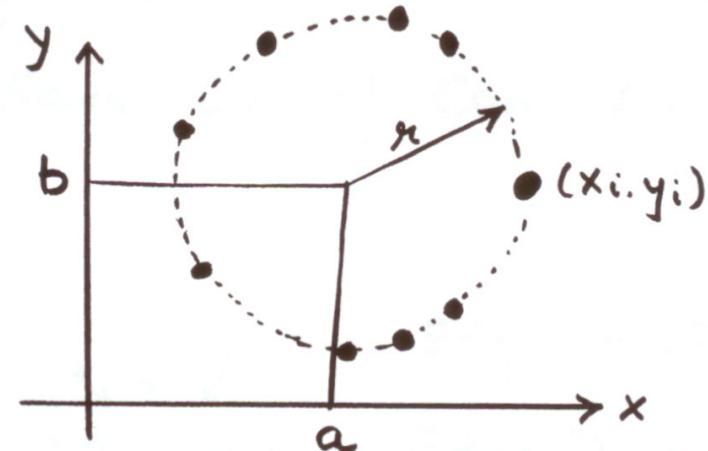
# Finding Circles by Hough Transform

Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

If radius is known: (2D Hough Space)

Accumulator Array  $A(a, b)$

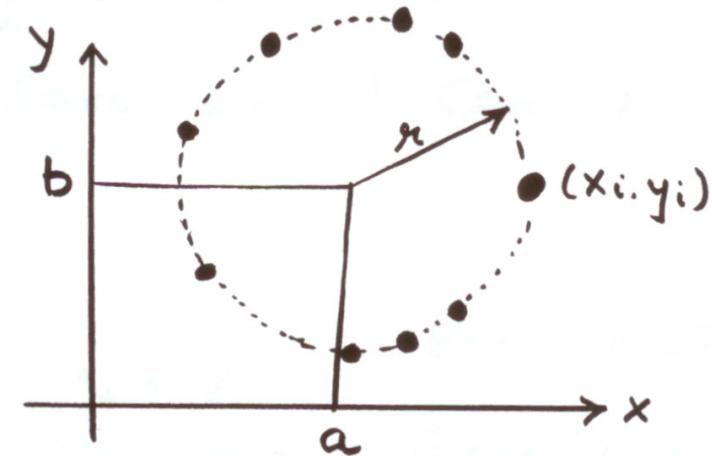




# Finding Circles by Hough Transform

**Equation of Circle:**

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$



**If radius is not known: 3D Hough Space!**

**Use Accumulator array  $A(a, b, r)$**



## Real World Circle Examples

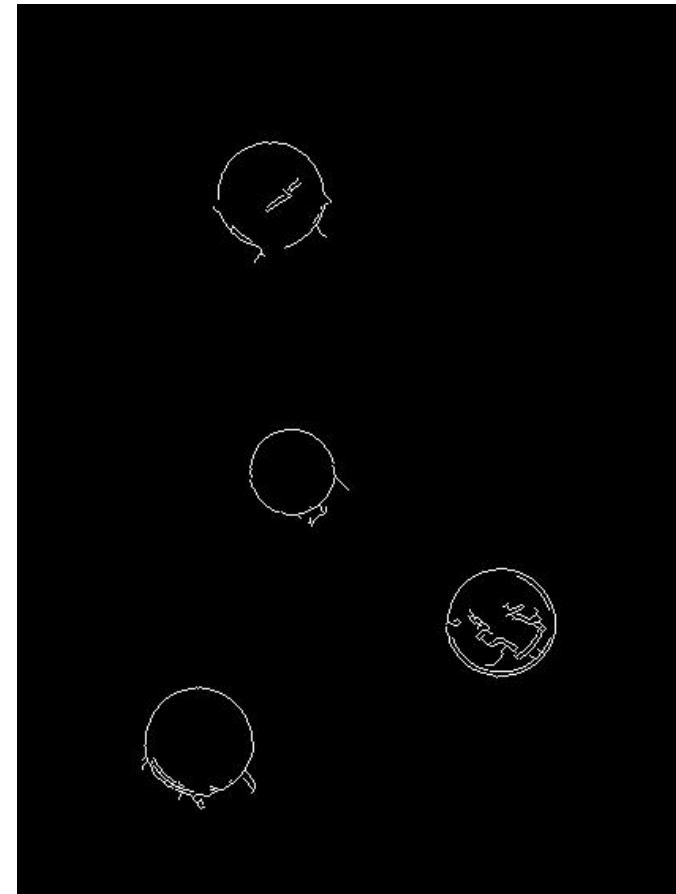


**Crosshair indicates results of Hough transform,  
bounding box found via motion differencing.**



# Finding Coins

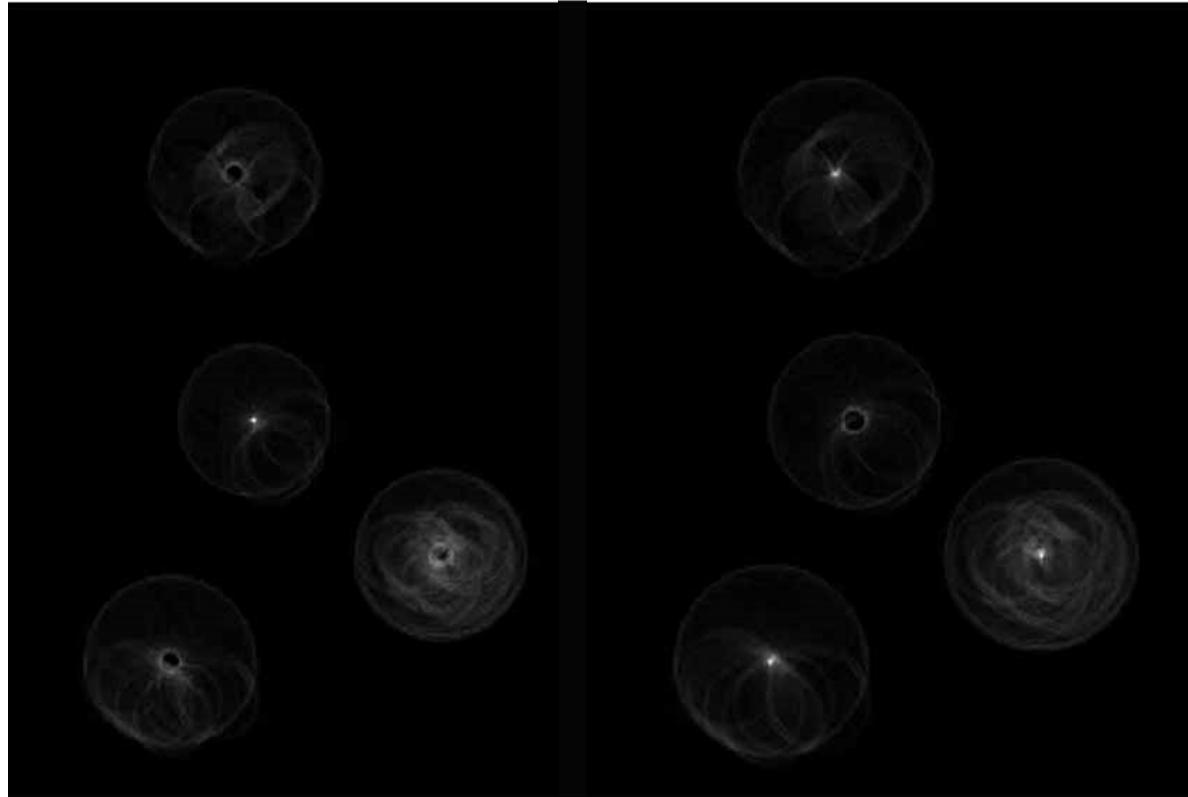
Original      Edges (note noise)





## Finding Coins (Continued)

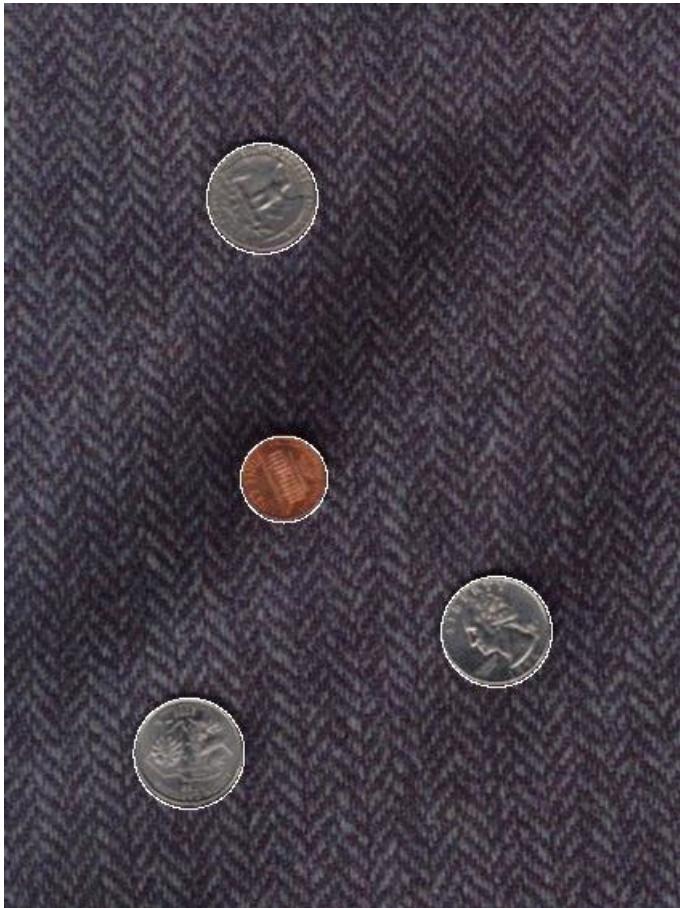
Penny



Quarter



## Finding Coins (Continued)



**Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.**



# Hough Transform: Comments

- Works on Disconnected Edges
- Relatively insensitive to occlusion
- Effective for simple shapes (lines, circles, etc)
- Often used to find symbols on scanned maps etc
- Trade-off between work in Image Space and Parameter Space
- Handling inaccurate edge locations:
  - Increment Patch in Accumulator rather than a single point



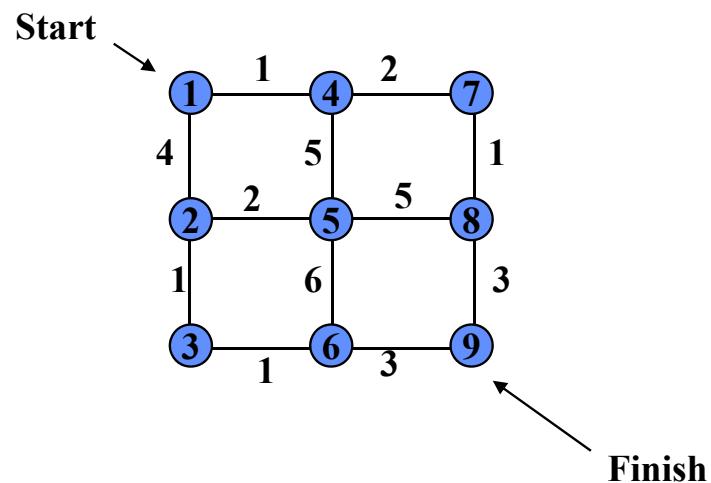
# Graph Theoretic Approaches

- Gradient detection methods by themselves are seldom suitable for preprocessing when there is any noise present
- A better approach is a global approach based on treating the pixel as a graph and finding a low cost paths that correspond to significant edges
- These methods tend to work well but are usually computationally expensive



## Finding a Low Cost Path

Consider a set of nodes,  $n_i$ , with differing costs associated with paths between the nodes,  $c_i$ . A graph in which the arcs are directed is called a *directed graph*. A sequence of nodes  $n_1, n_2, \dots, n_k$  with each node  $n_i$  being a successor of node  $n_{i-1}$  is called a *path*.



Cost of the path is

$$c = \sum_{i=2}^k c(n_{i-1}, n_i)$$



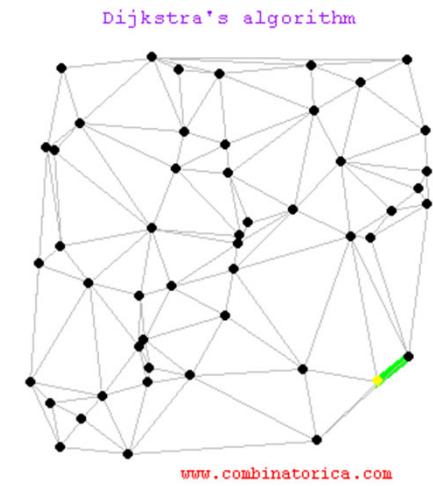
## Principle of Optimality

- “If a node  $n$  is on the optimal (lowest cost) path between nodes and A and B, then the paths  $An$  and  $nB$  must also be optimal”
- For example, this means that if the shortest route between say Brisbane and Sydney includes Newcastle, then that section of the route between Newcastle and Sydney must also be the shortest route between those two cities.
- This principle may be considered self evident, but it allows great simplification in computation of optimal paths because suboptimal routes can be very rapidly discarded.



# Dijkstra's Algorithm

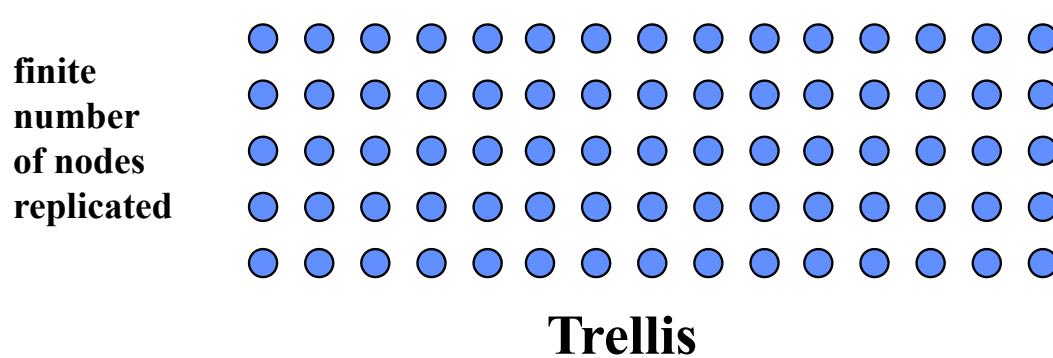
- **Shortest Path from single source problem on a graph (or network)**
  - Given a connected graph  $G=(V,E)$ , a weight  $d:E \rightarrow R^+$  and a fixed vertex  $s$  in  $V$ , find a shortest path from  $s$  to each vertex  $v$  in  $V$ .
- **Dijkstra's Algorithm**
  - Dijkstra's algorithm is known to be a good algorithm to find a shortest path.
  - Set  $i=0$ ,  $S_0 = \{u_0=s\}$ ,  $L(u_0)=0$ , and  $L(v)=\infty$  for  $v \neq u_0$ . If  $|V|=1$  then stop, otherwise go to step 2.
  - For each  $v$  in  $V \setminus S_i$ , replace  $L(v)$  by  $\min\{L(v), L(u_i) + d_{u_i v}\}$ . If  $L(v)$  is replaced, put a label  $(L(v), u_i)$  on  $v$ .
  - Find a vertex  $v$  which minimizes  $\{L(v) : v \in V \setminus S_i\}$ , say  $u_{i+1}$ .
  - Let  $S_{i+1} = S_i \cup \{u_{i+1}\}$ .
  - Replace  $i$  by  $i+1$ . If  $i=|V|-1$  then stop, otherwise go to step 2.
  - The time required by Dijkstra's algorithm is  $O(|V|^2)$ . It will be reduced to  $O(|E|\log|V|)$  if heap is used to keep  $\{v \in V \setminus S_i : L(v) < \infty\}$ .





## Viterbi Search on a Trellis

- The Viterbi algorithm is very famous and this algorithm is used in many applications including Hidden Markov Modeling and the V32/V90 telephone modem
- This algorithm is a very fast and optimal method for searching a trellis for a minimum cost path
- Similar to Dijkstra's shortest paths from single source algorithm except that self-transitions are allowed and have a cost.





# Viterbi Algorithm as Used in HMMs

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \quad (32a)$$

$$\psi_1(i) = 0. \quad (32b)$$

2) Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T \\ 1 \leq j \leq N \quad (33a)$$

$$\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T \\ 1 \leq j \leq N. \quad (33b)$$

3) Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \quad (34a)$$

$$q_T^* = \operatorname{argmax}_{1 \leq i \leq N} [\delta_T(i)]. \quad (34b)$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T - 1, T - 2, \dots, 1. \quad (35)$$

Set initial costs

Move through trellis  
keeping only  
optimal paths

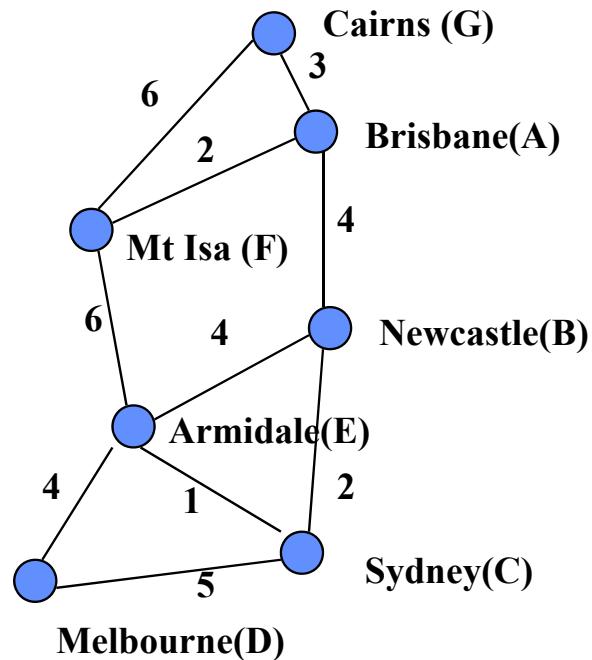
Find destination node

Backtrack to find path



## Viterbi-Like Shortest Path Search

**Find Lowest Cost Path  
from Brisbane to Sydney**





## Algorithm

1. Find all cities reachable with one hop from Brisbane. Prune to keep only best path for each destination city.
2. Find all cities reachable with two hops from Brisbane. Prune to keep best path for each destination city. etc.

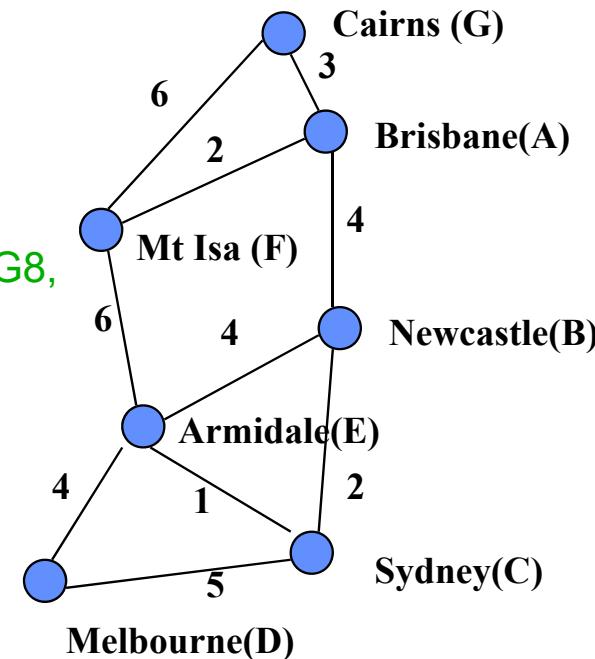
**Continue until destination city is found, and  
all paths exceed current destination city  
path cost. (need to ensure that no low cost,  
high hopcount path exists)**



## Worked Example

A	B	C	D	E	F	G
0	4	6	-	8	2	3

- Looking for node C (Sydney)
- One Hop: AB4, AF2, AG3
- All Paths Optimal: AB4, AF2, AG3
- Two Hops: ABE8, ABC6, AFG8, AFE8, AGF9
- Prune to Optimal paths: ABC6, ABE8, AFG8, AGF9
- No shorter paths possible
- Solution: ABC6





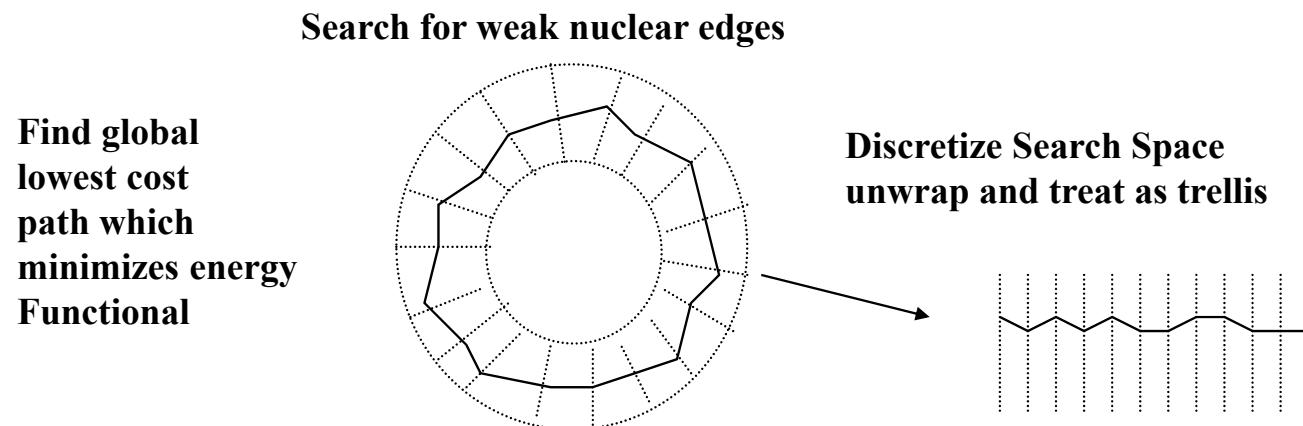
## Notes

- The number of optimal paths at each stage will never exceed the number of nodes since we can immediately prune sub-optimal paths.
- If we had a trellis of 30 nodes with a path length of 30, we would require  $30^{30} \sim 2 \times 10^{44}$  path evaluations to find the optimal path by exhaustive searching
- With the Viterbi algorithm it only takes of the order of  $30 \times 30 \sim 1000$  evaluations (also depends on average node fan-out). That is, the algorithm is linear in computational complexity.



## Application

- This method has been used to speed up a patented cell image segmentation algorithm as described in
- *Pascal Bamford and Brian Lovell, “Unsupervised Cell Nucleus Segmentation with Active Contours,” Signal Processing – Special Issue on Deformable Models and Techniques for Signal and Image Processing, V 71, pp 203-213*





# **Case Study: A Methodology for Quality Control in Cell Nucleus Segmentation**

Slides courtesy Pascal Bamford



## Cytometric Image Analysis System

***Slide In***



***Slide Handling  
Slide Scanning  
Image Acquisition***

***Image  
Segmentation***

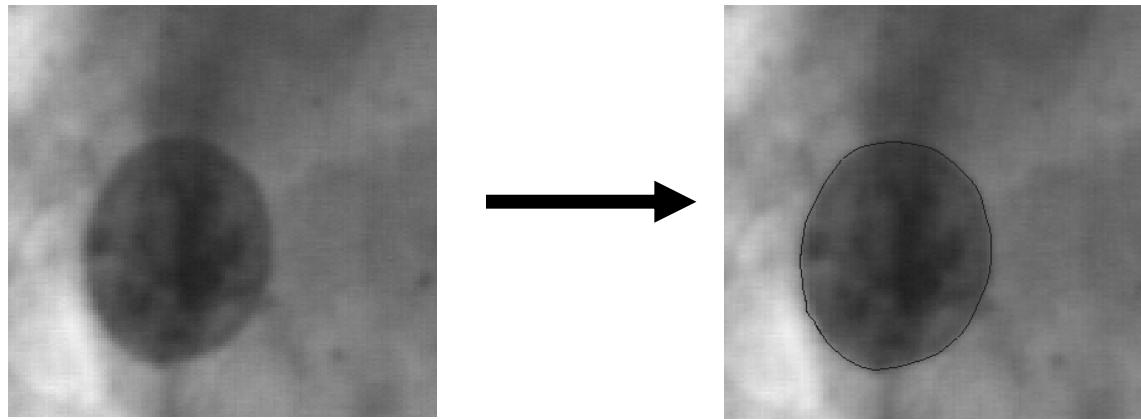
***Feature  
Extraction***

***Cell/Slide  
Classification***

***Diagnostic Score Out***



## Cell Nucleus Segmentation



The images:

- are of the order 128x128 pixels, 256 graylevels
- contain one desired object per image (maybe doublets)
- often contain many artefacts
- are highly variable due to staining and illumination



## Cost (Energy) of a Contour

$$\sum_i C(i, \lambda) = \sum_i \lambda \text{curv}(v_{i-1}, v_i, v_{i+1}) + (1 - \lambda)(1 - \vec{\nabla}I(v_i))$$

Where:

**Only one parameter**

$$\text{curv}(v_{i+1,j}, v_{i,j}, v_{i-1,j}) = \left( \frac{v_{i+1} - 2v_i + v_{i-1}}{v_{i+1} - v_{i-1}} \right)^2$$

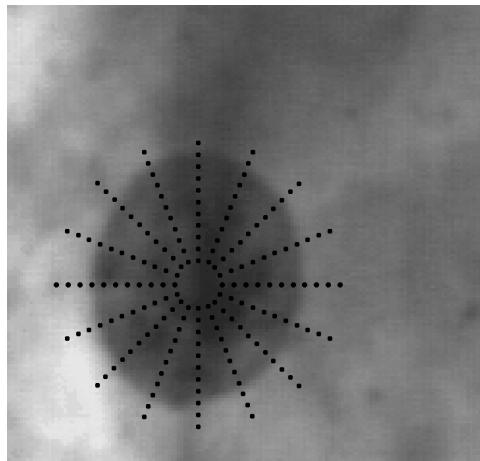
**is the curvature of the contour calculated locally over three adjacent points**

and

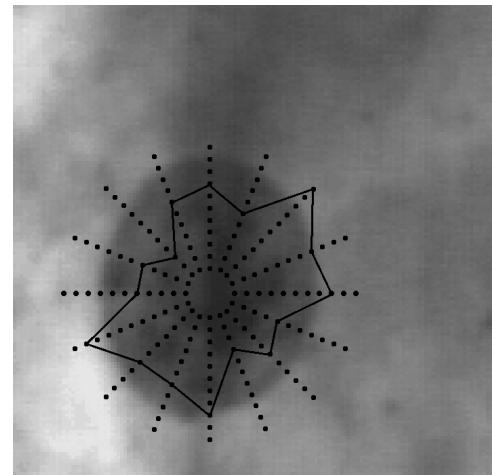
$\vec{\nabla}I(v_i)$  **is the directional gradient along a radius towards the centre of the object**



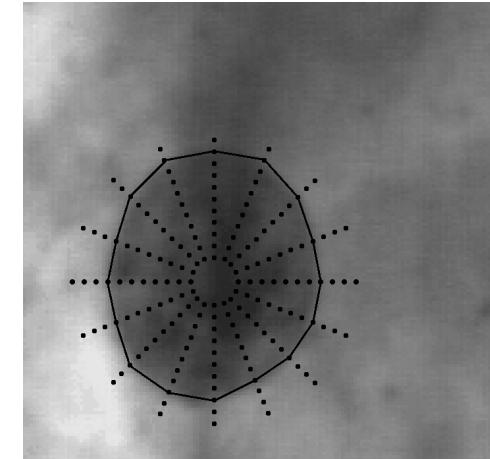
# Method



**Superimpose discrete search space on image**



**Evaluate every possible contour lying on the search space by choosing one point on each radius**

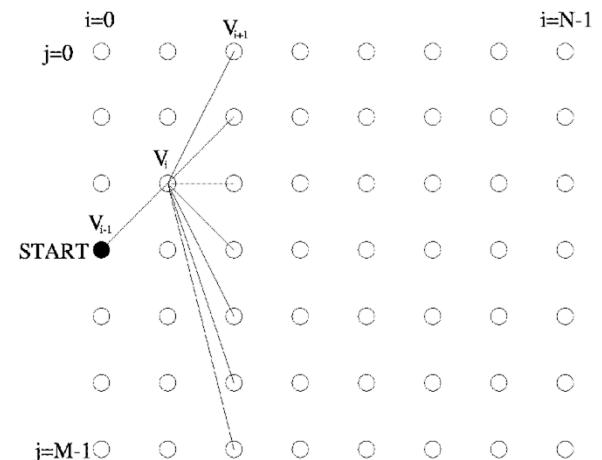
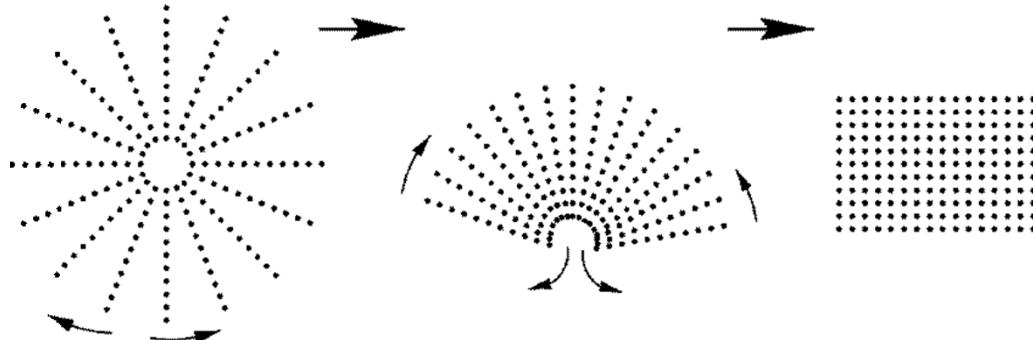


**Choose contour that minimises cost function below**



### Viterbi Trellis Search Space

$$S_i(v_{i+1}, v_i) = S_{i-1}(v_i, v_{i-1}) + \min[C(i, \lambda)]$$



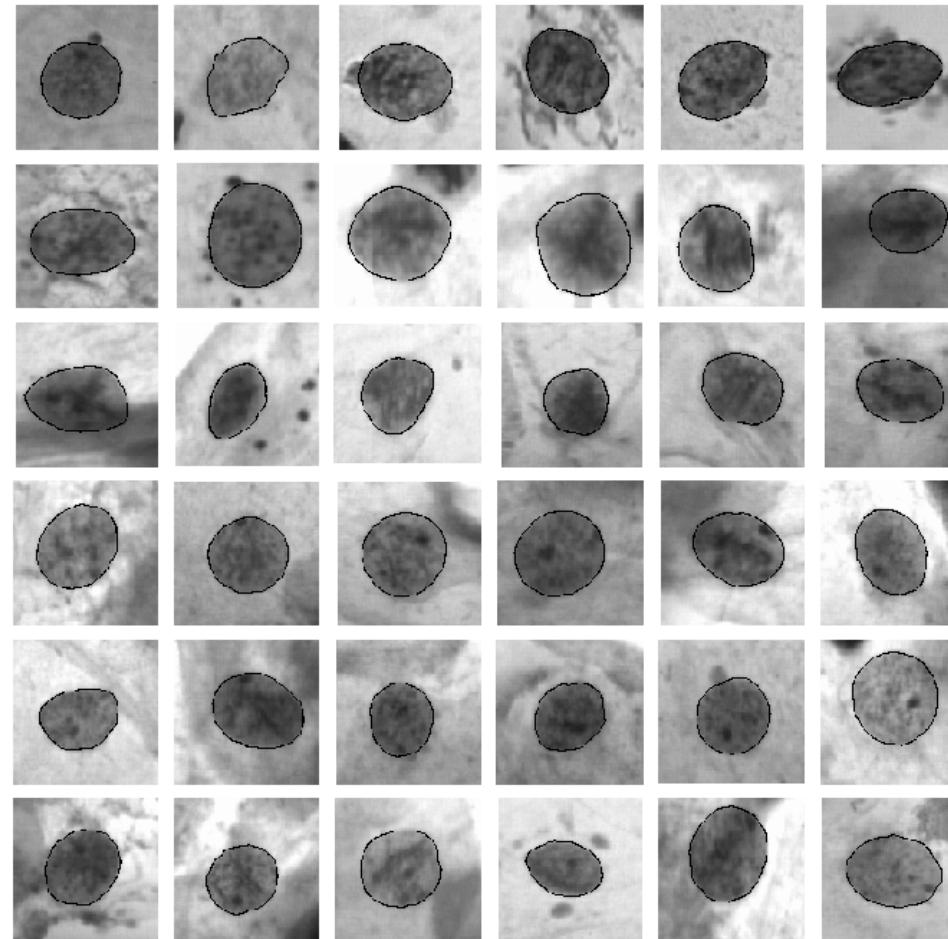


# Segmentation Performance

- The algorithm was run upon a dataset of 20,130 images with  $\lambda=0.8$
- Each image was examined (by a human) and the segmentation classified as 'pass' or 'fail'
- 20,057/20,130 (99.64%) of images were correctly segmented
- Of the 73 failures:
  - 44 could be rectified by a better marker,
  - 26 could be rectified by using another value of  $\lambda$ ,
  - 3 were unsegmentable and failed at all attempts



## Typical Segmentation Results



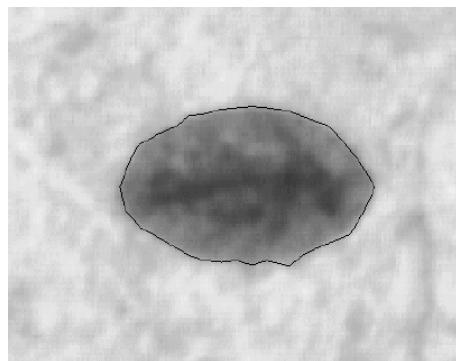


## Interpretation of Parameter Lambda

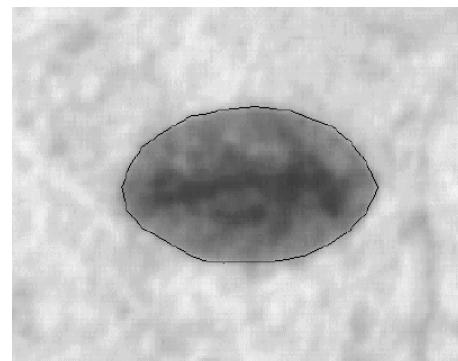
- This parameter determines the weighting between smoothness and requirement for contour to lie on high gradient edges
- $\lambda=0$ , segmentation based on gradient information alone; sensitive to “noise”
- $\lambda=1$ , segmentation based on smoothness constraint alone; produces circle and ignores image



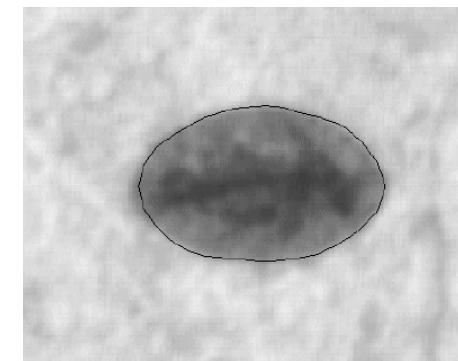
## Lambda Sensitivity



$\lambda=0.1$

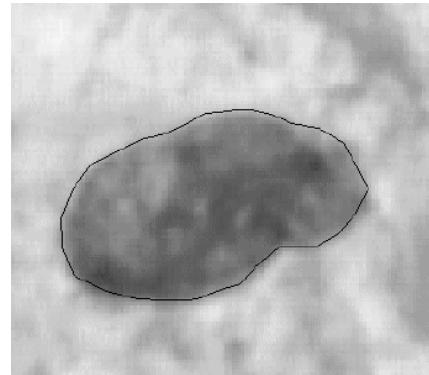
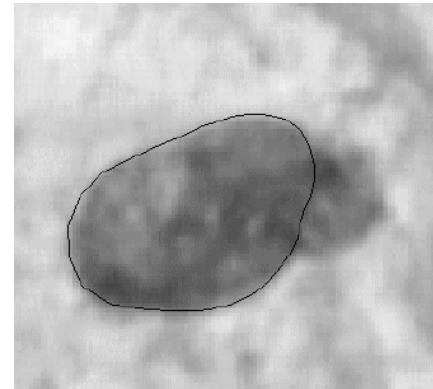
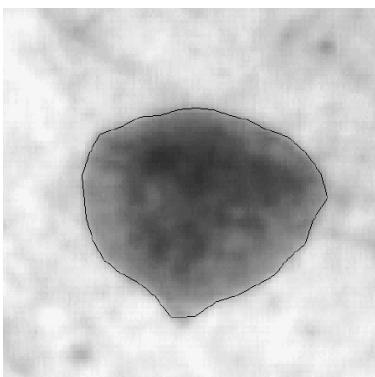
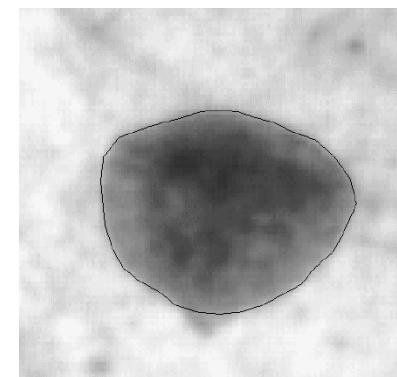


$\lambda=0.5$



$\lambda=0.9$

**Observation: ‘Easy’ to segment images are  
stable over a wide range of  $\lambda$**

 $\lambda=0.5$  $\lambda=0.7$  $\lambda=0.5$  $\lambda=0.8$ 

**Observation: 'Difficult' to segment images are  
not stable over a wide range of  $\lambda$**

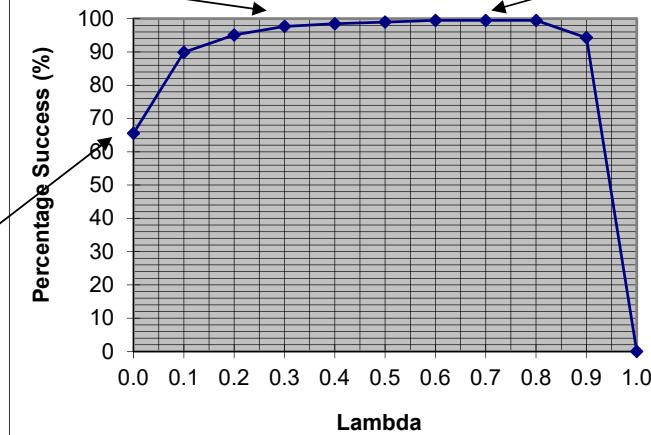


# Probability of Correct Segmentation

Harder images segmented correctly

Easy images segmented correctly

Overall highest probability of correct segmentation 99.6%



In order to gain an *a priori* estimate of the probability of a correct segmentation, the above graph was obtained by segmenting a subset of 772 images with values of  $\lambda$  in the range  $\lambda = 0.1k$  where  $k = 0,1,2,\dots,10$ .



## Classification into “Easy” and “Hard” Images

1. Segment all images with  $\lambda = 0.7$  say
2. Resegment at  $\lambda = 0.0, 0.1, 0.2, \dots$
3. Observation: Cell Images correctly segmented for low  $\lambda (<0.5)$  are strict subsets of the images correctly segmented at  $\lambda = 0.7$
4. If segmentation at  $\lambda = 0.7$  and  $\lambda = 0.0$  are the same, image was “easy” and probability of segmentation error at  $\lambda = 0.7$  is extremely low. Just about any method would work for these.
5. Similarly, if segmentation at  $\lambda = 0.7$  and  $\lambda = 0.1$  are the same, but different from  $\lambda = 0.0$  segmentation, image is “harder” with higher probability of segmentation error at  $\lambda = 0.7$ .
6. Achieve 100% correct segmentation by rejecting the images classed as “hard.” (about 10% of the database)



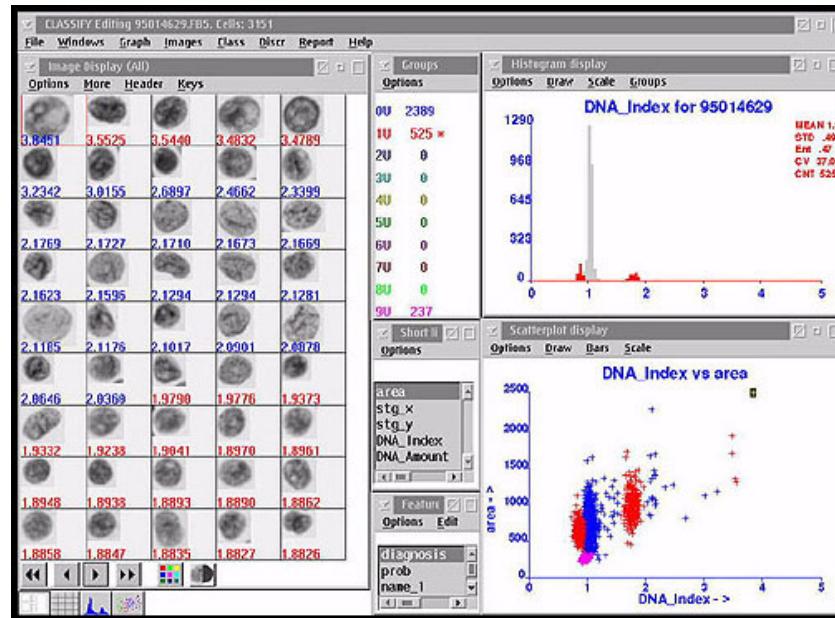
## AcCell-SavantTM



The AcCell-SavantTM is a fully-automated, high-resolution, absorbance microscopy-based image cytometer that processes Thionin-Feulgen-stained cytology preparations and presents analytical results regarding the cellular DNA content of processed samples.



## Cell Analysis Interface



Typical digital images of DNA-stained nuclei and the associated ploidy distribution from an aneuploid population of cells from a mesothelioma. The AcCell-Savant™ also displays in this sample the DNA index as a function of the area of the sampled nuclei.



# Outcomes

- We now have a reliable method of grading images into “easy” and “hard” classes without human intervention
- Can achieve 100% correct segmentation by automatically rejecting a small percentage of the images collected.



# Malignancy Associated Changes (MACS)

- MACS - specific sub-visual changes in cell morphometry, optical density and chromatin distribution in a proportion of visually normal cells in smears from or near malignant and pre-malignant lesions
- MAC analysis is a new way of looking at cells on a smear
- Made possible due to CCD camera development, advances in computing power and modern image analysis algorithms



## Special Features of MACS

- MACs can be detected in cells distant to dysplastic and malignant lesions, thereby reducing false negatives which result from inadequate sampling of lesions (no requirement for “diagnostic” cells in smear)
- MAC analysis can detect very subtle sub-visual cellular changes and therefore can detect cancer at an early stage allowing early intervention

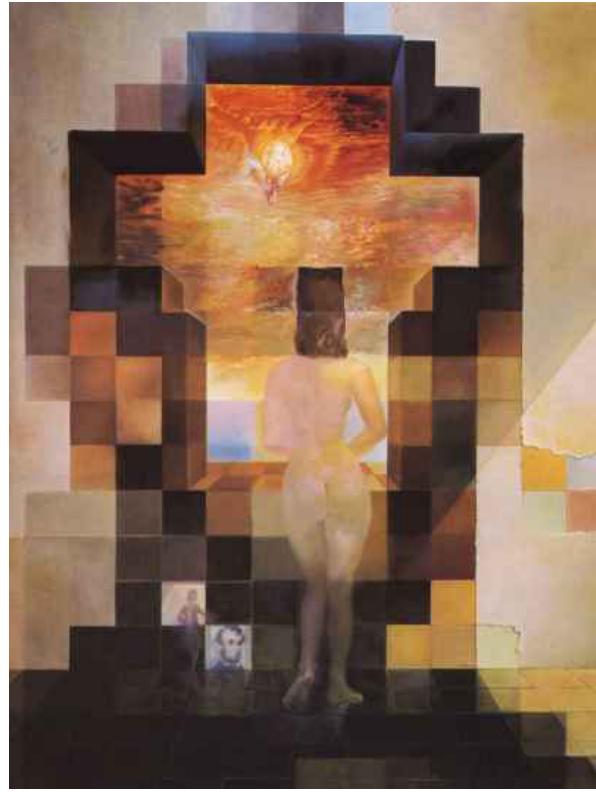


# MACS

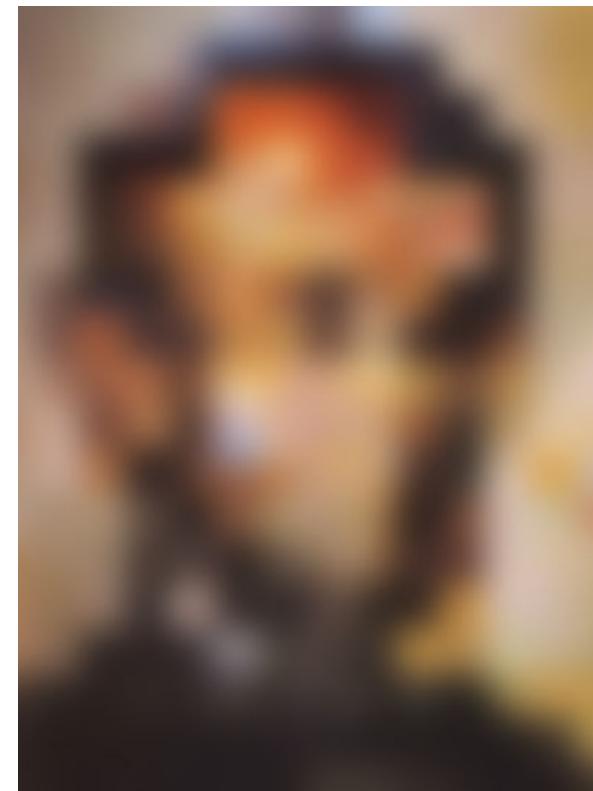
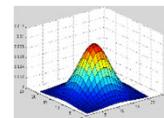
- Objective MAC scoring eliminates the subjectivity of visual smear assessment
- The MAC score gives an indication of the likely behaviour of lesions and therefore has prognostic value
- Needs less expensive hardware
- May enable screening for other cancers e.g., lung, oral, bladder.



# Global to Local Analysis



Dali





# Linear image transformations

- In analyzing images, it's often useful to make a change of basis.

$$\vec{F} = \vec{U}\vec{f}$$

transformed image      Vectorized image  
Fourier transform, or  
Wavelet transform, or  
Steerable pyramid transform

```
graph LR; A[transformed image] --> C["Fourier transform, or  
Wavelet transform, or  
Steerable pyramid transform"]; B[Vectorized image] --> C; C --> D["\vec{F} = \vec{U}\vec{f}"]
```



# Self-inverting transforms

Same basis functions are used for the inverse transform

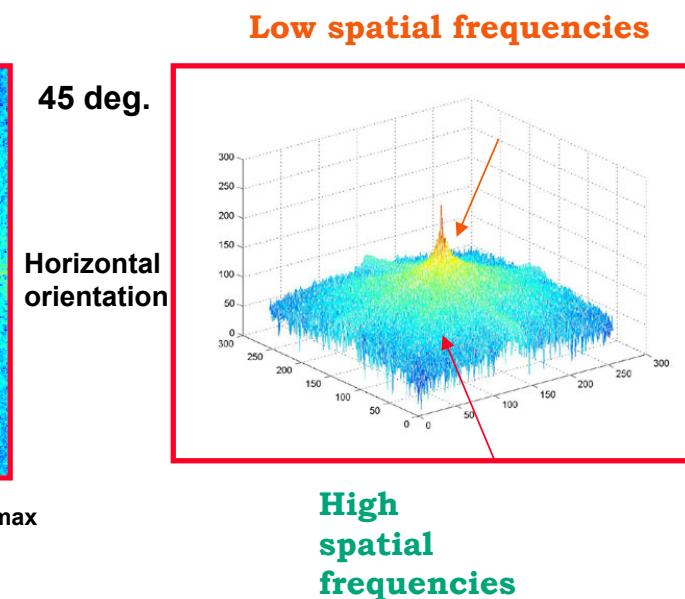
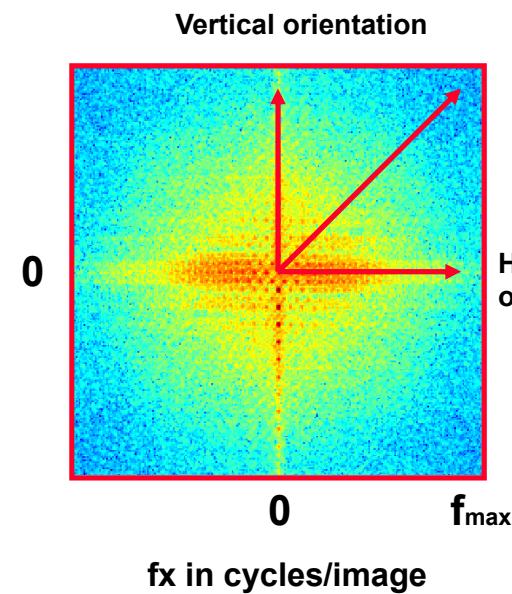
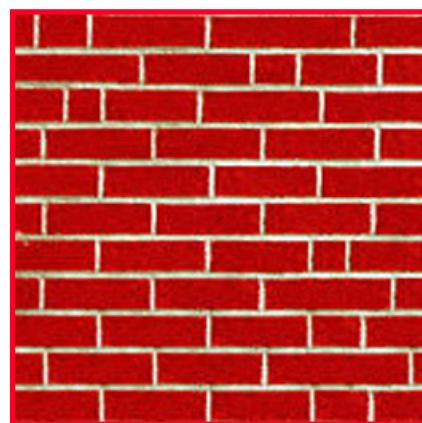
$$\vec{f} = U^{-1} \vec{F}$$

$$= U^+ \vec{F}$$



U transpose and complex conjugate

# How to interpret a Fourier Spectrum

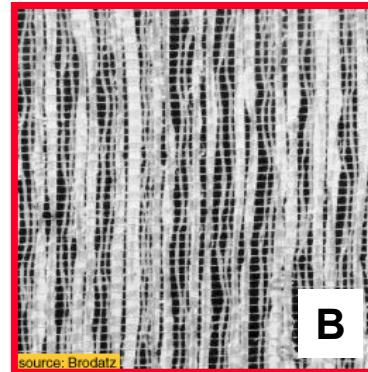




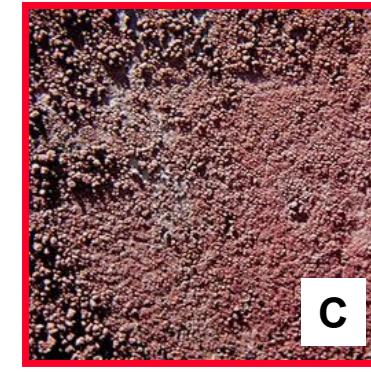
# Fourier Amplitude Spectrum



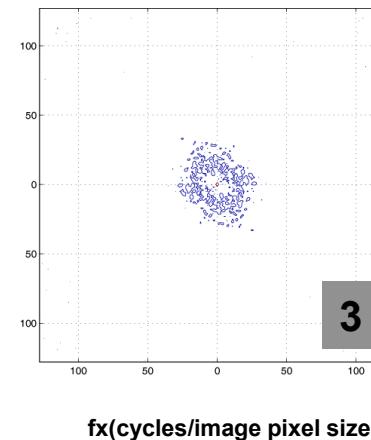
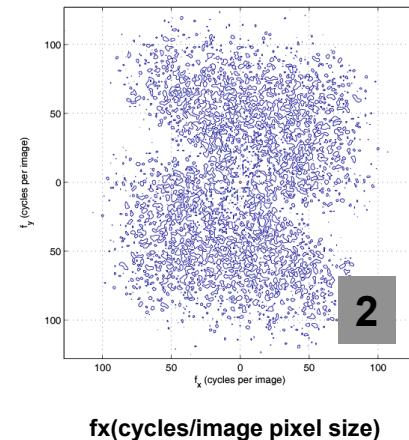
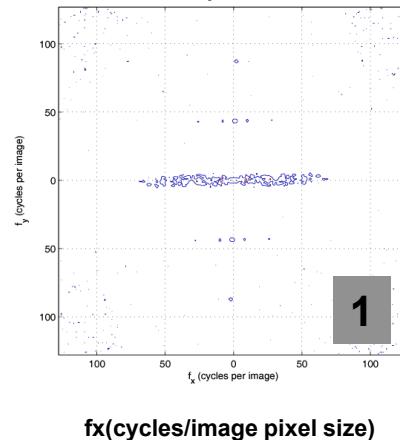
A



B



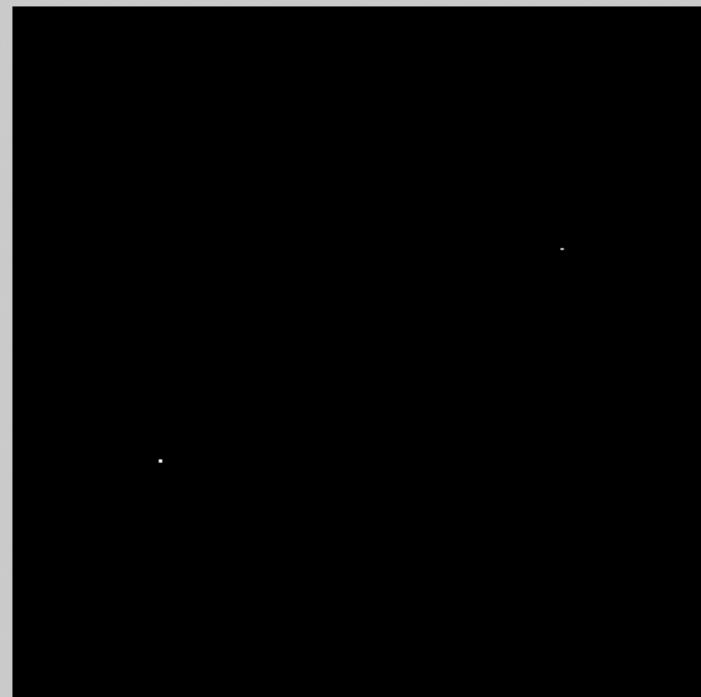
C



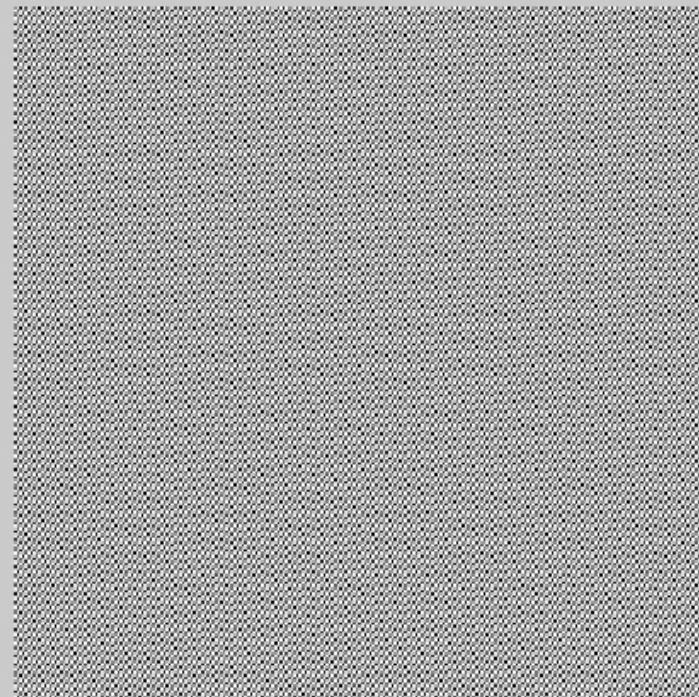


2

2



#1: Range [0, 1]  
Dims [256, 256]



#2: Range [0.000109, 0.0267]  
Dims [256, 256]

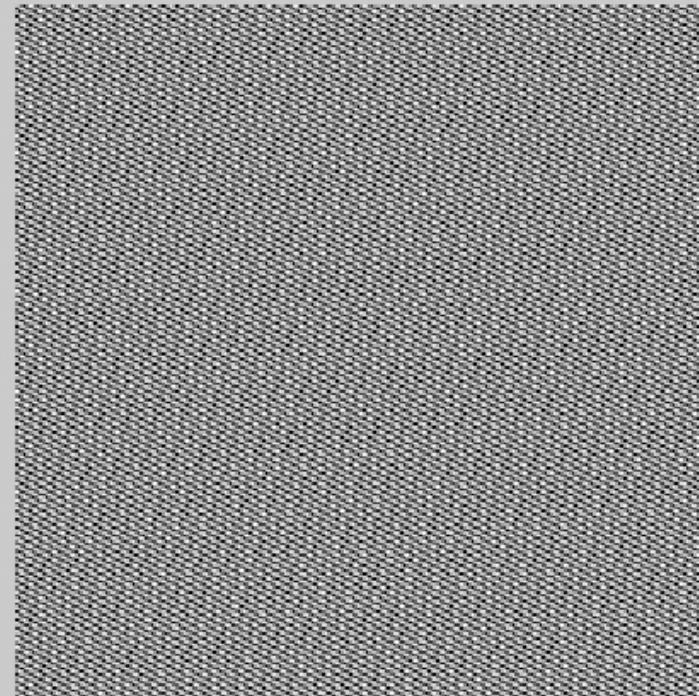


# 6

6



#1: Range [0, 1]  
Dims [256, 256]



#2: Range [1.89e-007, 0.226]  
Dims [256, 256]

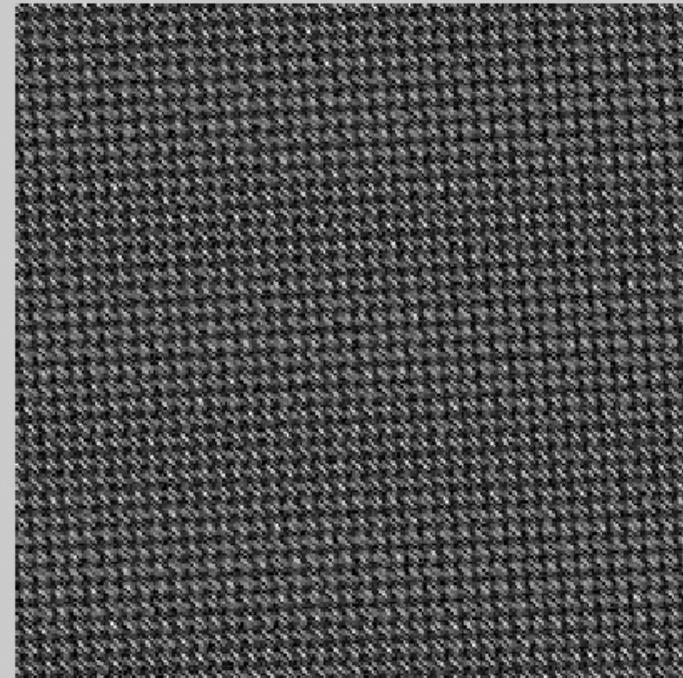


18

18



#1: Range [0, 1]  
Dims [256, 256]

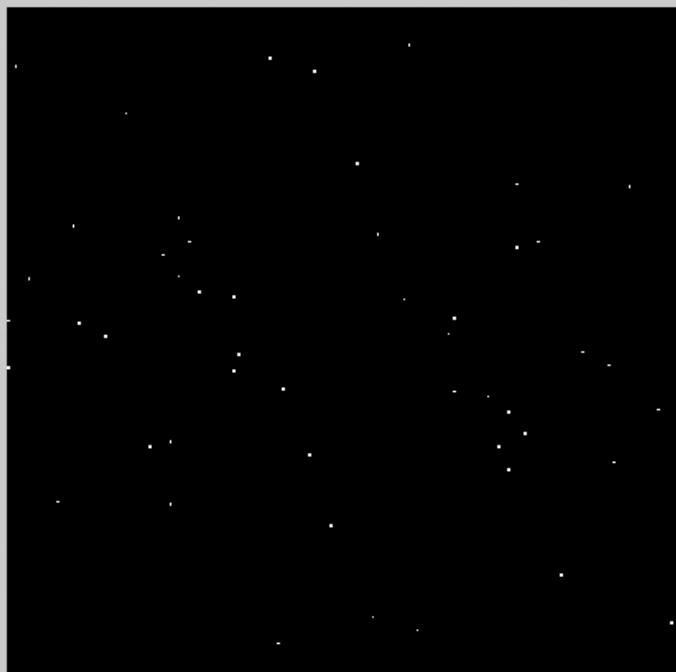


#2: Range [4.79e-007, 0.503]  
Dims [256, 256]

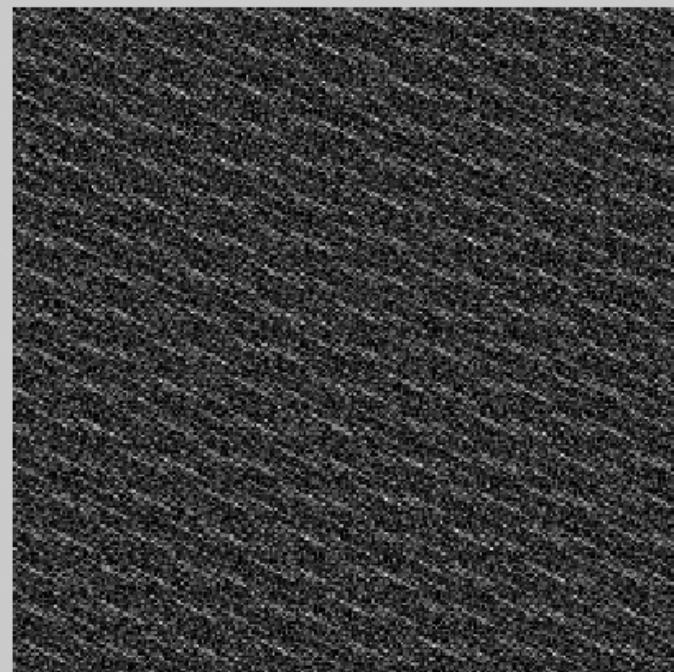


50

50



#1: Range [0, 1]  
Dims [256, 256]



#2: Range [8.5e-006, 1.7]  
Dims [256, 256]

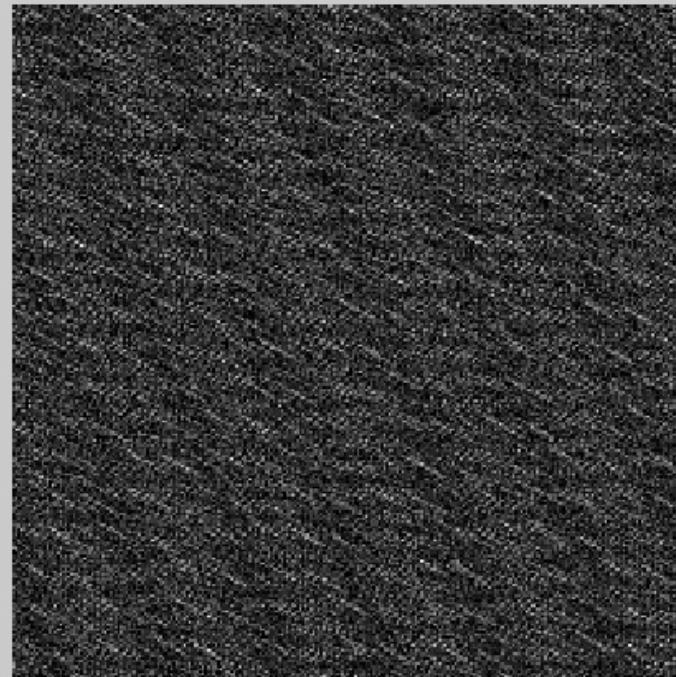


82

82



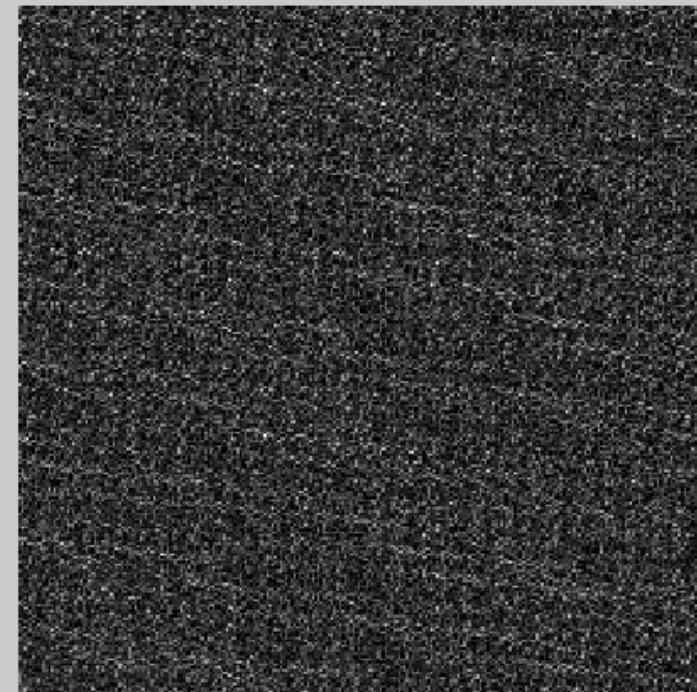
#1: Range [0, 1]  
Dims [256, 256]



#2: Range [3.85e-007, 2.21]  
Dims [256, 256]



136

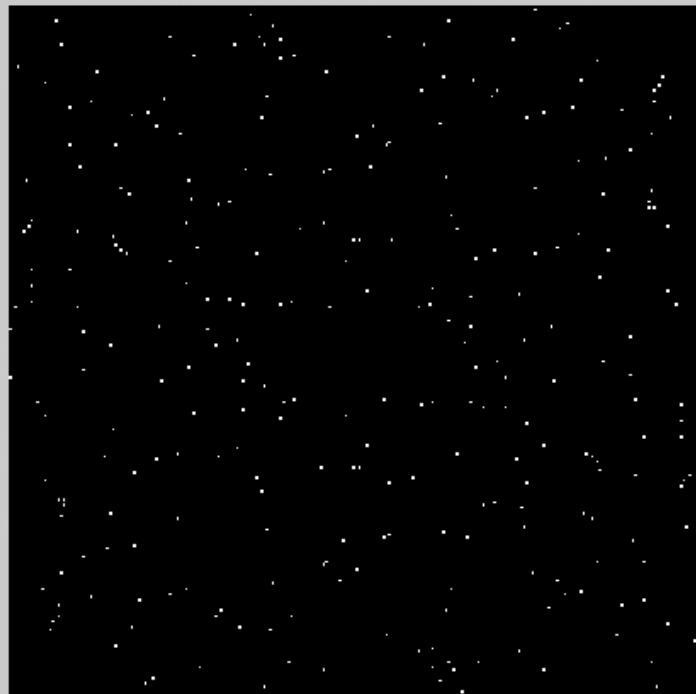


#2: Range [8.25e-006, 3.48]  
Dims [256, 256]

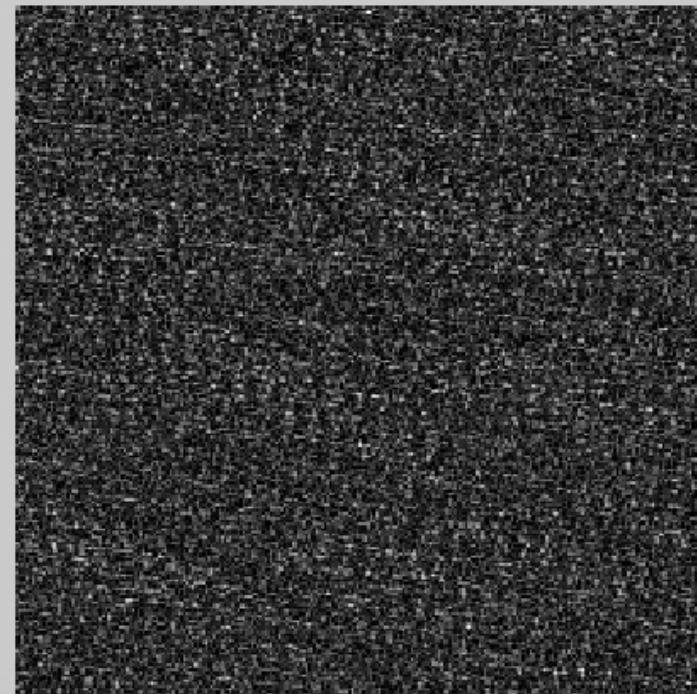


**282**

282



#1: Range [0, 1]  
Dims [256, 256]

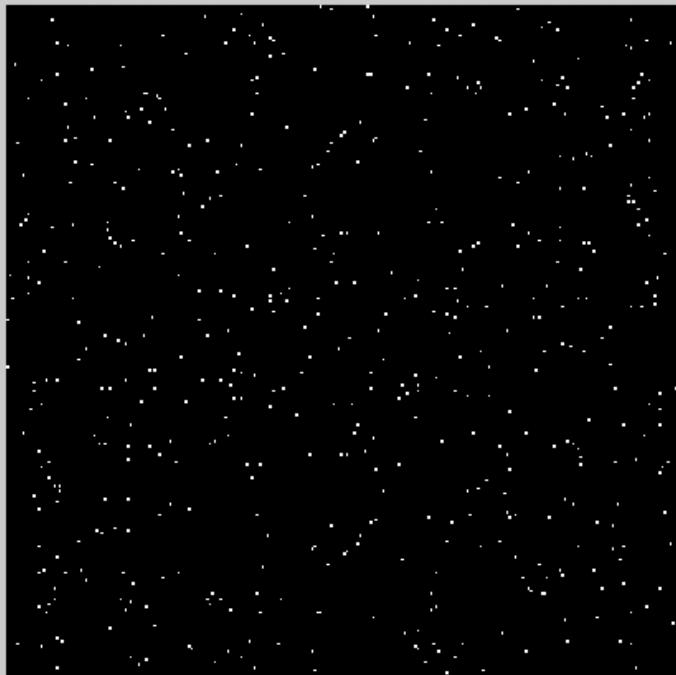


#2: Range [1.39e-005, 5.88]  
Dims [256, 256]

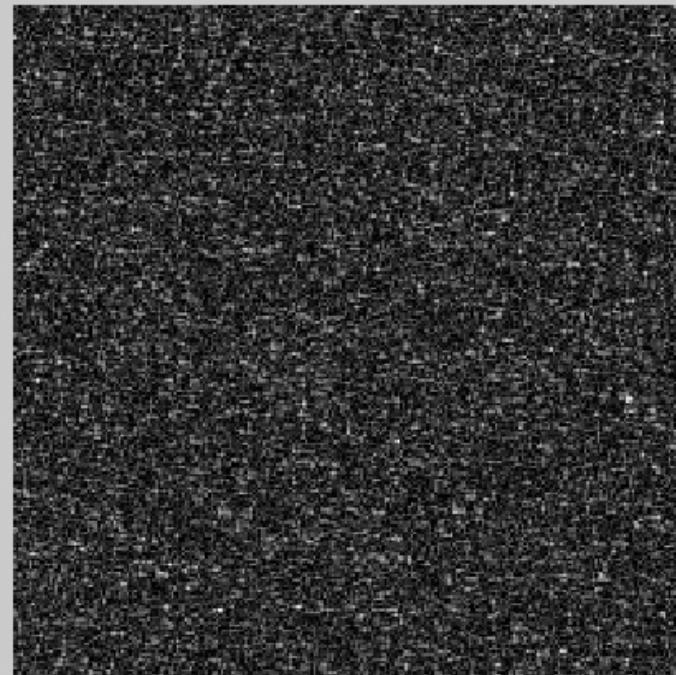


**538**

538



#1: Range [0, 1]  
Dims [256, 256]

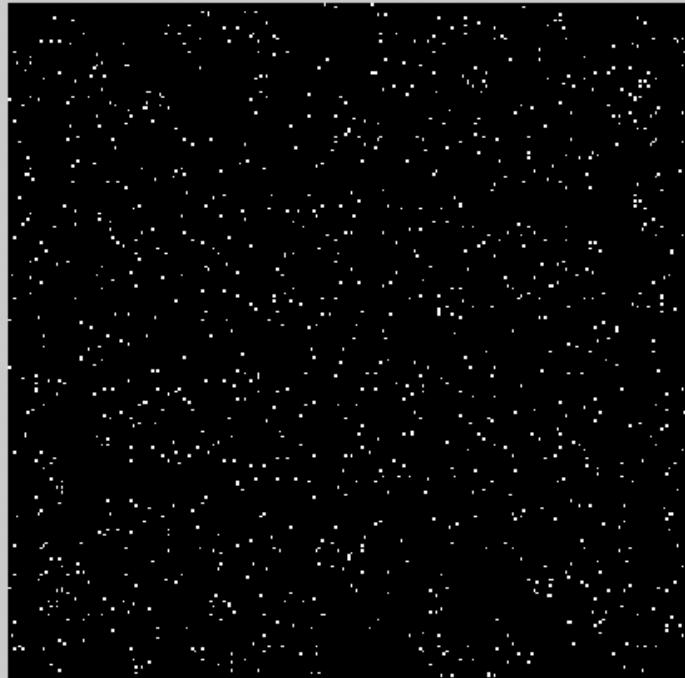


#2: Range [6.17e-006, 8.4]  
Dims [256, 256]

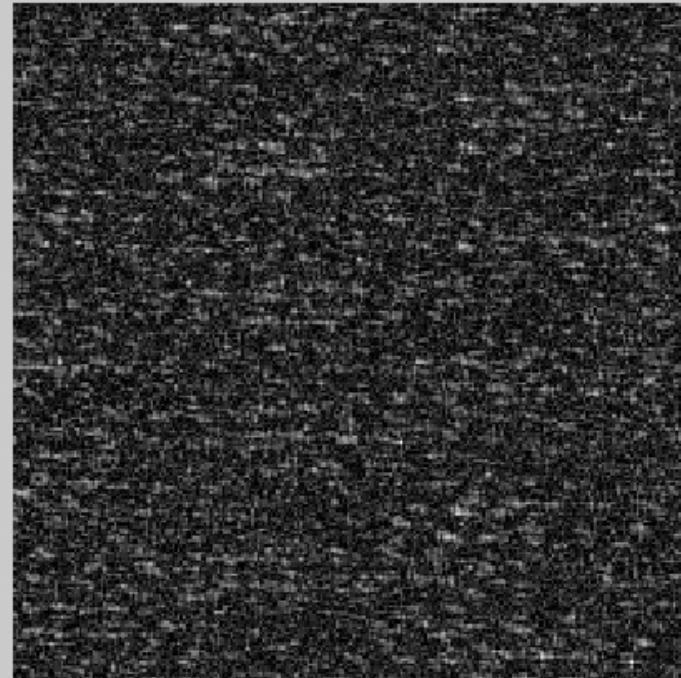


# 1088

1088



#1: Range [0, 1]  
Dims [256, 256]

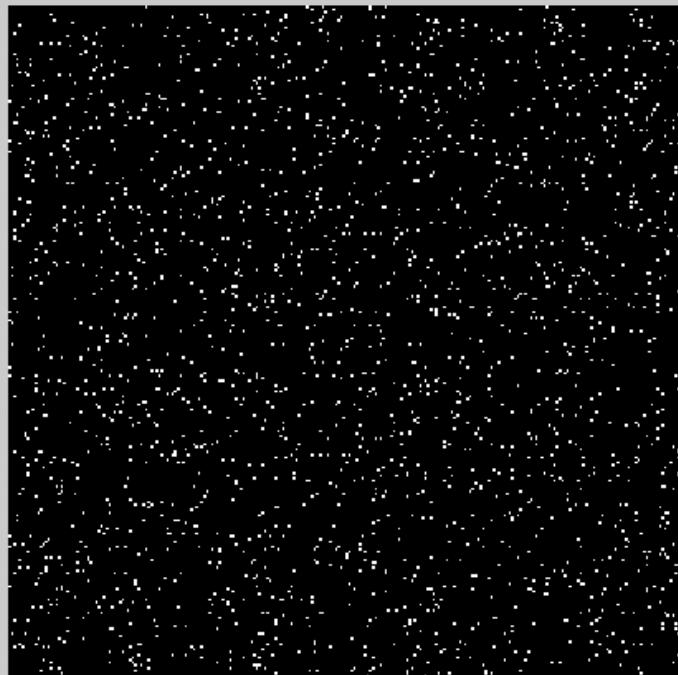


#2: Range [9.99e-005, 15]  
Dims [256, 256]

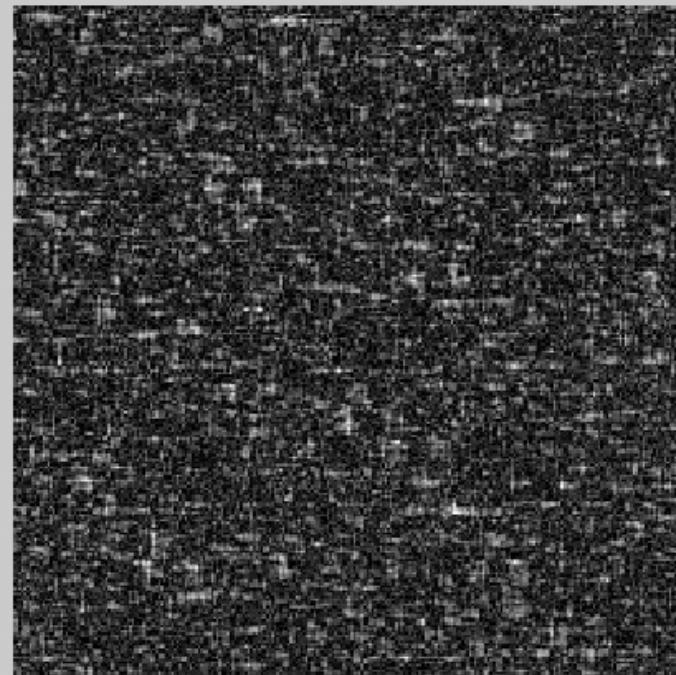


# 2094

2094



#1: Range [0, 1]  
Dims [256, 256]

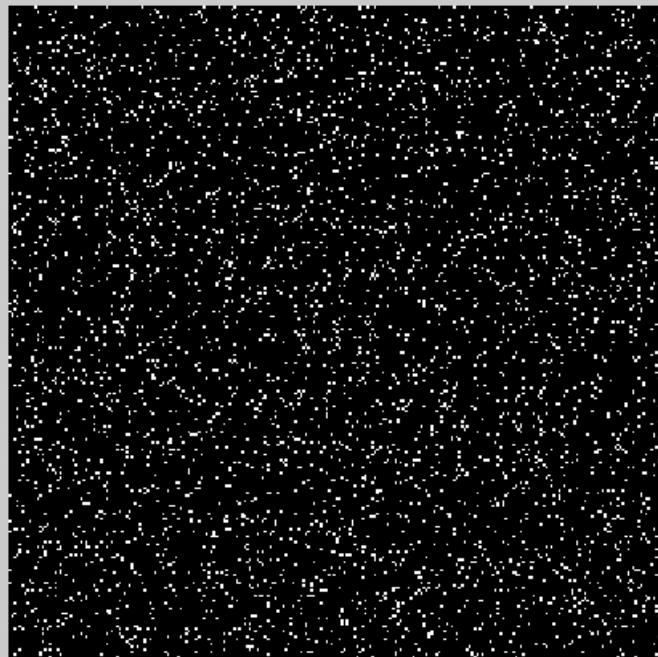


#2: Range [8.7e-005, 19]  
Dims [256, 256]

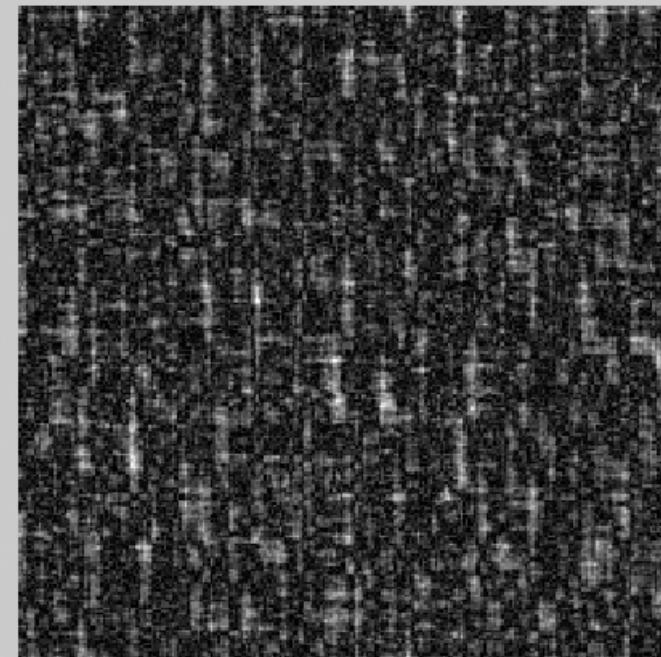


# 4052.

4052



#1: Range [0, 1]  
Dims [256, 256]

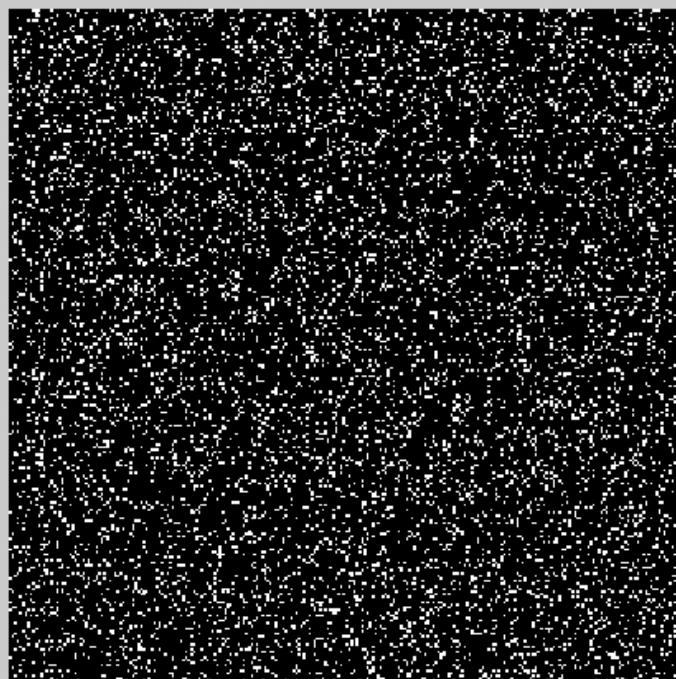


#2: Range [0.000556, 37.7]  
Dims [256, 256]

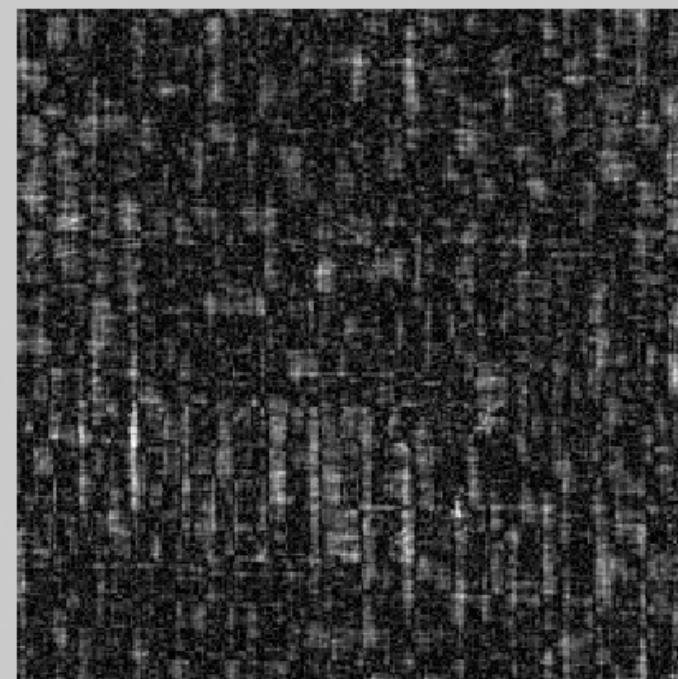


# 8056.

8056



#1: Range [0, 1]  
Dims [256, 256]

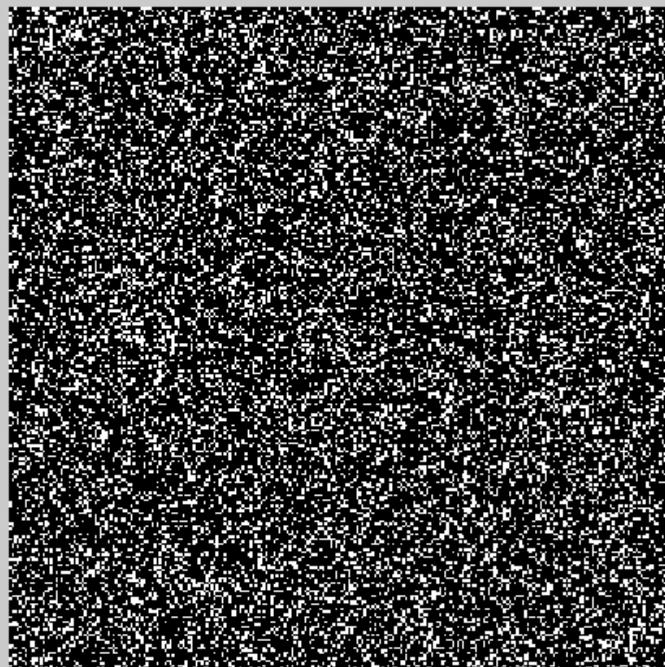


#2: Range [0.00032, 64.5]  
Dims [256, 256]

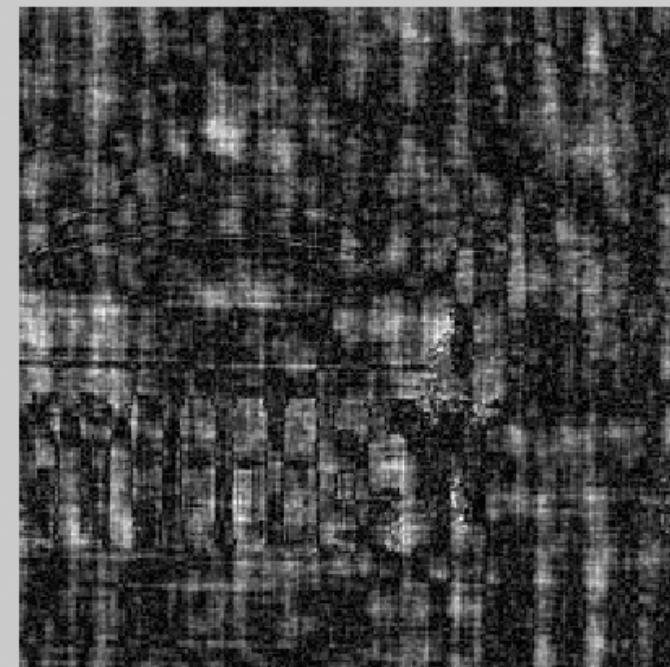


# 15366

15366



#1: Range [0, 1]  
Dims [256, 256]

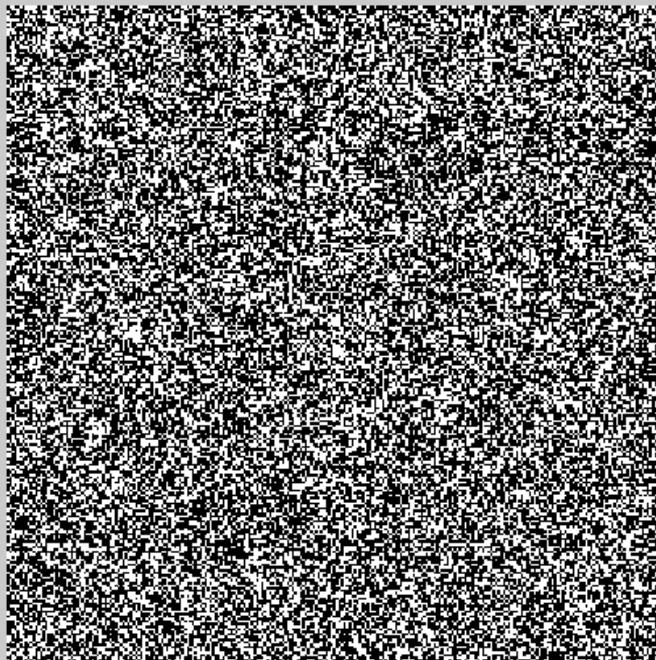


#2: Range [0.000231, 91.1]  
Dims [256, 256]

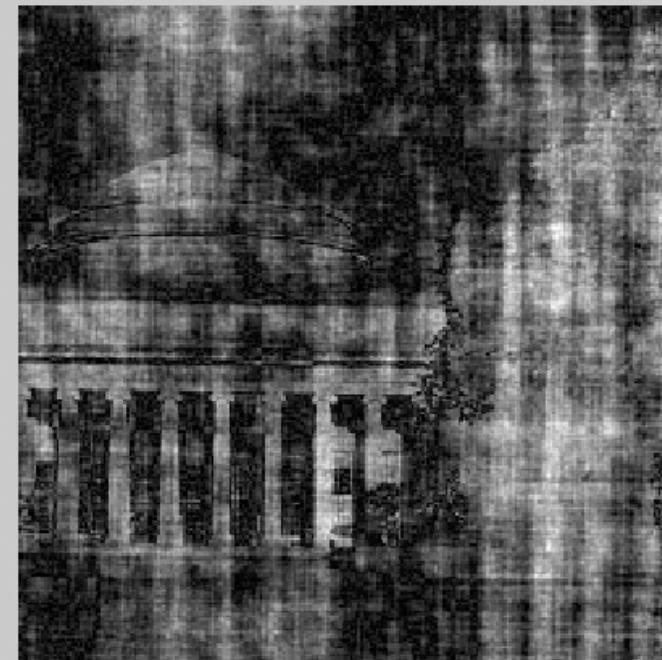


# 28743

28743



#1: Range [0, 1]  
Dims [256, 256]

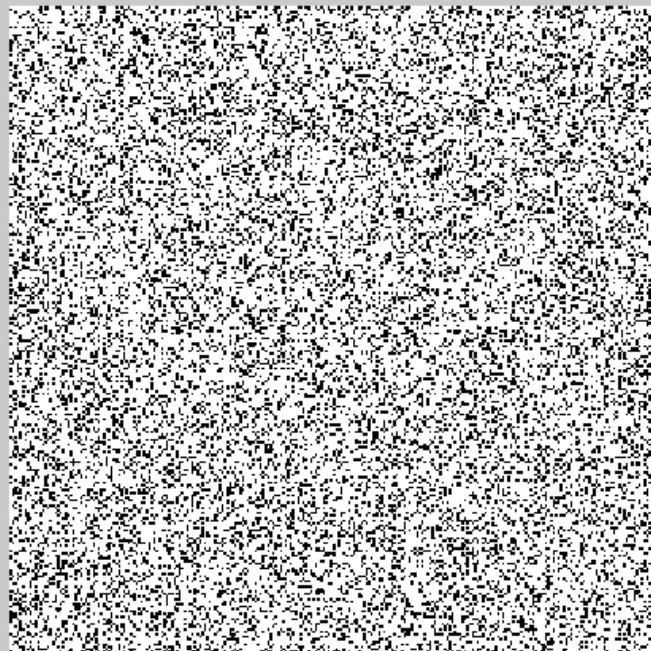


#2: Range [0.00109, 146]  
Dims [256, 256]

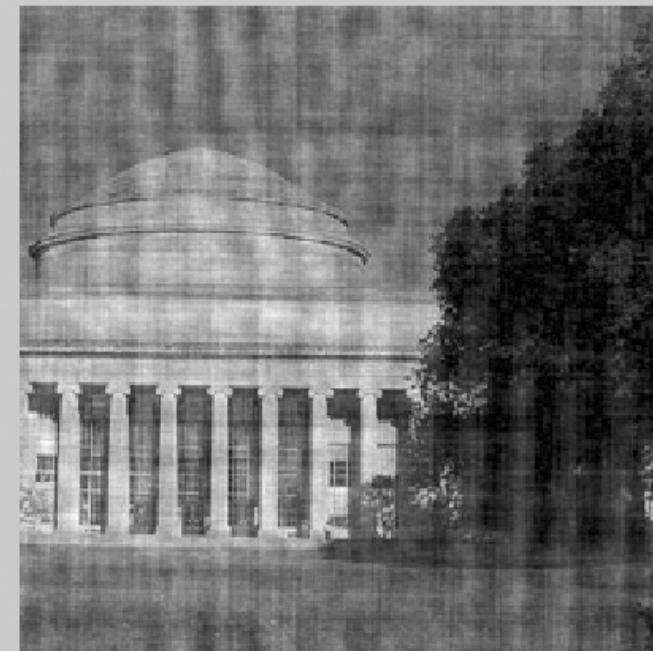


49190.

49190



#1: Range [0, 1]  
Dims [256, 256]



#2: Range [0.00758, 294]  
Dims [256, 256]



# 65536.

65536.



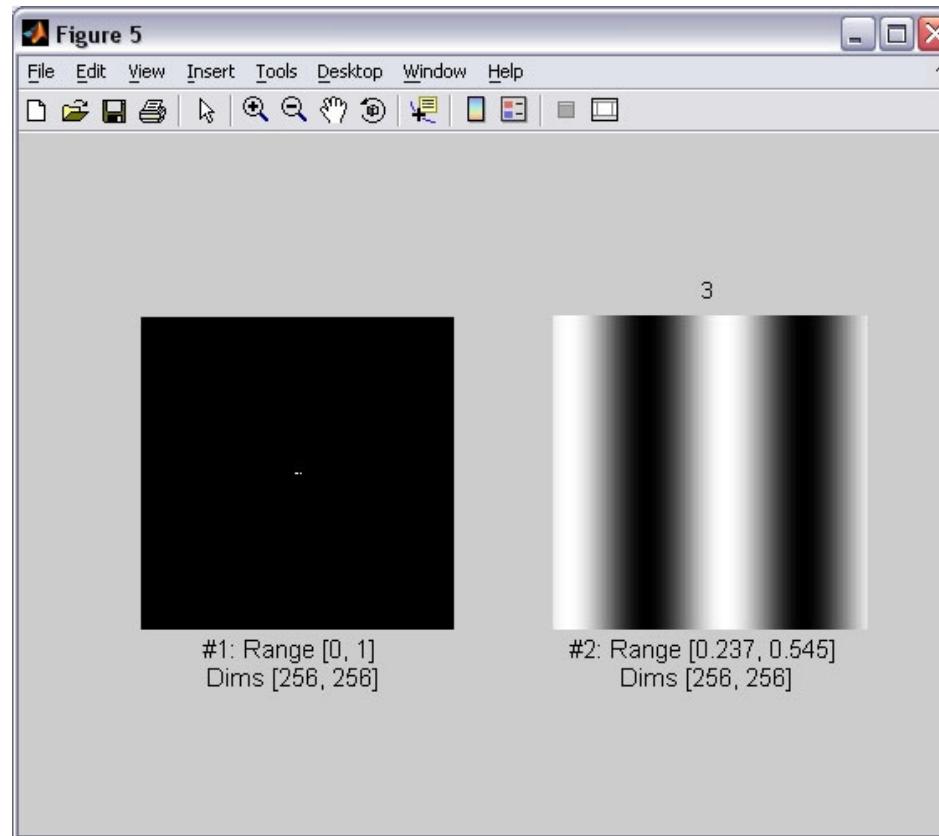
#1: Range [0.5, 1.5]  
Dims [256, 256]



#2: Range [4.43e-015, 255]  
Dims [256, 256]



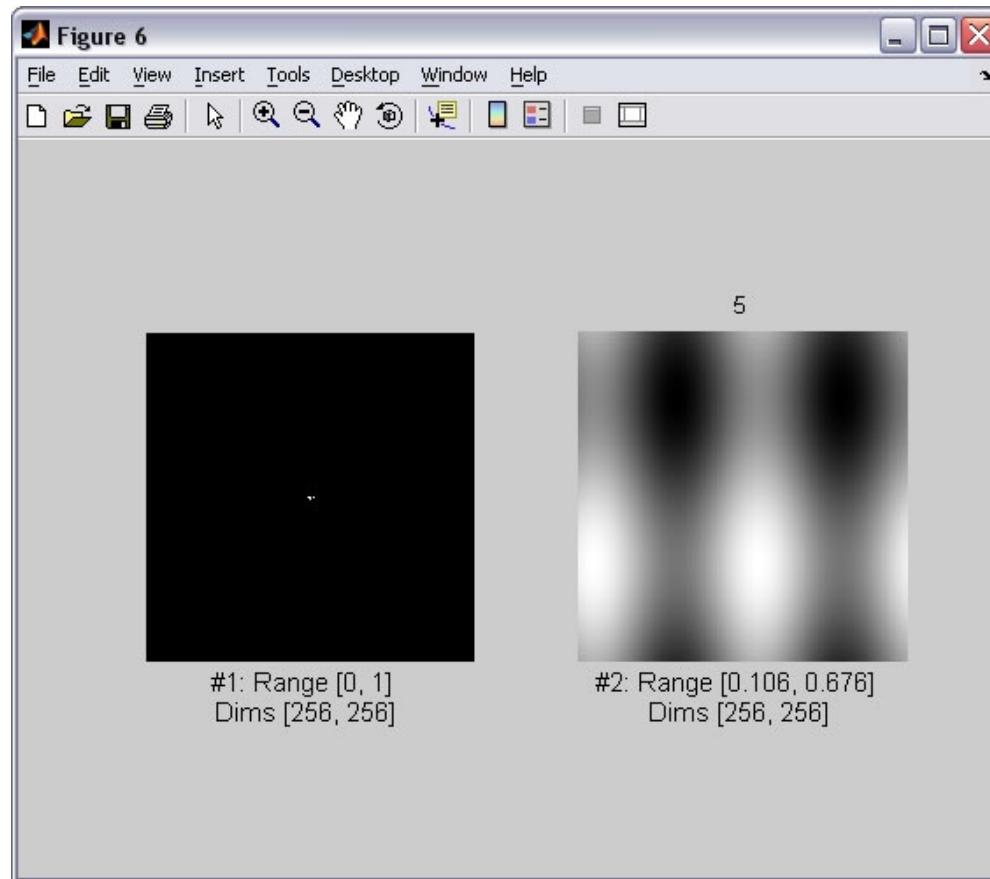
3



Now, a similar sequence of images, but selecting Fourier components in descending order of magnitude.

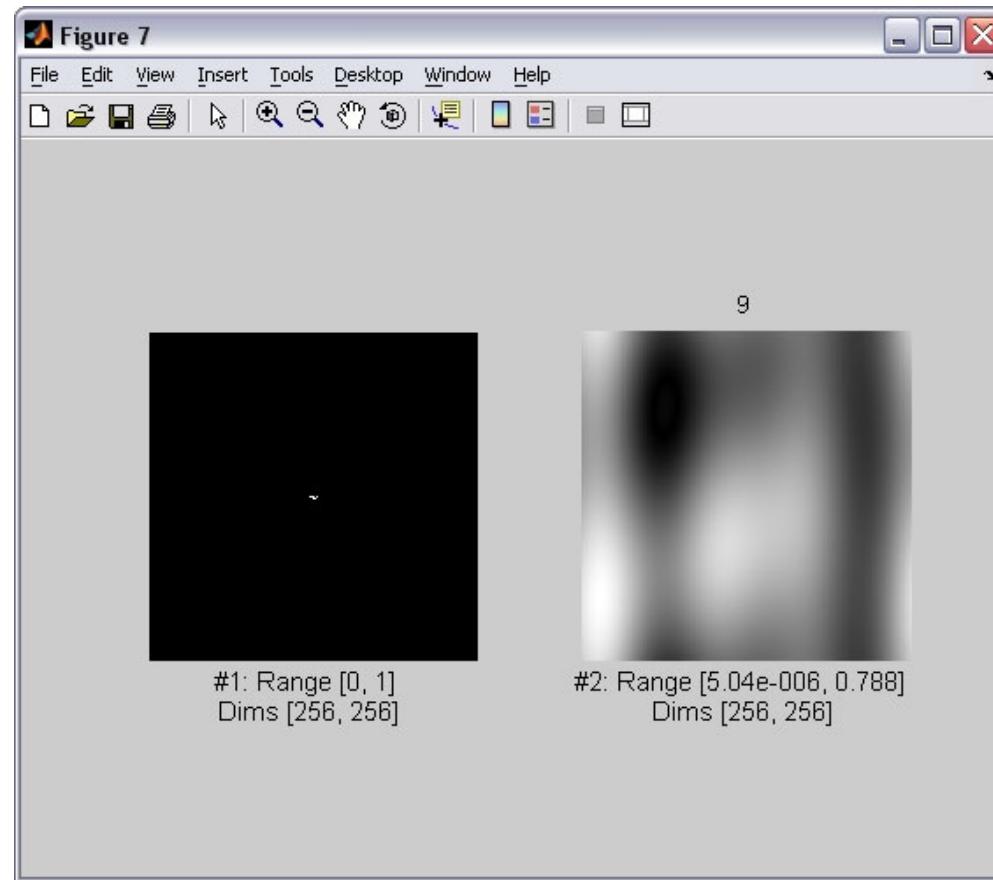


# 5



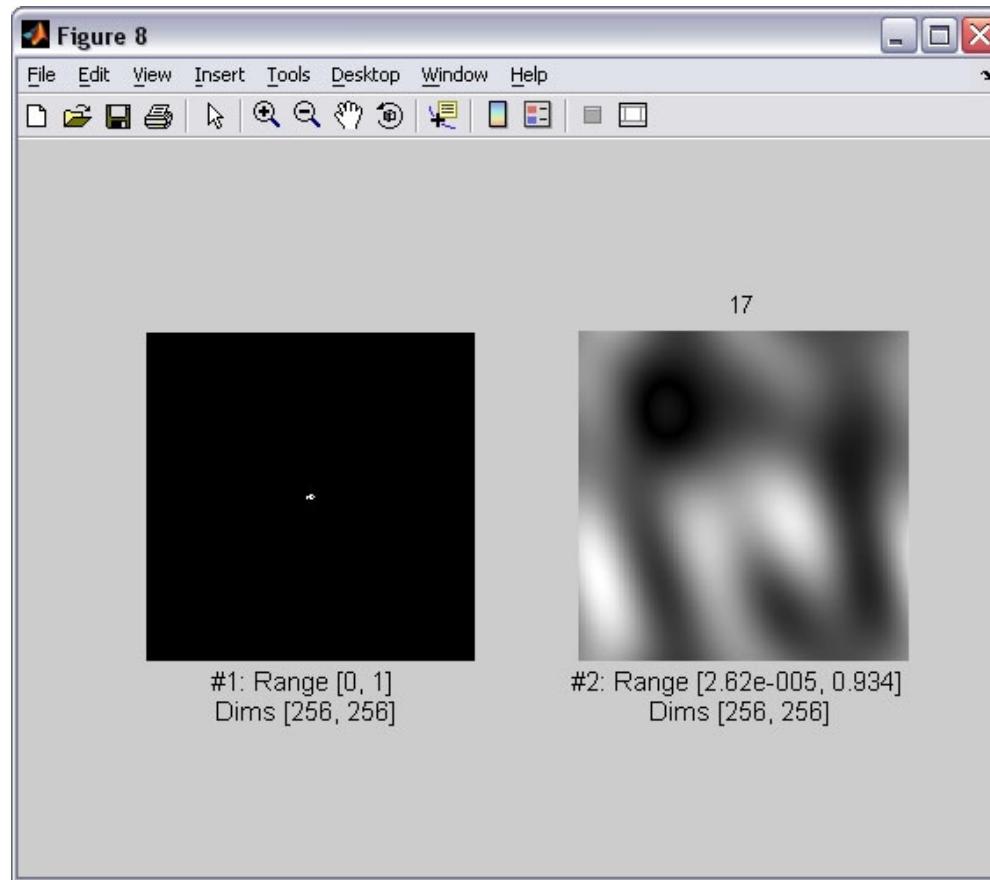


9



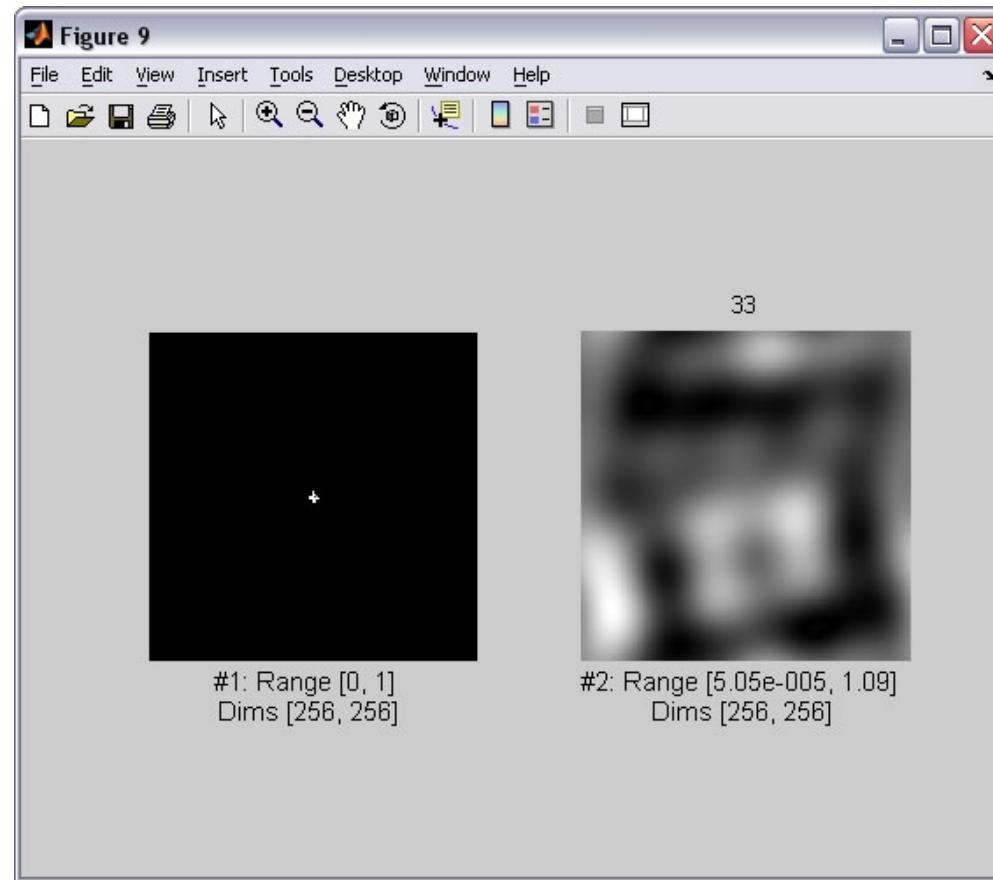


17



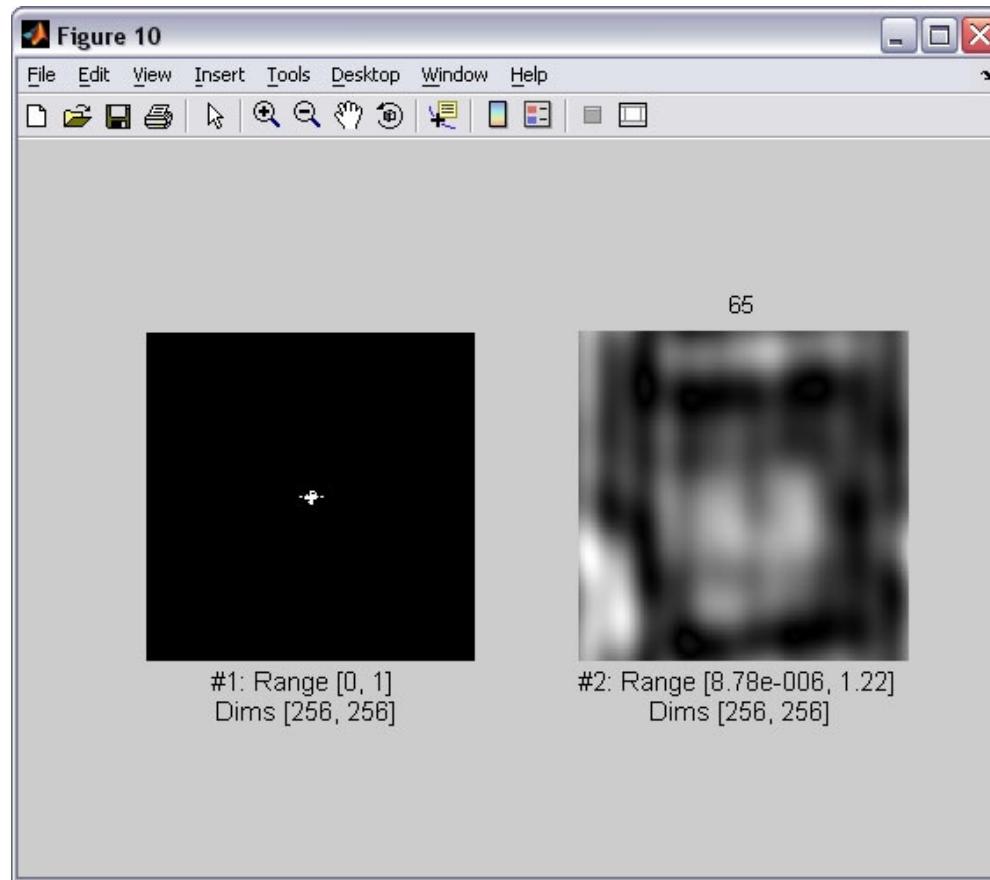


33



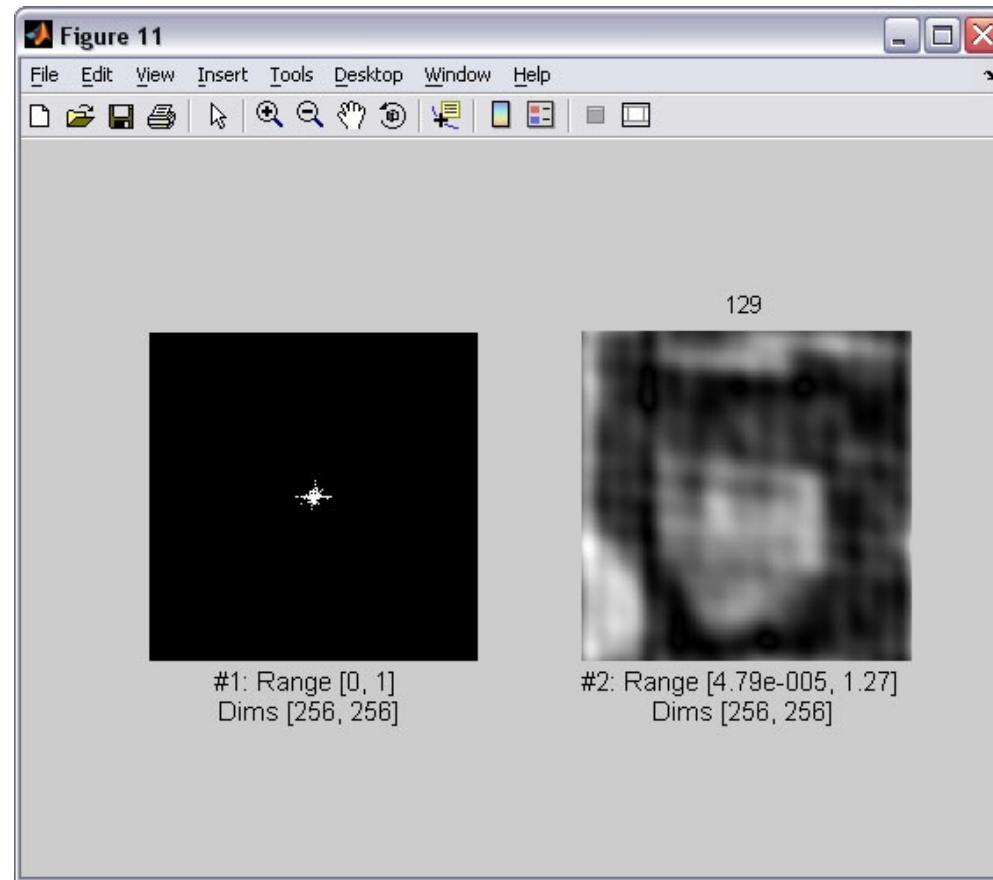


65



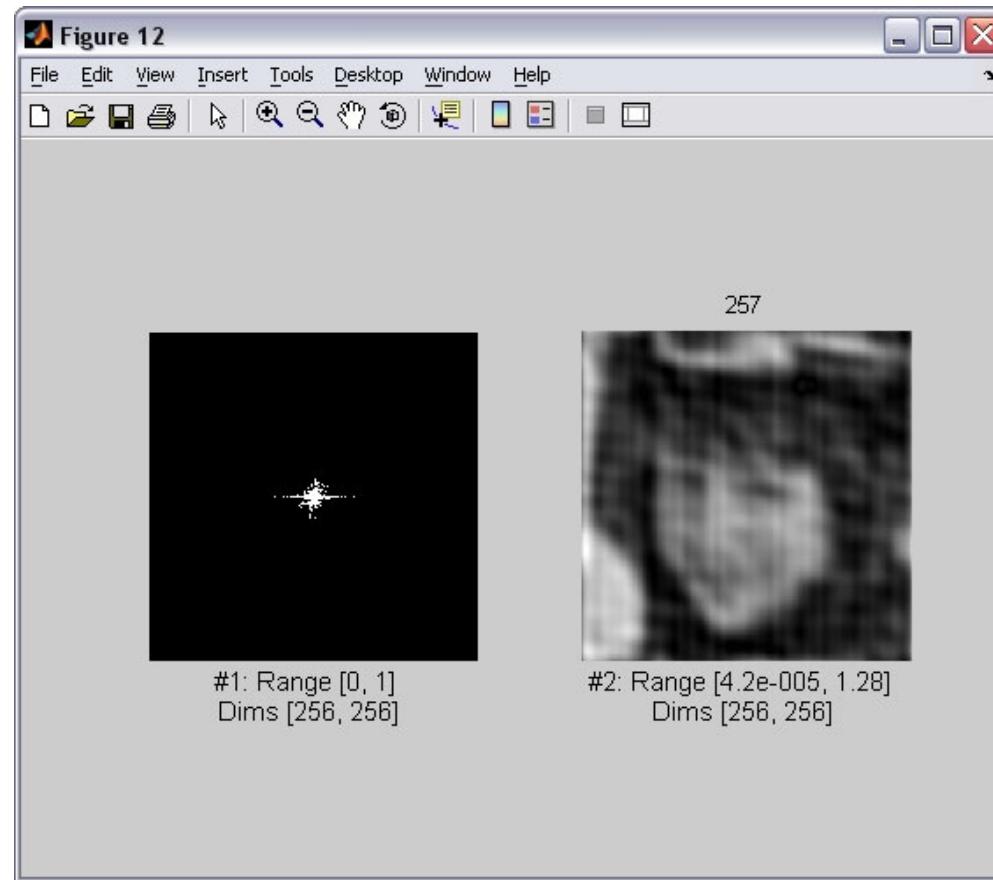


129



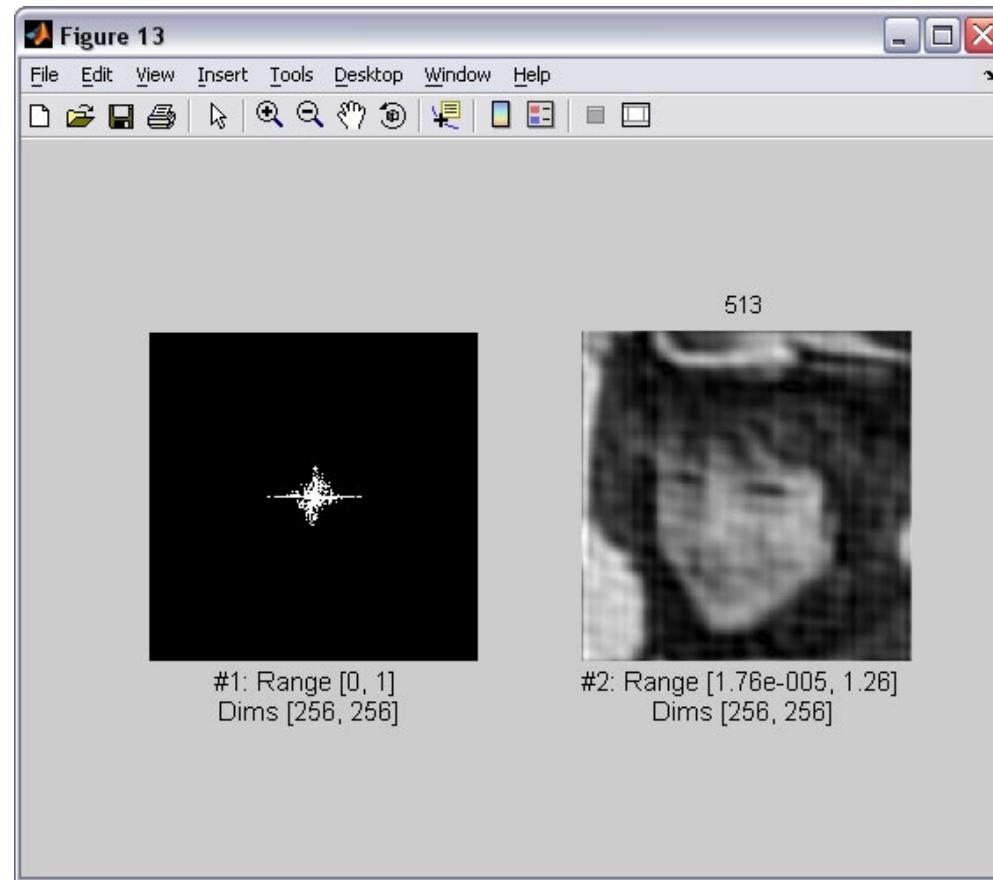


257



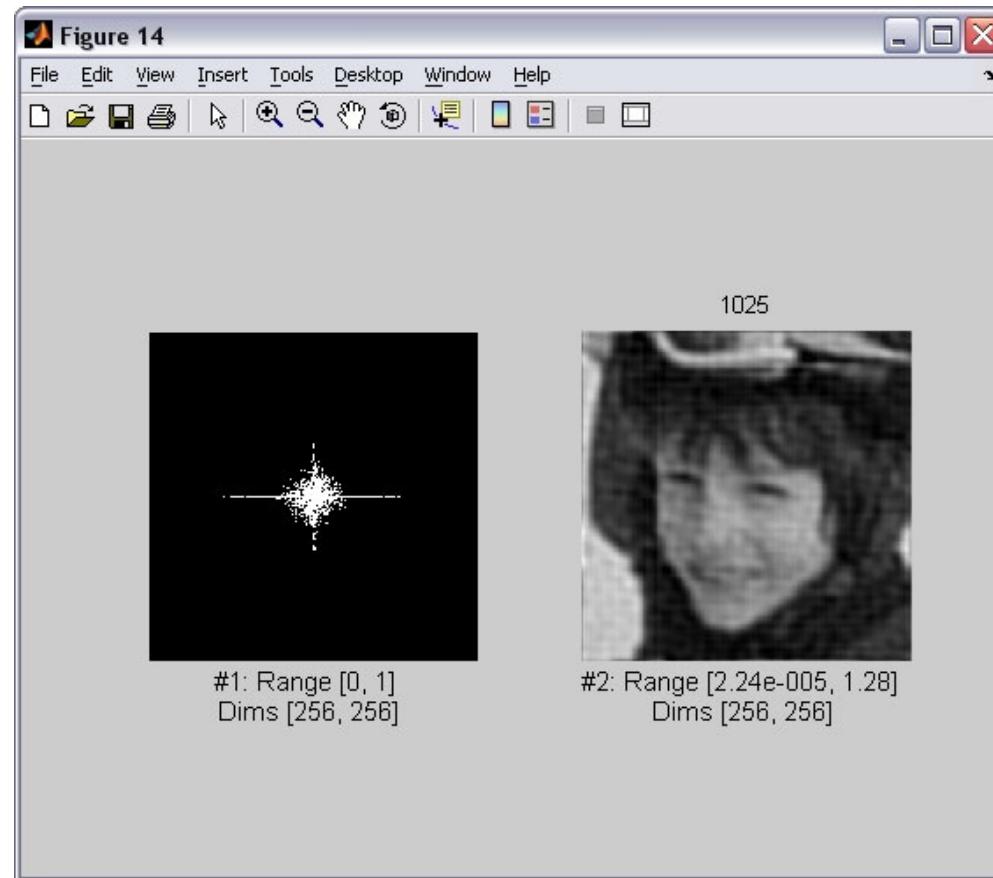


513



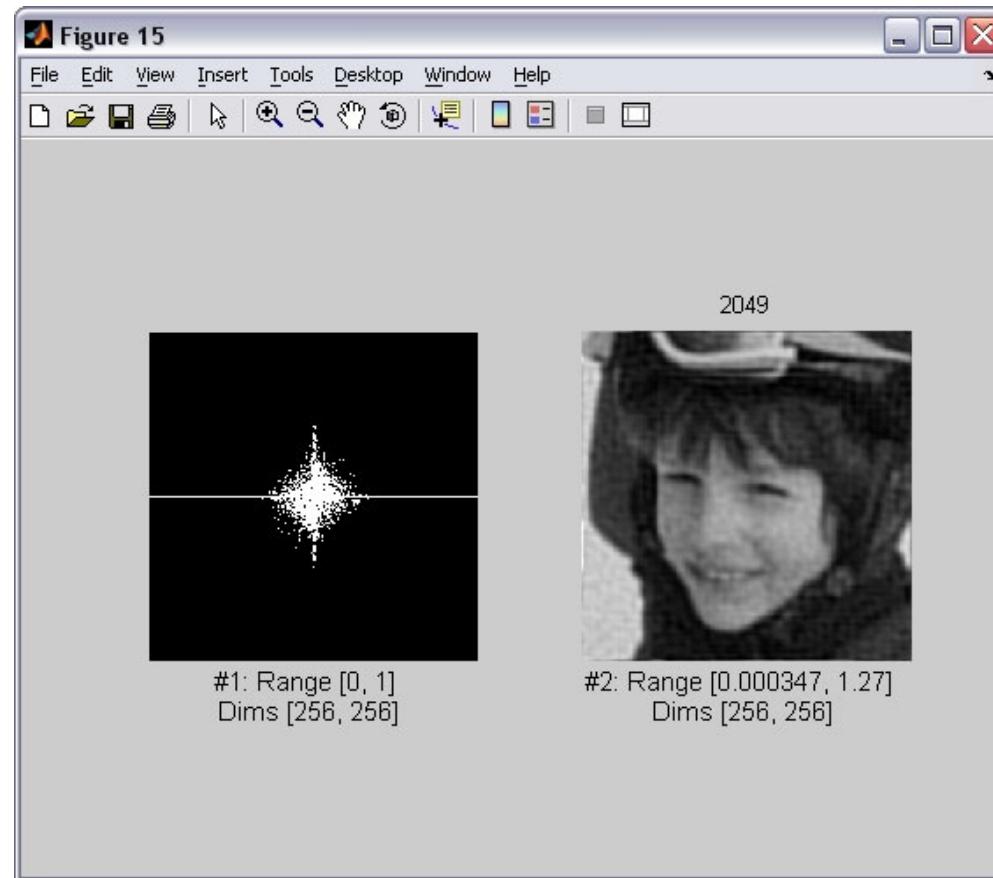


# 1025



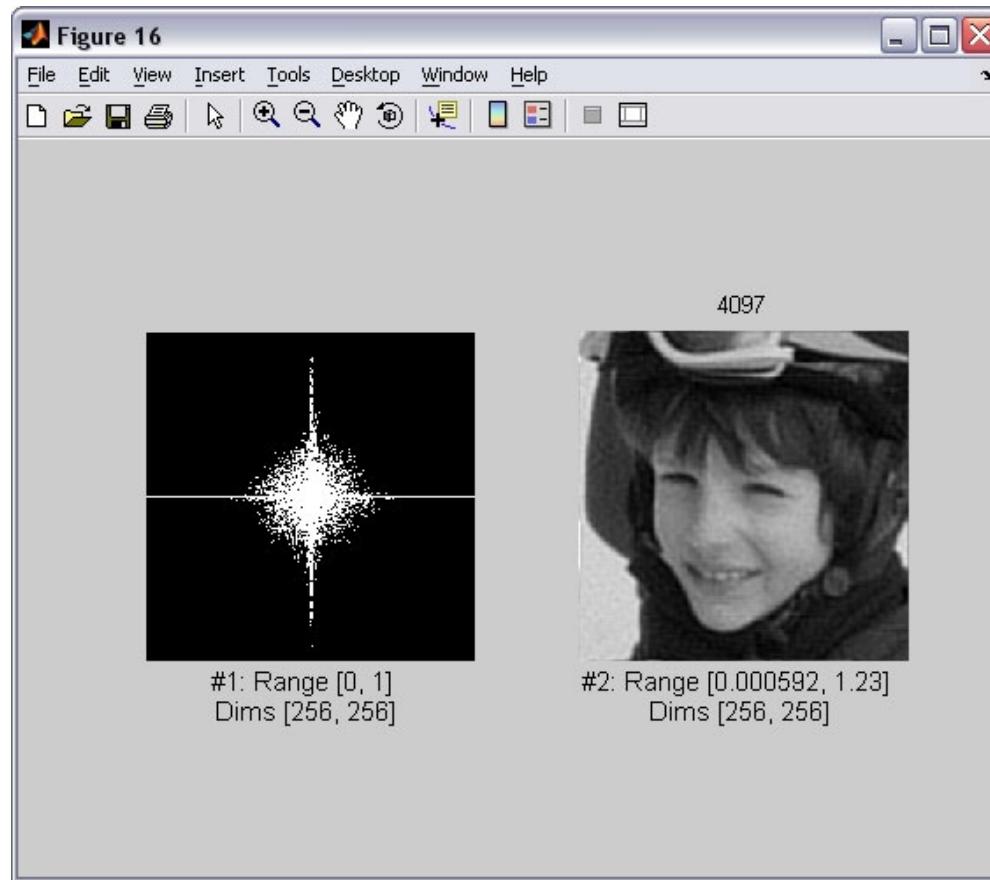


**2049**



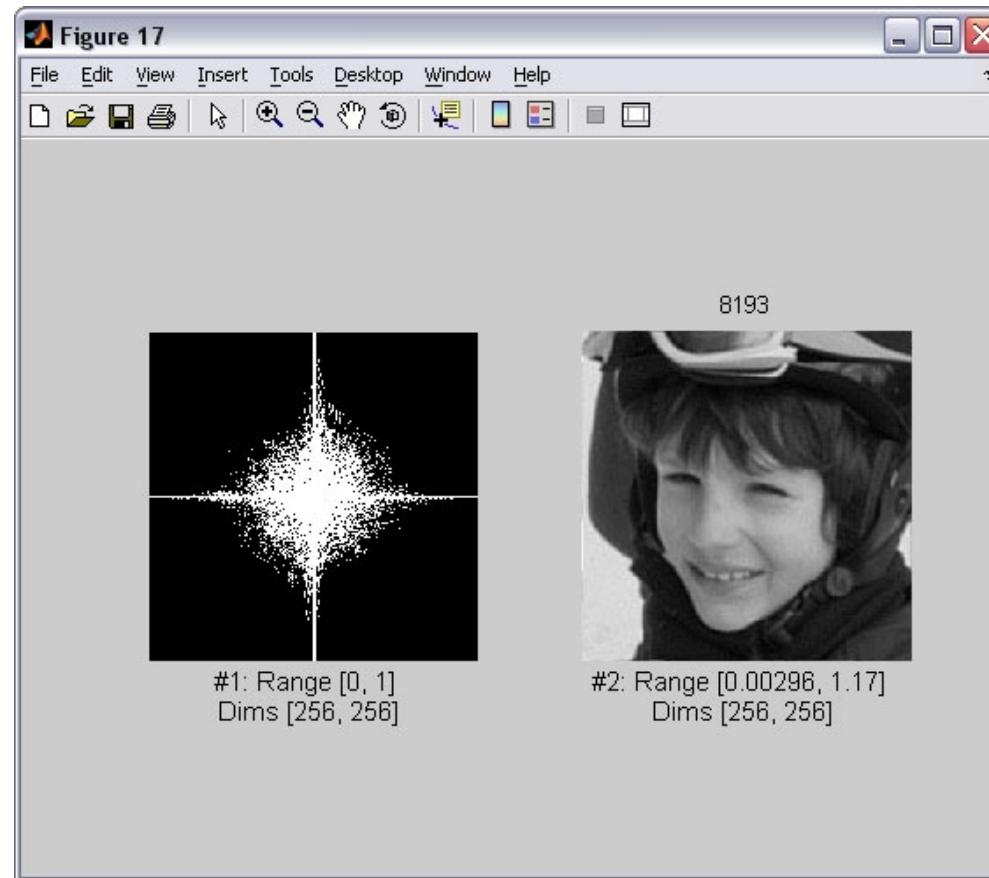


# 4097



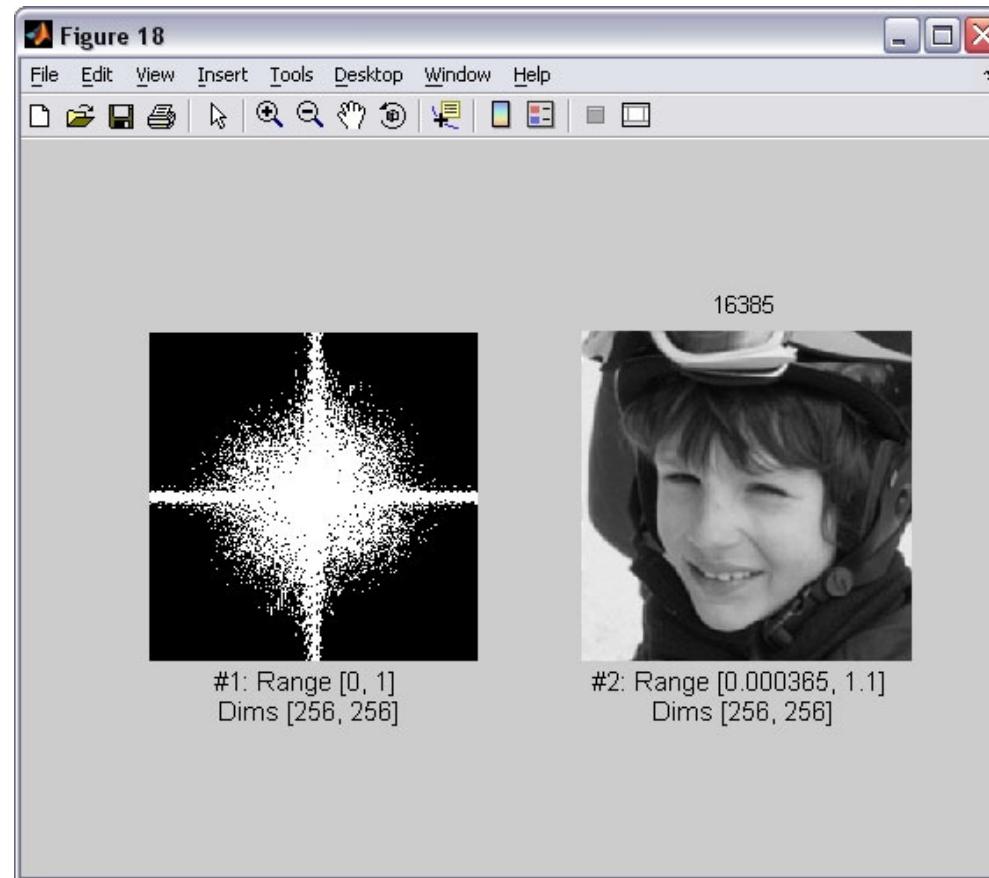


8193



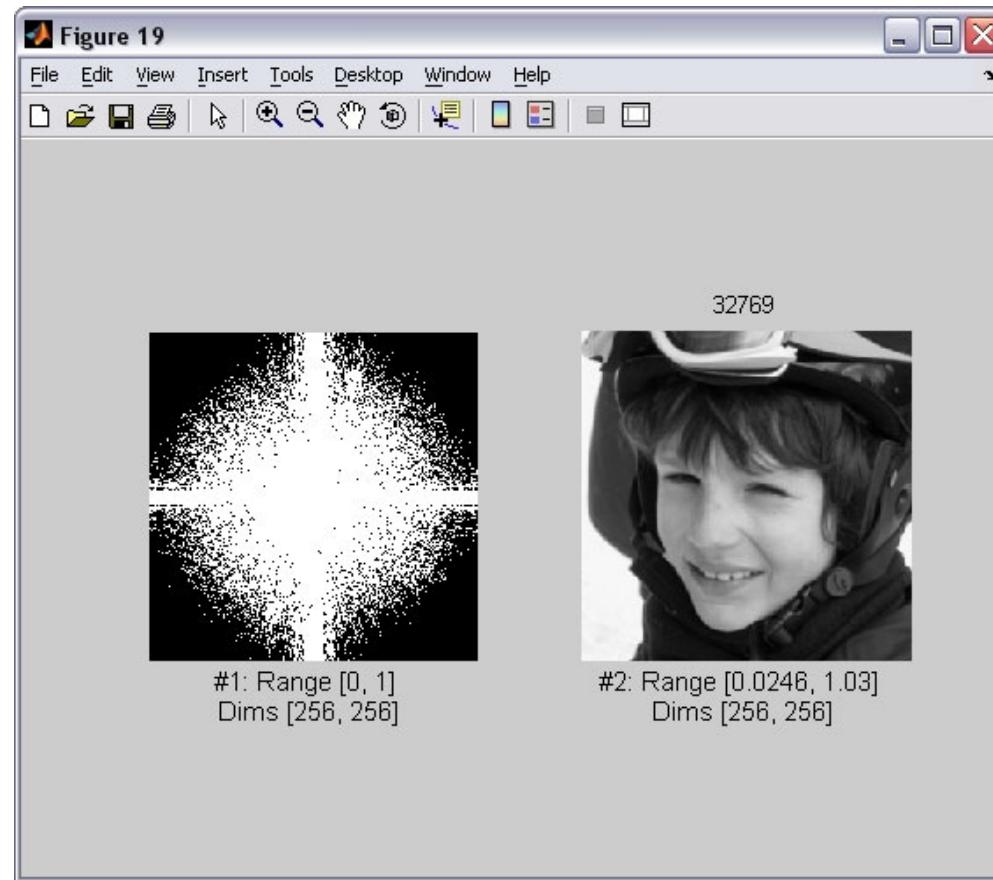


**16385**



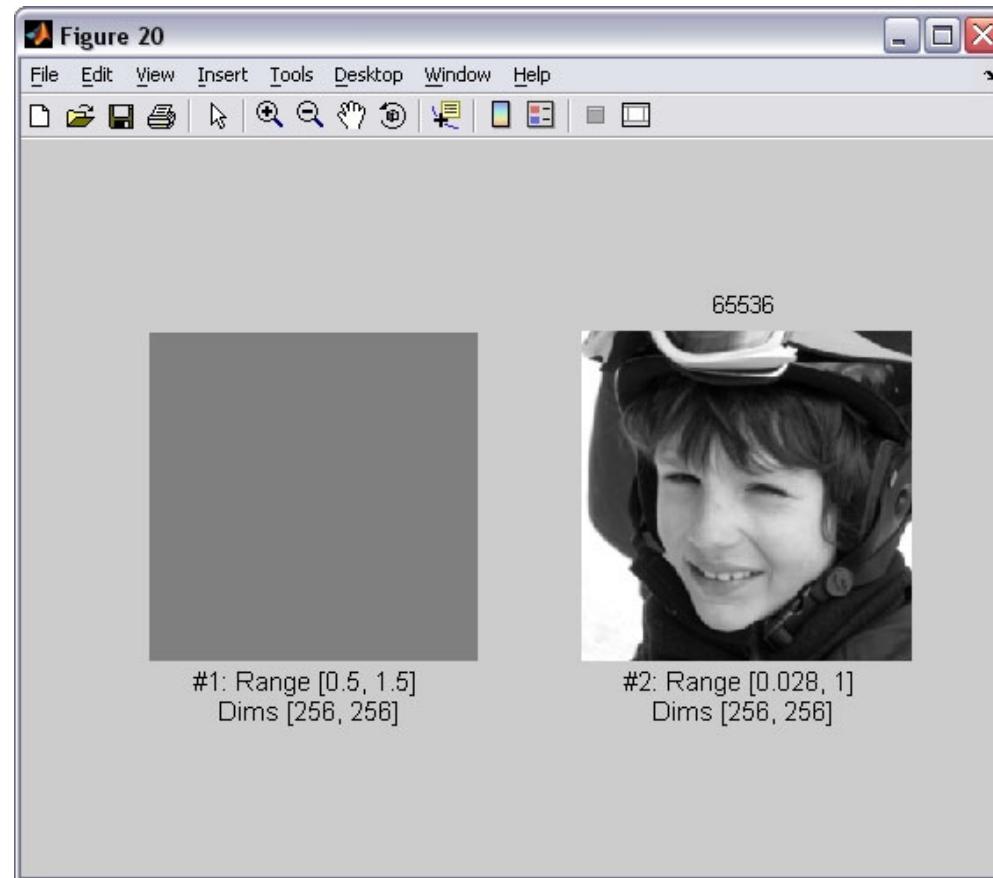


**32769**



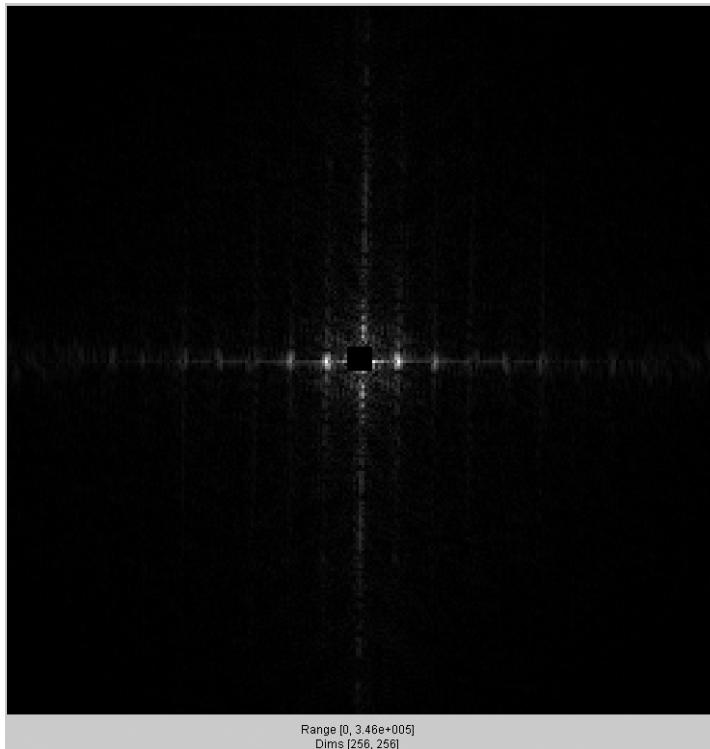


# 65536



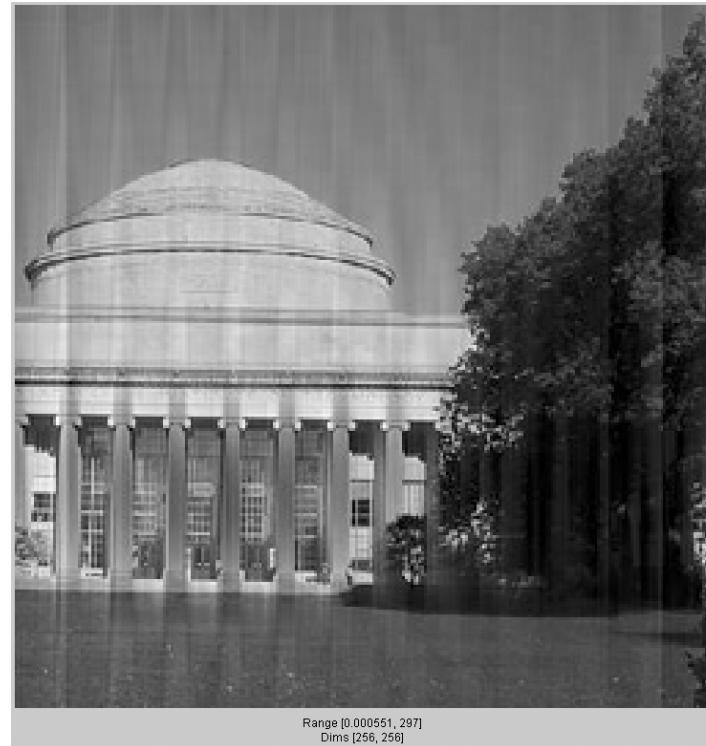
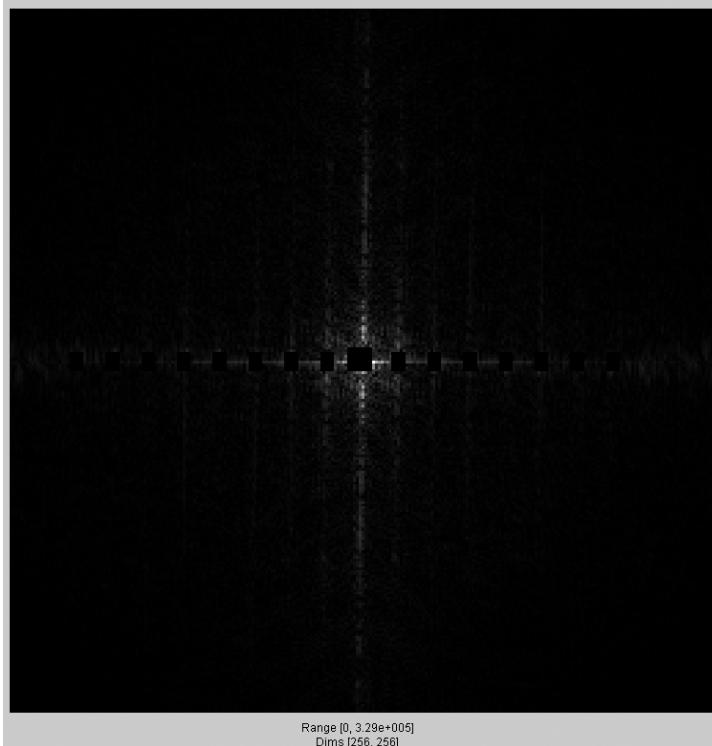


# Fourier transform magnitude





# Masking out the fundamental and harmonics from periodic pillars





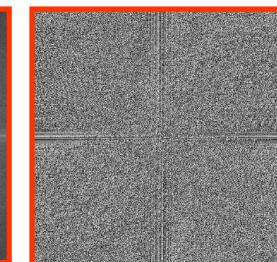
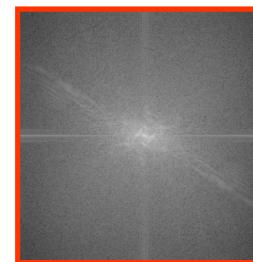
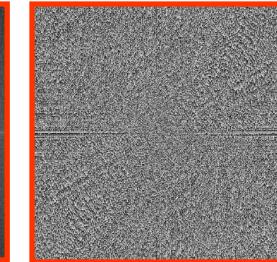
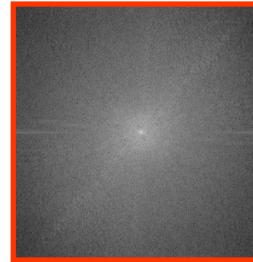
# Fourier Transform

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform

Magnitude



Phase





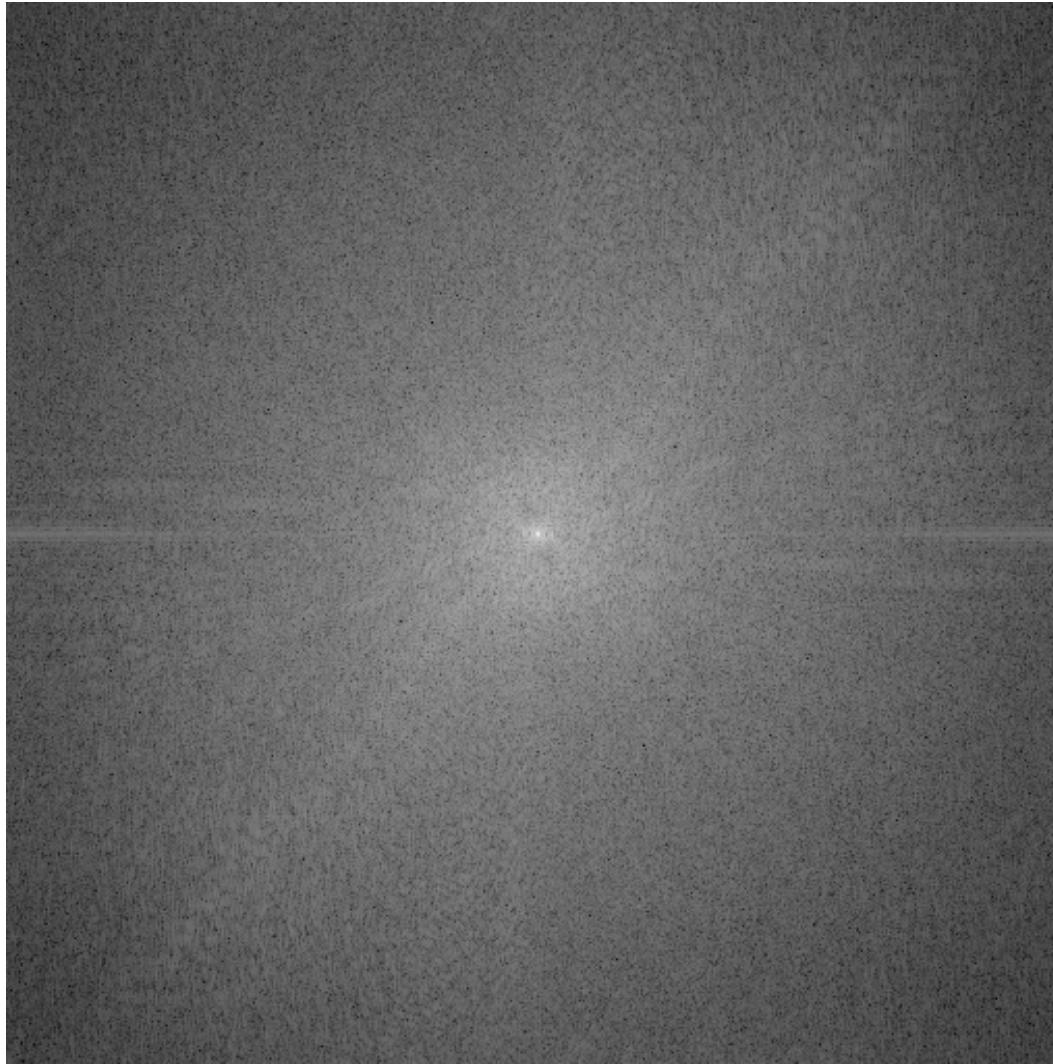
# Phase and Magnitude

- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?



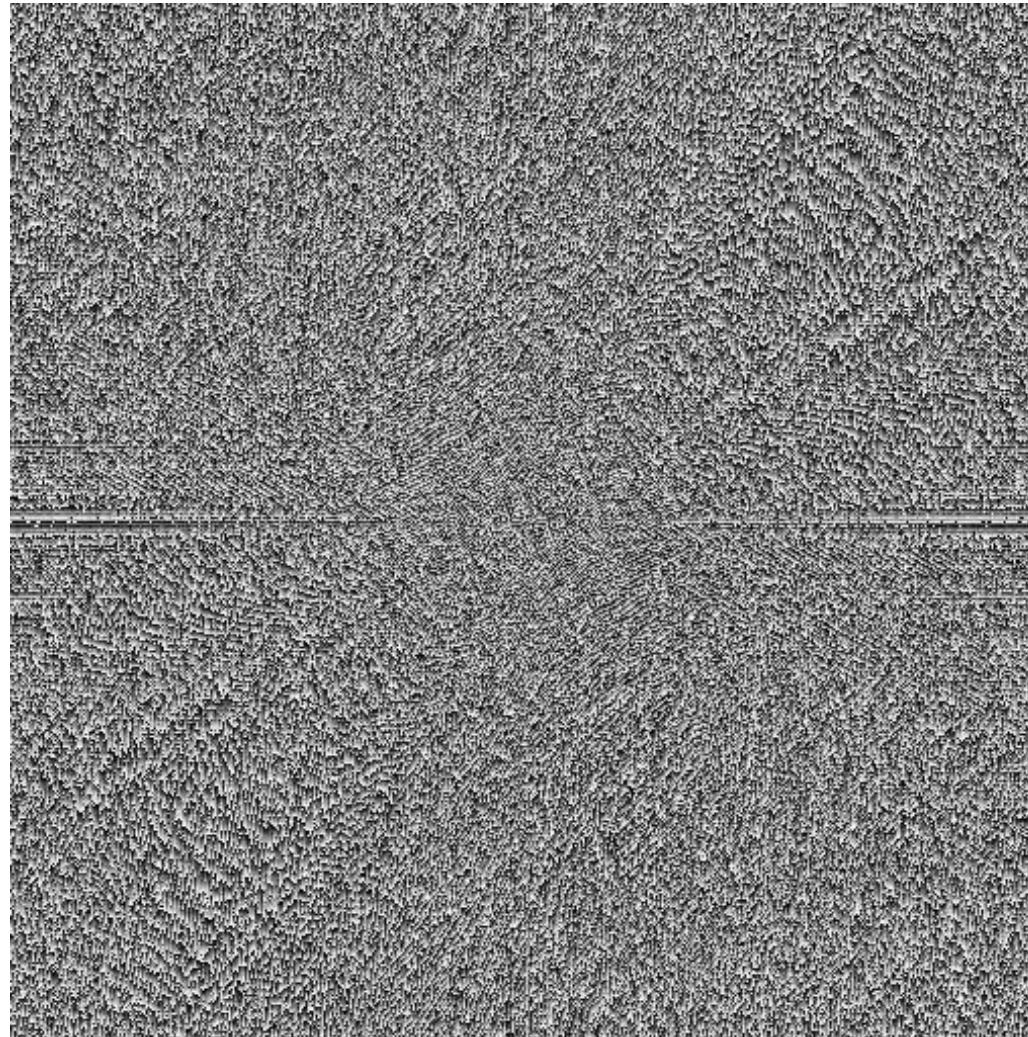


This is the  
magnitude  
transform  
of the  
cheetah pic





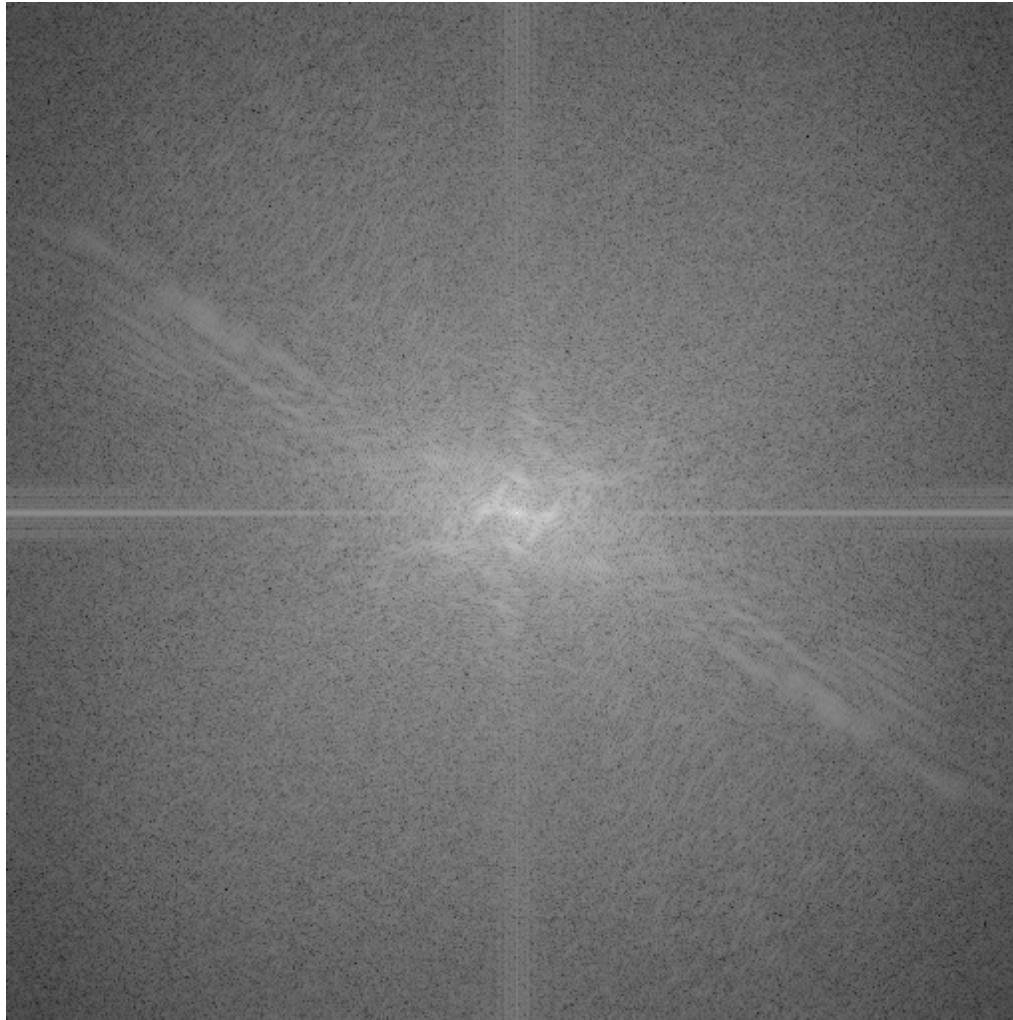
This is the  
phase  
transform  
of the  
cheetah pic





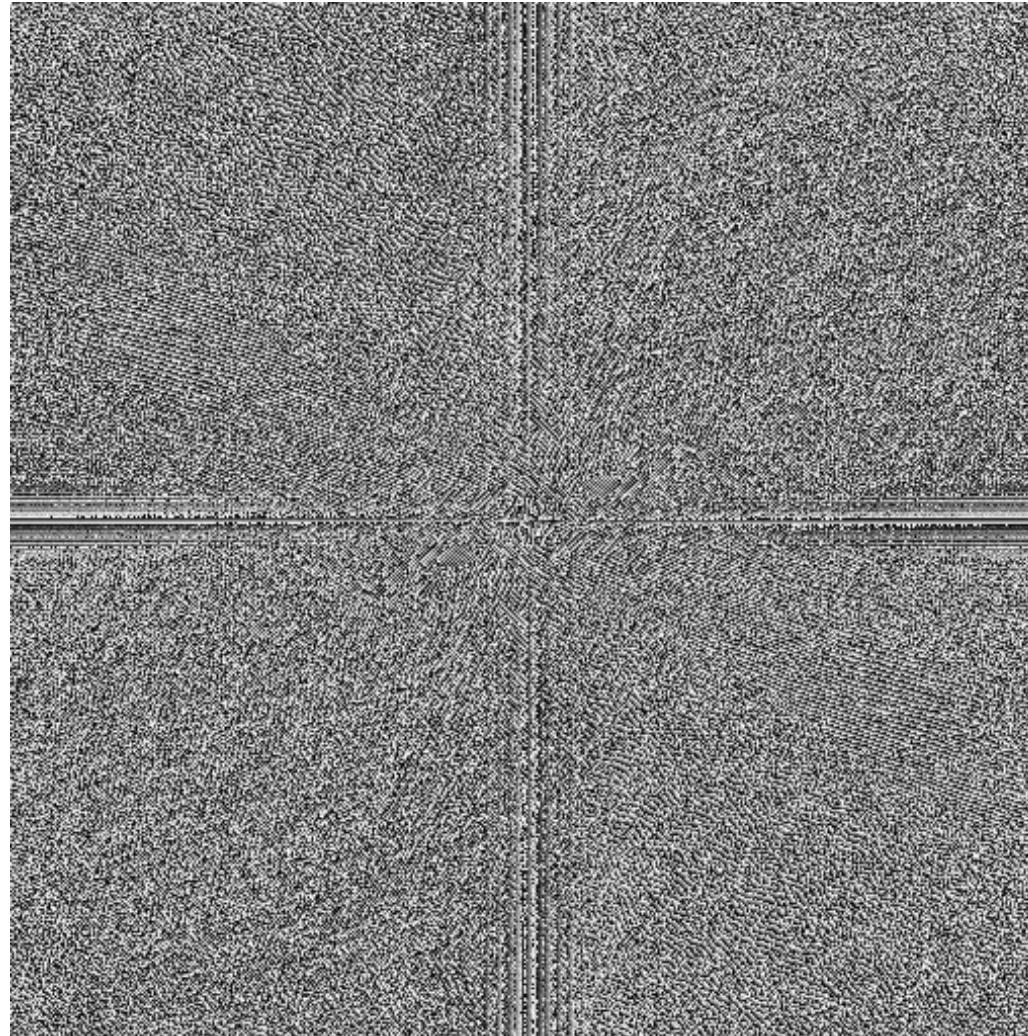


**This is the  
magnitude  
transform  
of the  
zebra pic**



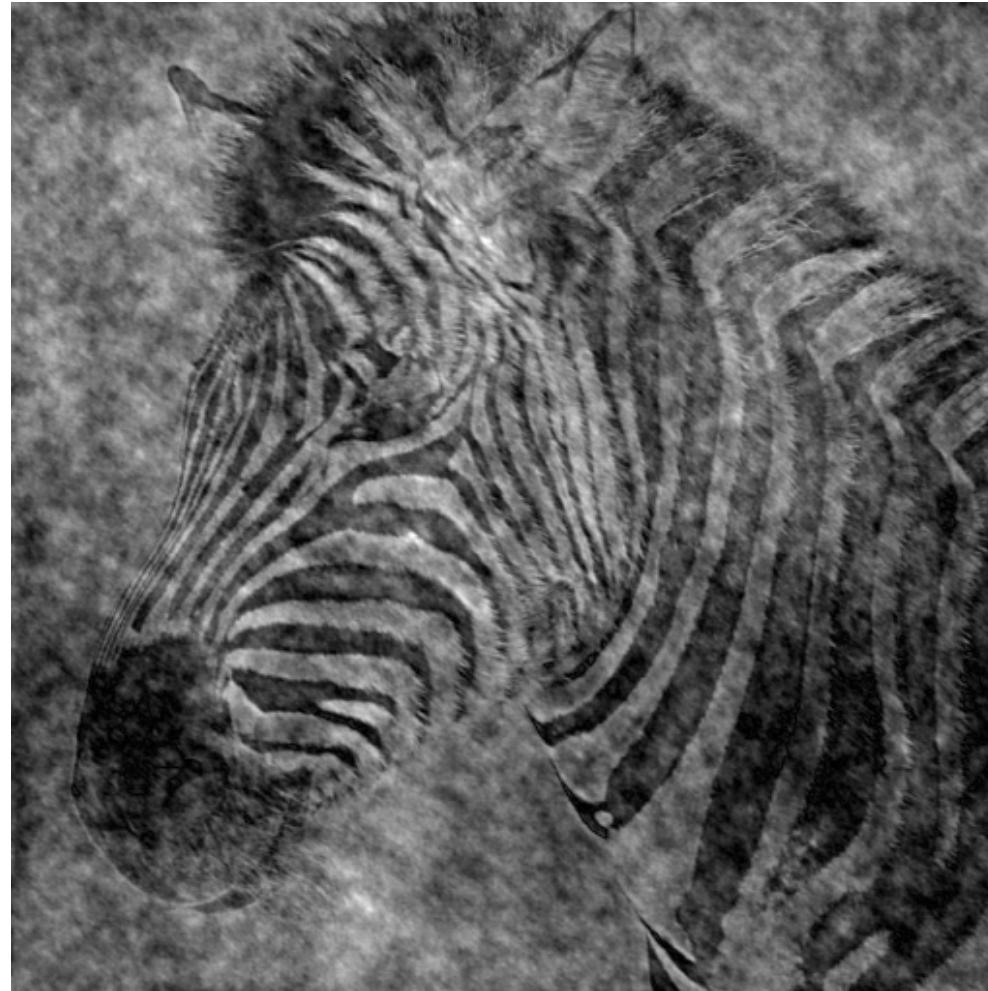


This is the  
phase  
transform  
of the  
zebra pic





**Reconstruction  
with zebra  
phase, cheetah  
magnitude**





**Reconstruction  
with cheetah  
phase, zebra  
magnitude**





# Phase and Magnitude



Image with cheetah phase  
(and zebra magnitude)

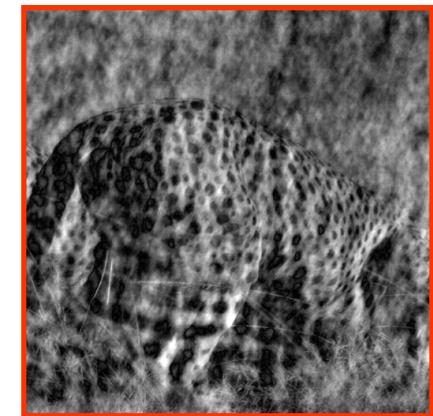
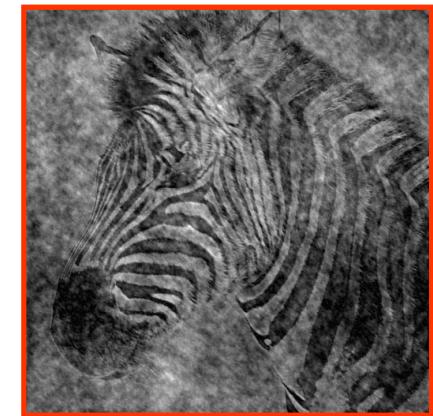
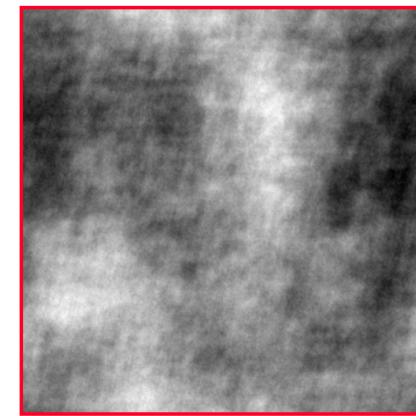
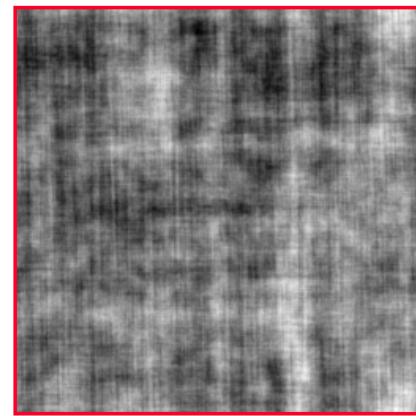
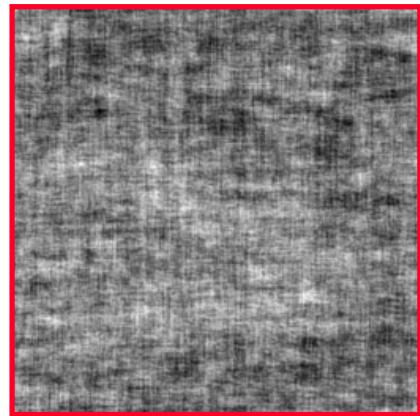


Image with zebra phase  
(and cheetah magnitude)



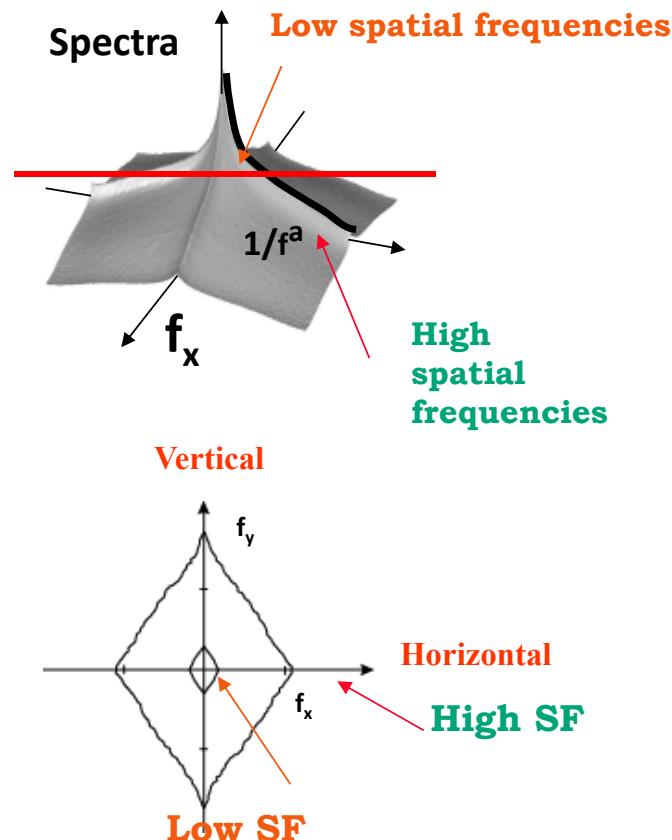


# Randomizing the phase



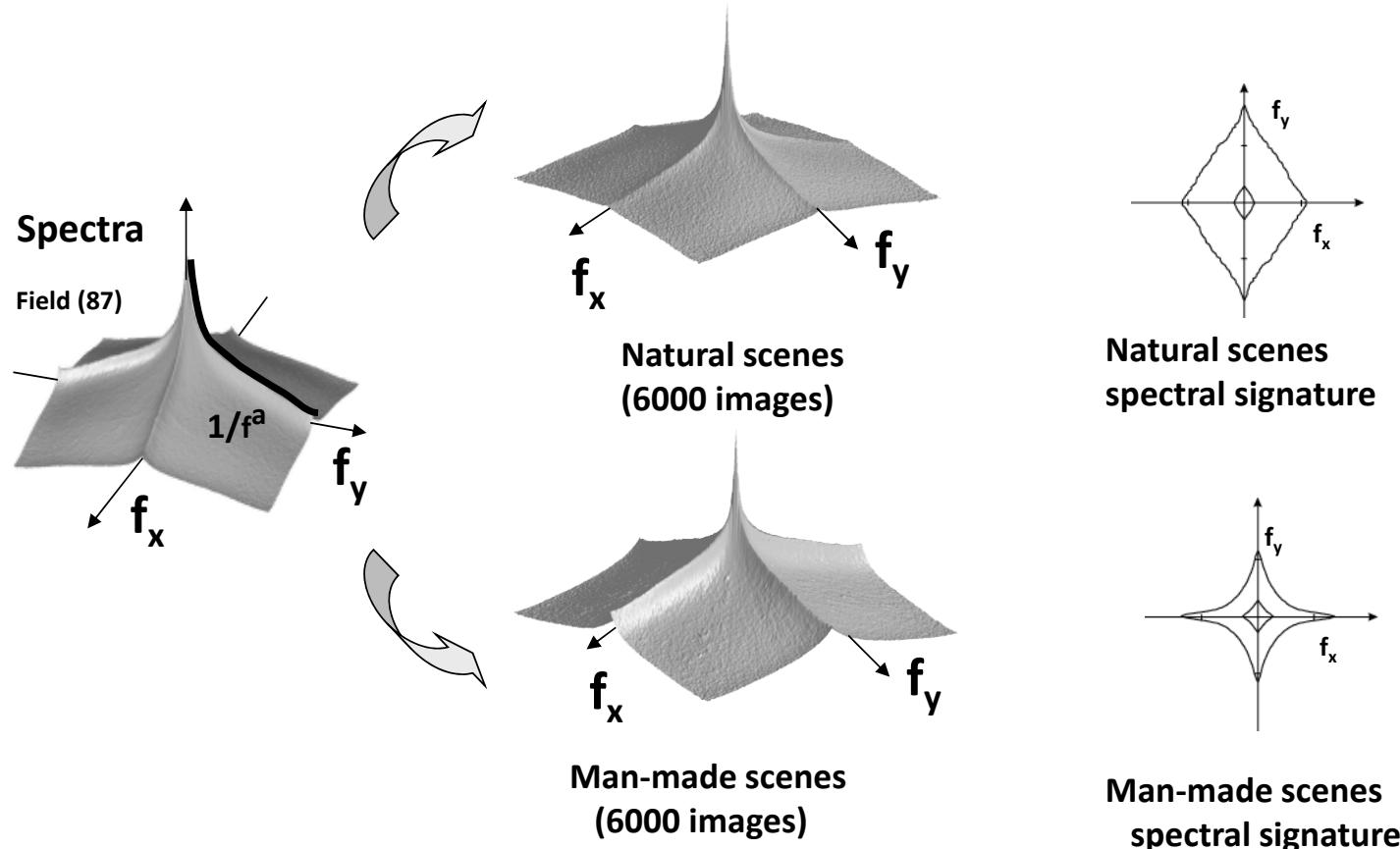


# Fourier Characteristics of Natural Images



Torralba and Oliva, *Statistics of Natural Image Categories*. Network: Computation in Neural Systems 14 (2003) 391-412.

## Power Spectrum of Images

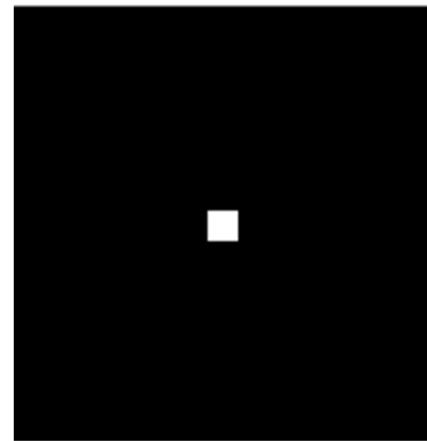
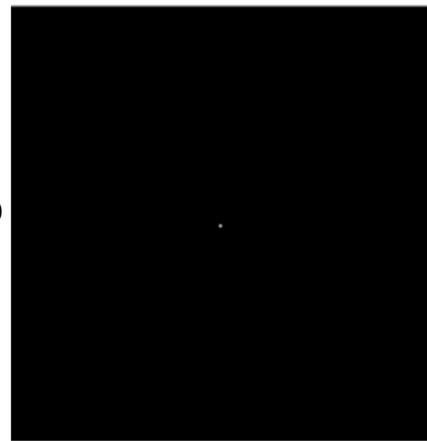


Torralba and Oliva, *Statistics of Natural Image Categories*. Network: Computation in Neural Systems 14 (2003) 391-412.

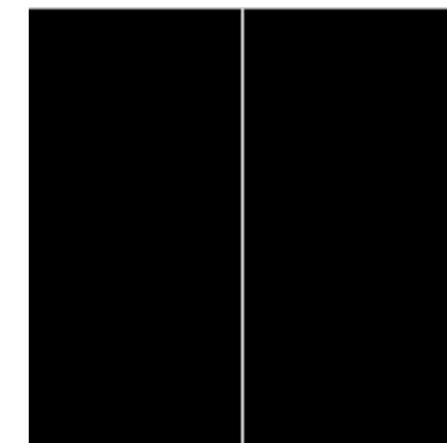
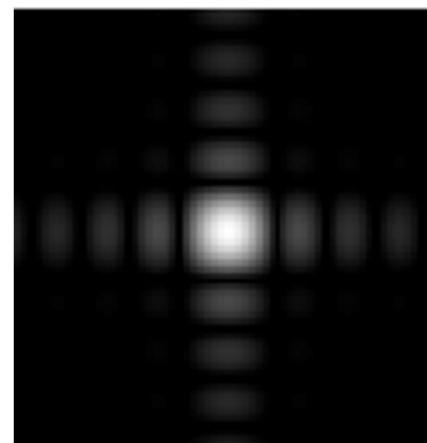
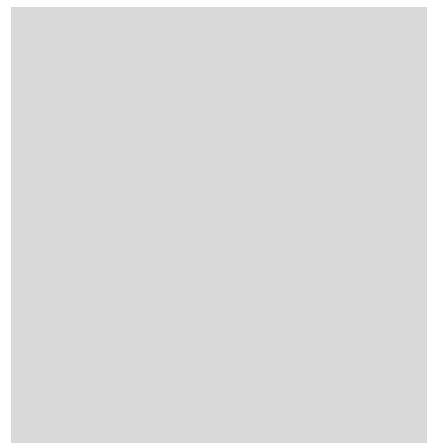


# Some important Fourier Transforms

Image



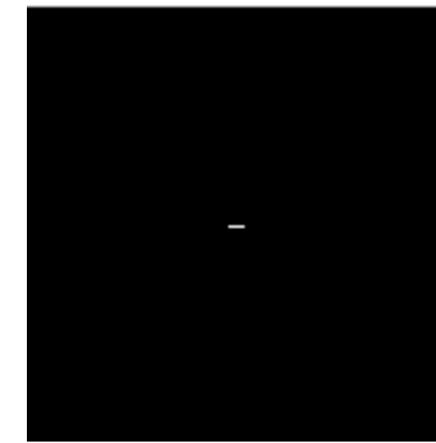
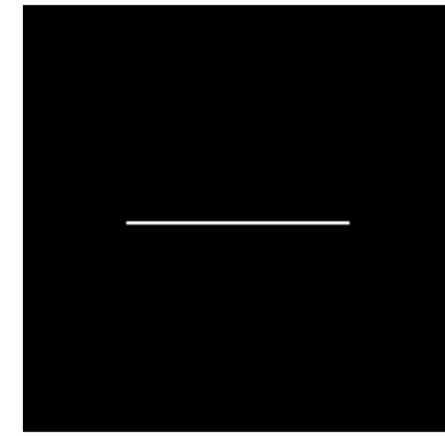
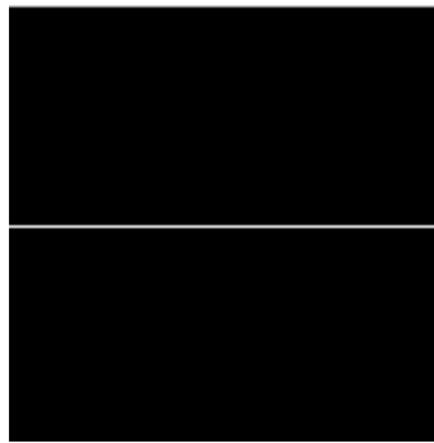
Magnitude FT



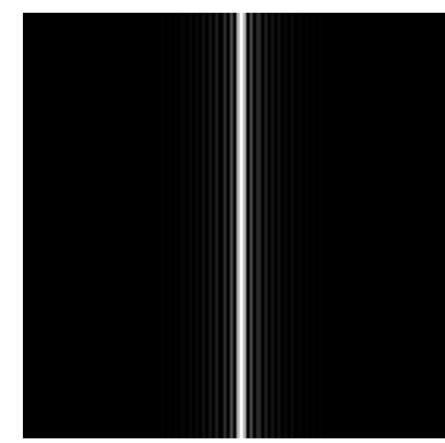
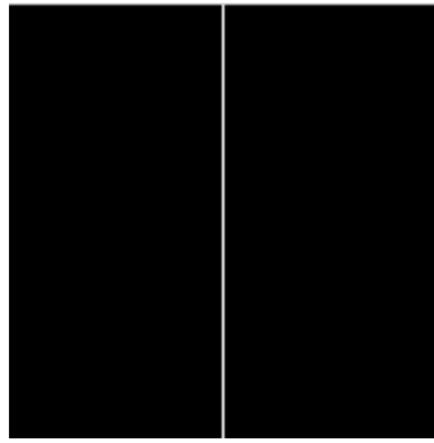


# Some important Fourier Transforms

Image



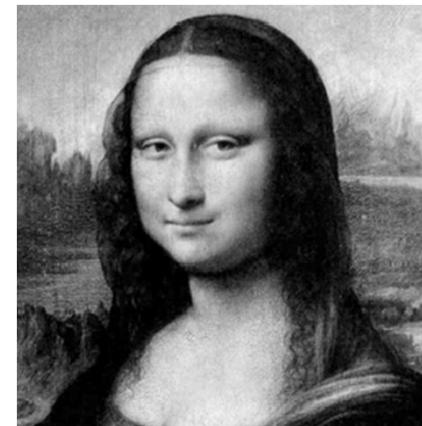
Magnitude FT



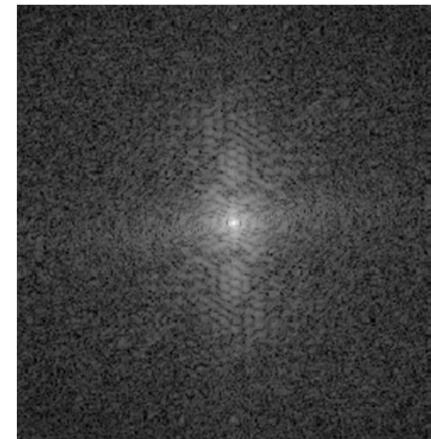
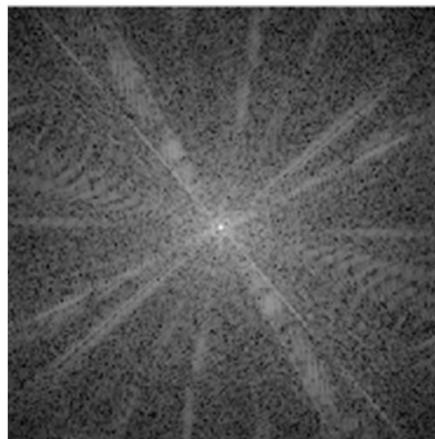


## The Fourier Transform of some important images

Image



Log(1+Magnitude FT)





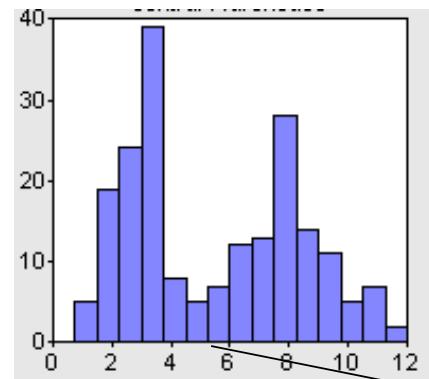
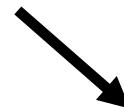
## Binary Images



- **Images with only two values (0 or 1)**
- **Simple to process and analyze**
- **Very useful for industrial applications**



# Selecting a Threshold



Bimodal Histogram

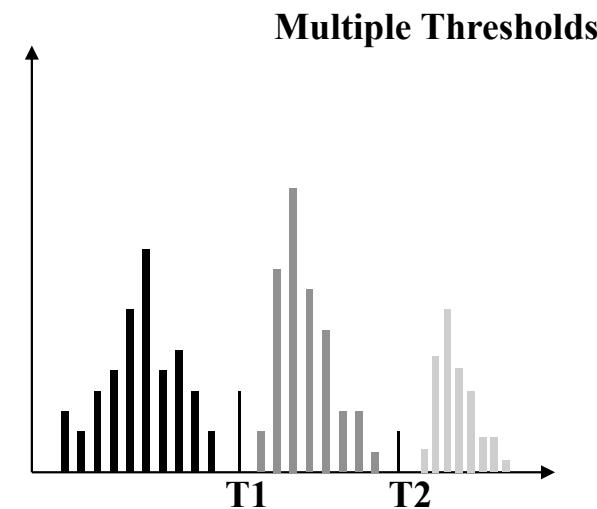
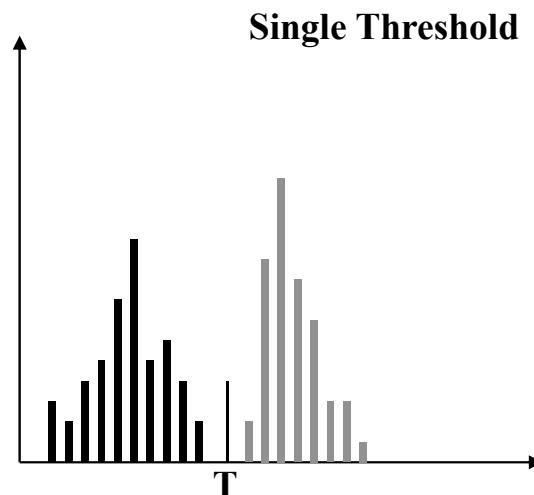
Threshold





# Thresholding Techniques

- Thresholding is one of the most important approaches to image segmentation
- In this method, pixels that are alike in grayscale (or some other feature) are grouped together





## Thresholding

- Often a image histogram is used to determine the best setting for the threshold(s)
- Some images (such as scanned text) will tend to be bimodal and a single threshold is suitable
- Other images may have multiple modes and multiple thresholds may be helpful
- In general multilevel thresholding is less reliable than single level thresholding. Mostly because it is very difficult to determine thresholds that adequately separate objects of interest



## Thresholding Decision

- Thresholding may be viewed as an operation that involves tests against a function  $T$  of the form

$$T = T[x, y, (p(x, y), f(x, y))]$$

where  $f(x, y)$  is the graylevel at the point  $(x, y)$  and  $p(x, y)$  denotes some local property of the point (such as the average gray level of a neighbourhood centred on  $(x, y)$ ).

- A thresholded image is defined as

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

Thus pixels labelled 1, say, correspond to objects, and pixels labelled 0, say, correspond to the background



## Local versus Global Thresholding

- If  $T$  depends only on  $f(x,y)$ , the threshold is called *global*
- If  $T$  depends on both  $f(x,y)$  and neighbourhood  $p(x,y)$  the threshold is called *local*
- If, in addition,  $T$  depends on the spatial coordinates  $x$  and  $y$ , the threshold is called *dynamic*

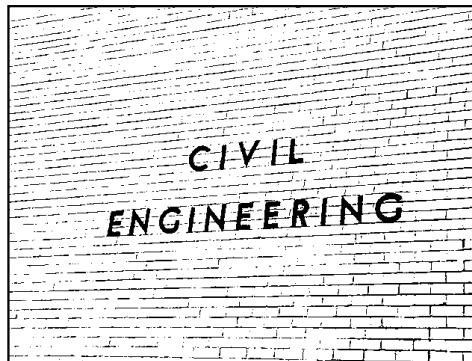


## Example of Global Thresholding

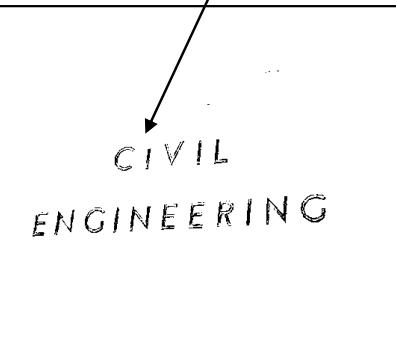
Global  
Thresholding



Increasing Threshold



Foreground  
“washed out”





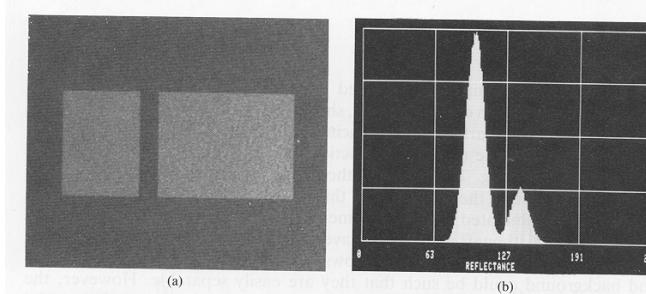
## The Role of Illumination

- The formation of an image can be viewed as the product of a reflectance component  $r(x,y)$  and an illumination component  $i(x,y)$
- Even though the reflectance of an object may have a histogram that allows easy separation of foreground and background, the effect of non-uniform illumination is to smear the histogram making simple thresholding ineffective
- If we know the illumination function (through calibration say), then we can sometimes eliminate this effect making global thresholding a practical method

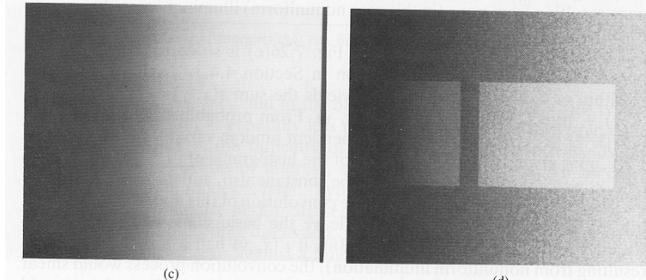


## Example

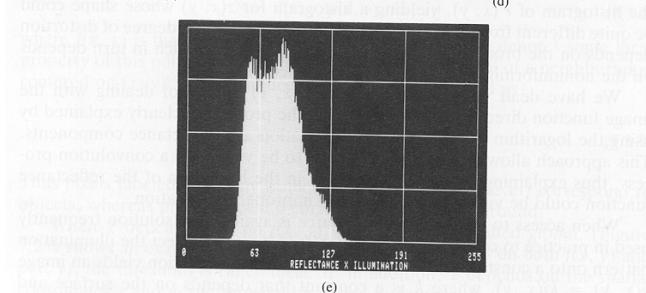
Reflectance  
Function



Illumination  
Function



Reflectance  
Histogram



Image

Image  
Histogram



## Simple Global Thresholding

- In practice, global thresholding can be expected to be successful in highly controlled environments such as industrial inspection applications, where illumination control is feasible
- Note that uniform illumination is required for this method to work, or at least some sort of compensation for non-uniform illumination
- Usually a successful segmentation is highly dependent on the choice of thresholds
- There are many methods for automatic determination of these thresholds

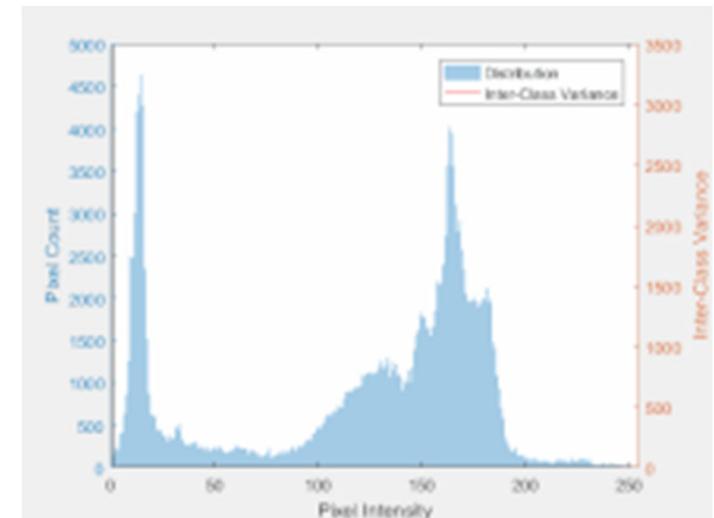


## Automatic Thresholding – Otsu's Method

Otsu's method is used to perform automatic image thresholding. In the simplest form, the algorithm returns a single intensity threshold that separate pixels into two classes, foreground and background.

Matlab graythresh()

Otsu doesn't work well if foreground area is small compared to background area



Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys. Man. Cyber.* 9 (1): 62–66.



# Adaptive Thresholding



## Adaptive Mean Threshold

threshold value is the mean of neighbourhood area

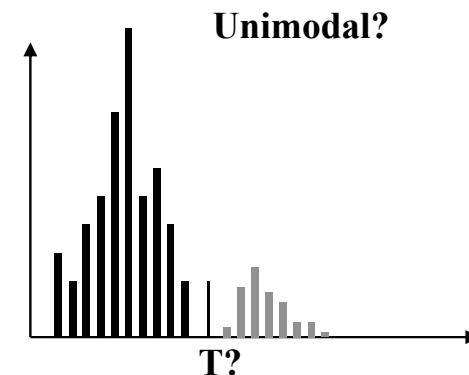
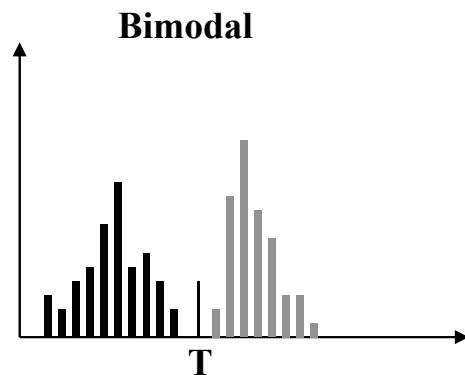
## Adaptive Gaussian Threshold

threshold value is the weighted sum of neighbourhood values where weights are a gaussian window



## Problems with Histogram

- What if the number of pixels in the foreground is much smaller than the number of pixels in the background?
- Perhaps we should only consider pixels near edges (high gradient)





## Example of Using Gradient

- Convert image into three-level image as follows

- $$s(x, y) = \begin{cases} 0 & \text{if } \nabla f < T \\ + & \text{if } \nabla f \geq T \text{ and } \nabla^2 f \geq 0 \\ - & \text{if } \nabla f \geq T \text{ and } \nabla^2 f < 0 \end{cases}$$

- This image can be converted to binary as follows
  - Scanning along either horizontally or vertically a transition from light to dark must correspond to a  $-+$  sequence, the interior is either labelled 0 or +, and the transition from dark to light will correspond to the sequence  $+-$
  - Thus a string of the form  $(\dots)(-+)(0 \text{ or } +)(+-)(\dots)$  would have the inner parenthesis labelled as object (1).
  - From this string parsing each pixel can be labelled either 0 or 1





## Motion in Segmentation

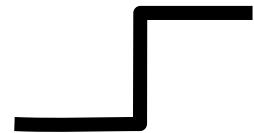
- Motion is a powerful cue used by humans to extract objects of interest from the background
- For fixed camera situations, a very common technique is simple frame differencing to detect change
- This technique is only applicable when the two frames are registered and the illumination is relatively constant
- Often isolated points in the difference image arise from noise, so some noise rejection may be necessary



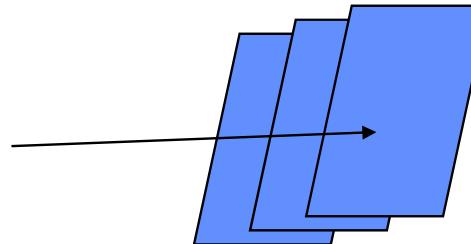
## Frame Differencing as an Edge Detector

- Frame differencing is just a method for differentiating (finding the gradient) of an image temporally rather than the more usual spatial filters.
- In that case, why not use some other edge detectors that have better properties?
- Significant improvements can be made by using more sophisticated filter kernels.

**Frame differencing  
kernel**

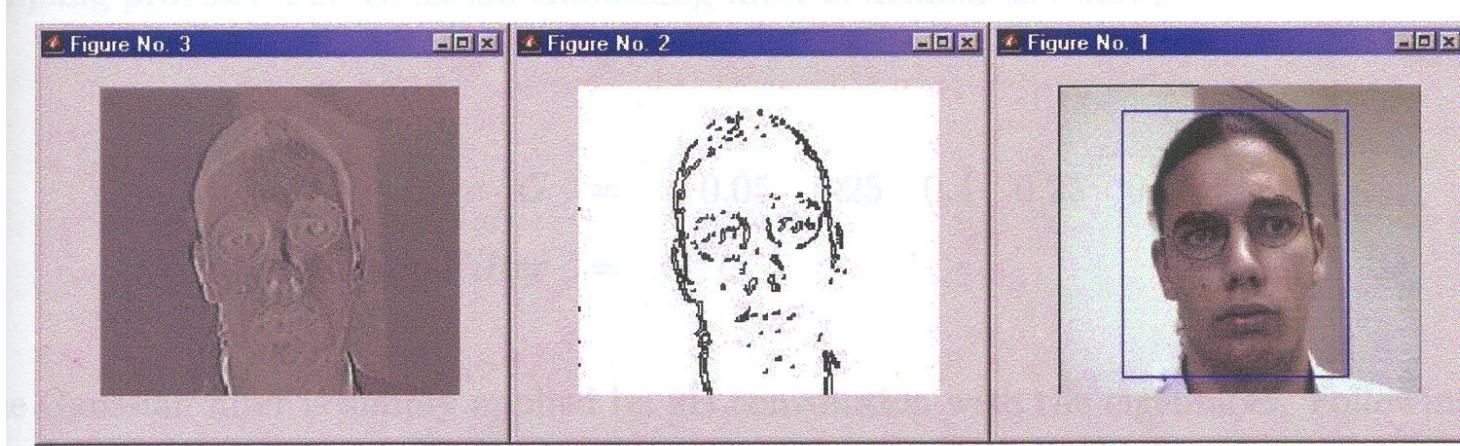


**Image Sequence**





## Temporal Filtering Example



- This example uses the zero crossings of the Laplacian of Gaussian edge detector implemented against time over several frames to detect head movement.

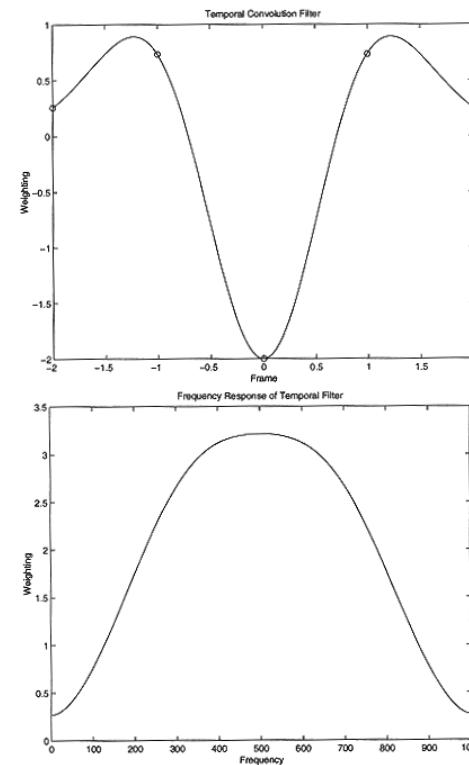


# Laplacian of Gaussian

## Temporal Convolution Filter

### Frequency Response of Temporal Filter

**Note that this is basically  
a high pass filter response  
similar to a differentiator**





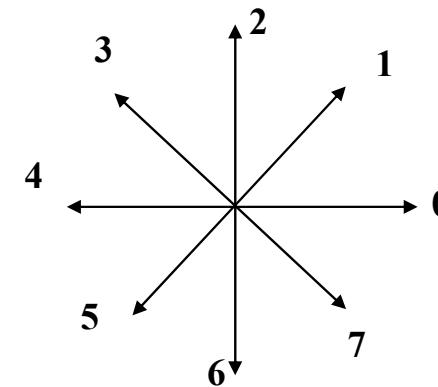
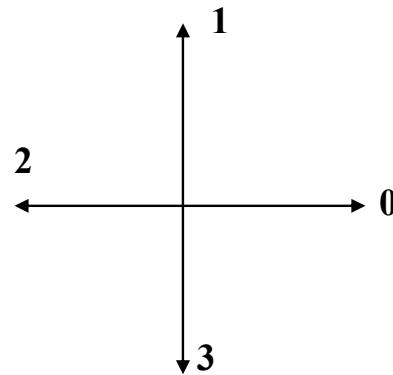
## Image Representation

- Once an image is segmented into regions, the regions must generally be described in a suitable for future processing
- There are generally two options
  - Represent the region in terms of its boundary
  - Represent the region in terms of its pixels
- The first method is often used if we are primarily concerned with the shape of a region
- The second method is more useful when we are concerned with internal properties such as colour and texture
- In general the features selected should be insensitive as possible to changes in size, translation, and rotation.



## Chain Codes

- Chain codes are used to represent boundaries by a connected sequence of straight-line segments of specified length and direction.
- Typically this representation is based on the 4 or 8 connectivity of the segments.



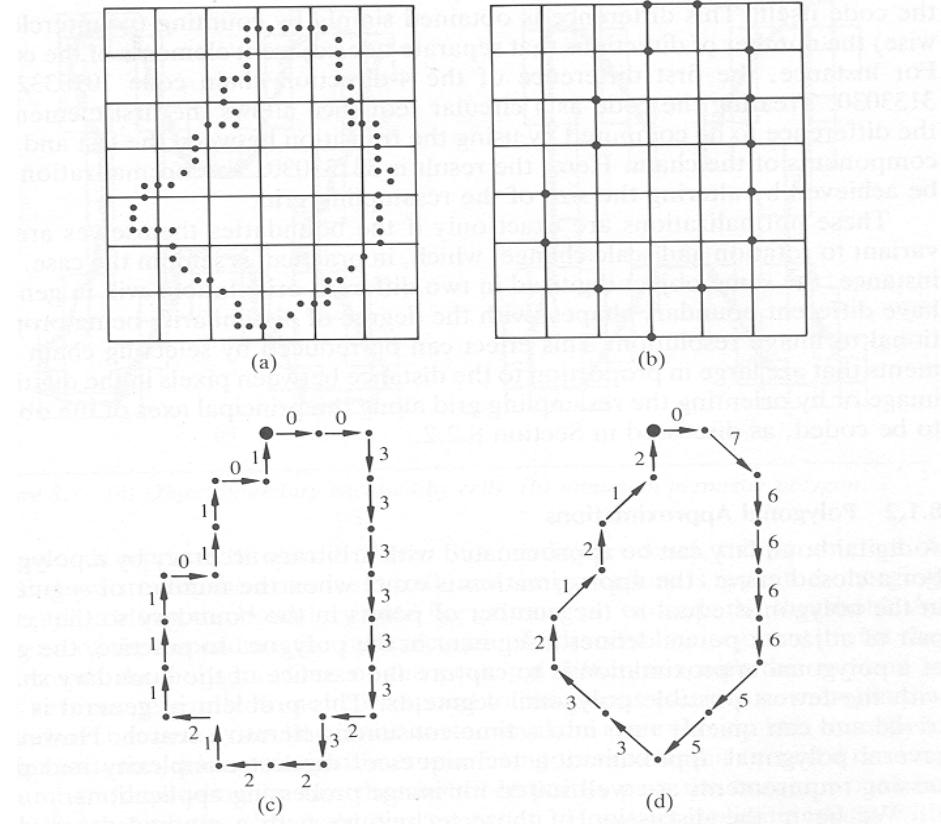


## Chain Code Representation

- A chain code could be made by following the pixels around a boundary in, say, a clockwise direction
- The main problem with this method is that the chain codes are too long and the representation is much too sensitive to noise on the boundary
- Usually the boundary is resampled onto a larger grid spacing to avoid this problem
- The chain code will depend upon the starting point, but we can circumvent this problem by treating the code a circular sequence and rotating the code to form an integer of maximum magnitude
- We can make the code rotationally invariant by using relative angles rather than absolute
- Size normalization can be performed by scaling the resampling grid.



# Chain Code Example



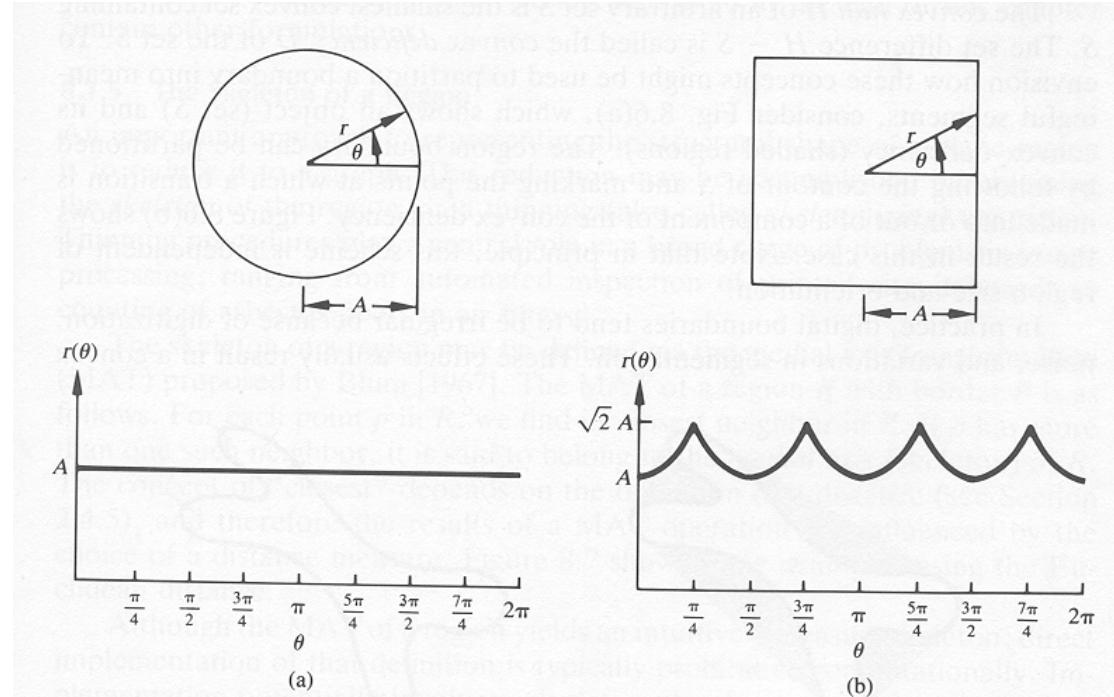


## Signatures

- A signature is a 1D functional representation of a boundary and may be generated in a number of ways
- One of the simplest is to plot the distance from the centroid to the boundary as a function of angle
- Signatures generated by this approach are invariant to translation but depend on rotation and scaling
- Normalization to rotation can be achieved by selecting the point furthest from the centroid as the starting point – if this point is unique
- Another way is to pick the point on the principle eigen-axis furthest from the centroid



## Example of Signature



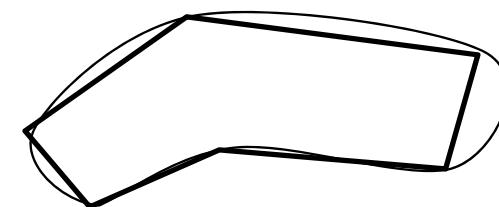
**Problems: Robustness, normalisation, representing concave shapes**

**Many variations on this technique**



## Interesting Variations

- To allow for a wider range of shapes (including concave), divide boundary into uniform intervals and record angle changes between line segments traversing boundary.
- Take Fourier Transform of coefficients to make representation rotationally and scale invariant
  - leads to called *Fourier Shape Descriptors* which can be quite powerful
  - shape can be decomposed into Fourier components with first second harmonics etc
  - Most reliable shape information is contained in the low frequency terms



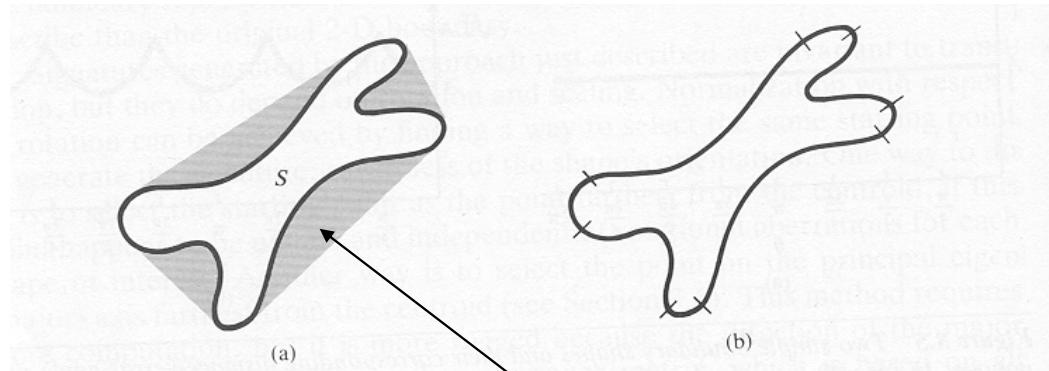


## Convex Hull

- Decomposing a boundary into segments can often be useful
- Decomposition reduces the boundary's complexity and thus simplifies the description process
- This technique is particularly attractive when the boundary contains one or more significant concavities that carry shape information
- The *convex hull*  $H$  of an arbitrary set  $S$  is the smallest convex set containing  $S$ .
- The set difference  $H-S$  is called the *convex deficiency*  $D$  of the set  $S$



## Example of Convex Hull



**convex deficiency**

**The region boundary can be partitioned by finding the contour of  $S$  and marking the points at which the transition is made into or out of a component of the convex deficiency**

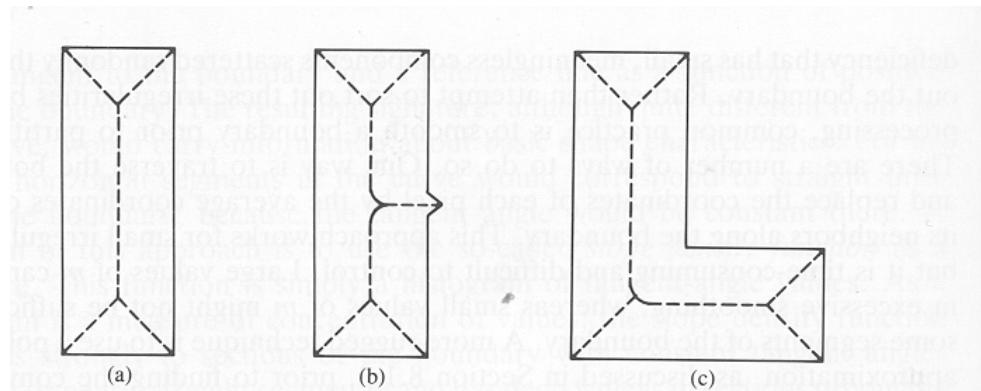


## Skeleton of a Region

- An important technique for representing the structural shape of a plane region is to reduce it to a graph
- This may be accomplished by obtaining the skeleton of the region via a thinning (skeletonization) algorithm
- The skeleton may be defined by the medial axis transformation defined by Blum(1967).
- The MAT of a region  $R$  with border  $B$  is as follows
  - For each point  $p$  in  $R$ , we find its closest neighbour in  $B$
  - if  $p$  has more than one such neighbour, it is said to belong to the medial axis of  $R$
  - The definition of closest depends on the measure of distance. Frequently we use Euclidean distance.



## Medial Axes of Three Regions



**While the result is pleasing, the computation of MAT is expensive**

**Many algorithms have been proposed to approximate MAT at reduced computational load**



## Thinning Algorithm

- Approximate method successively deletes edge points subject to the constraint that it
  - does not remove end points
  - does not break connectedness
  - does not cause excessive erosion of the region
- Method consists of successive passes of two basic steps applied to the contour points of a region, where a contour point is a point having at least one 8-neighbour valued 0 (background)



# Algorithm

- Step1
  - Flag a contour point  $p$  for deletion if the following conditions are satisfied
    - a. number of 8-neighbours is between 2 and 6
    - b.  $S(p_1)=1$
    - c.  $p_2 \cdot p_4 \cdot p_6 = 0$
    - d.  $p_4 \cdot p_6 \cdot p_8 = 0$

where  $S(p_1)$  is the number of 0-1 transitions in the sequence  $p_2, p_3, \dots, p_9$

  - delete flagged points
- Step2
  - repeat Step 1 but change the last two conditions to
    - c.  $p_2 \cdot p_6 \cdot p_8 = 0$
    - d.  $p_2 \cdot p_4 \cdot p_8 = 0$
- Repeat until no more points to delete

p9	p2	p3
p8	p1	p4
p7	p6	p5



## Comments

- Condition a is violated when contour point p has only 1 or 7 8-neighbours valued 1
  - having only one implies that p is an end-point of a skeleton and should not be deleted
  - deleting p with 7 neighbours would cause erosion into the region
- Condition b is violated when it is applied to points on a stroke 1 pixel wide
  - hence this conditions prevents disconnection of segments of a skeleton



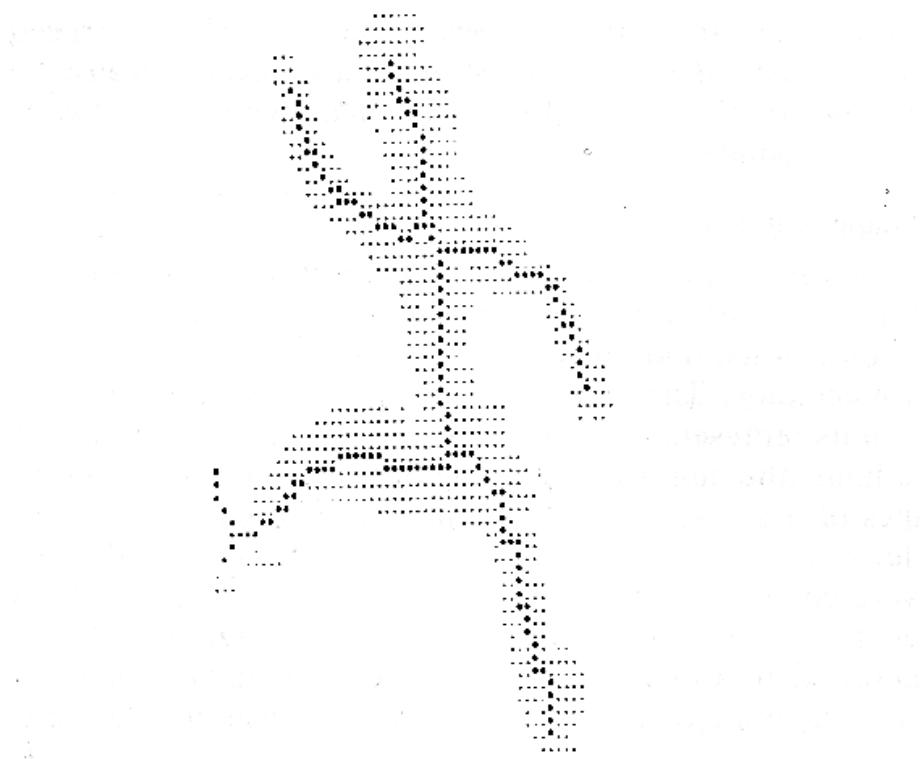
## More Comments

- Conditions c and d are satisfied simultaneously by the minimum set of values
  - $(p4=0 \text{ or } p6=0) \text{ or } (p2=0 \text{ and } p8=0)$
  - a point which satisfies these conditions is an east or south boundary point or a northwest corner point in the boundary
  - in either case p is not part of the skeleton and should be removed
  - similar arguments apply for the conditions of Step2

p9	p2	p3
p8	p1	p4
p7	p6	p5



## Skeletonization Example





## Morphology

- The word *Morphology* denotes a branch of biology which deals with form and structure of animals and plants
- Mathematical Morphology is a tool for extracting image components that are useful in the representation and description of region shape
- Morphological techniques can be used to find boundaries, skeletons, convex hulls and also for filtering, thinning and pruning.
- Morphological techniques are well-developed for binary images, but many methods can be successfully extended to grayscale.



# Dilation and Erosion

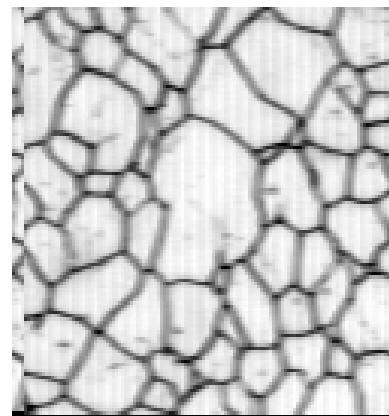
- The basic operations of dilation and erosion form the basis of many more sophisticated techniques

**Matlab**

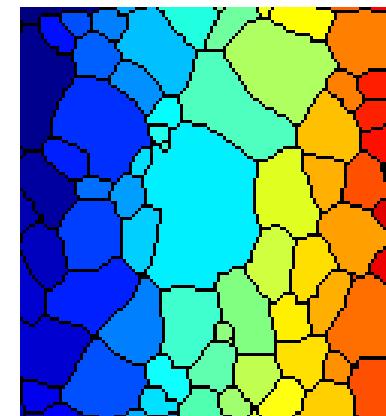
**Morphology**

**Demo**

**Original image**



**Labelled steel grains**





## Definitions

- Let  $A$  and  $B$  be sets in  $\mathbb{Z}^2$ , with components  $a=(a_1,a_2)$  and  $b =(b_1,b_2)$
- The *translation* of  $A$  by  $x=(x_1,x_2)$ , denoted  $(A)_x$  is defined by

$$(A)_x = \{c \mid c = a + x, \text{for } a \in A\}$$

- The *reflection* of  $B$ , denoted  $\hat{B}$ , is defined by

$$\hat{B} = \{x \mid x = -b, \text{for } b \in B\}$$



## More Definitions

- The *complement* of set A is

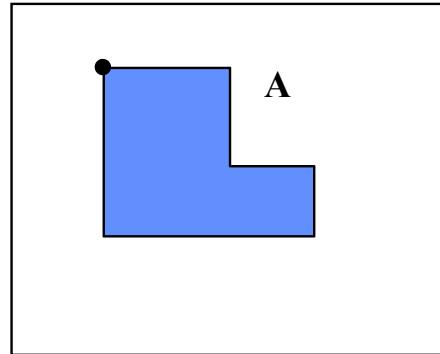
$$A^c = \{x \mid x \notin A\}$$

- The *difference* of sets A and B, denoted A-B, is defined by

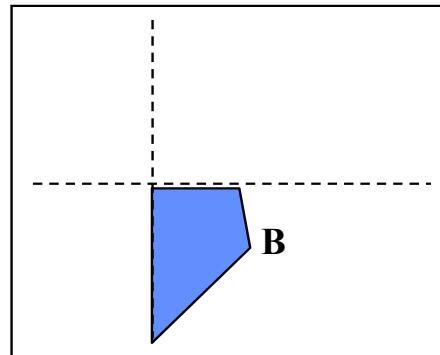
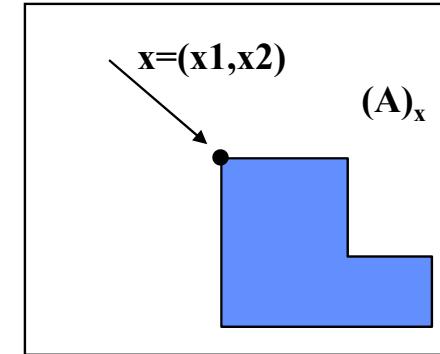
$$A - B = \{x \mid x \in A, x \notin B\} = A \cap B^c$$



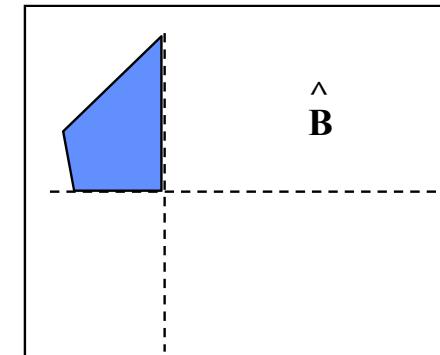
## Examples



Translation

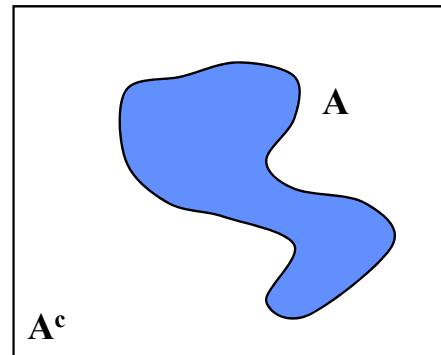


Reflection

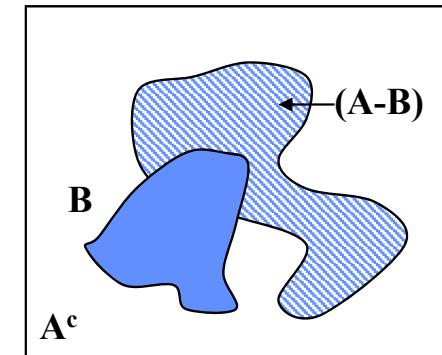




## More Examples



**Difference**



$$A - B = \{x \mid x \in A, x \notin B\} = A \cap B^c$$



## Dilation

- with  $A$  and  $B$  as sets in  $\mathbb{Z}^2$  and  $\emptyset$  denoting the empty set, the dilation of  $A$  by  $B$ , denoted  $A \oplus B$  is defined by

$$A \oplus B = \{x \mid (\hat{B})_x \cap A \neq \emptyset\}$$

↑  
reflected  
translated

- Thus the dilation process consists of obtaining the reflection of  $\hat{B}$  about its origin and then shifting this reflection by  $x$ . The dilation of  $A$  by  $B$  then is the set of all  $x$  displacements such that  $B$  and  $A$  overlap by at least one nonzero element.

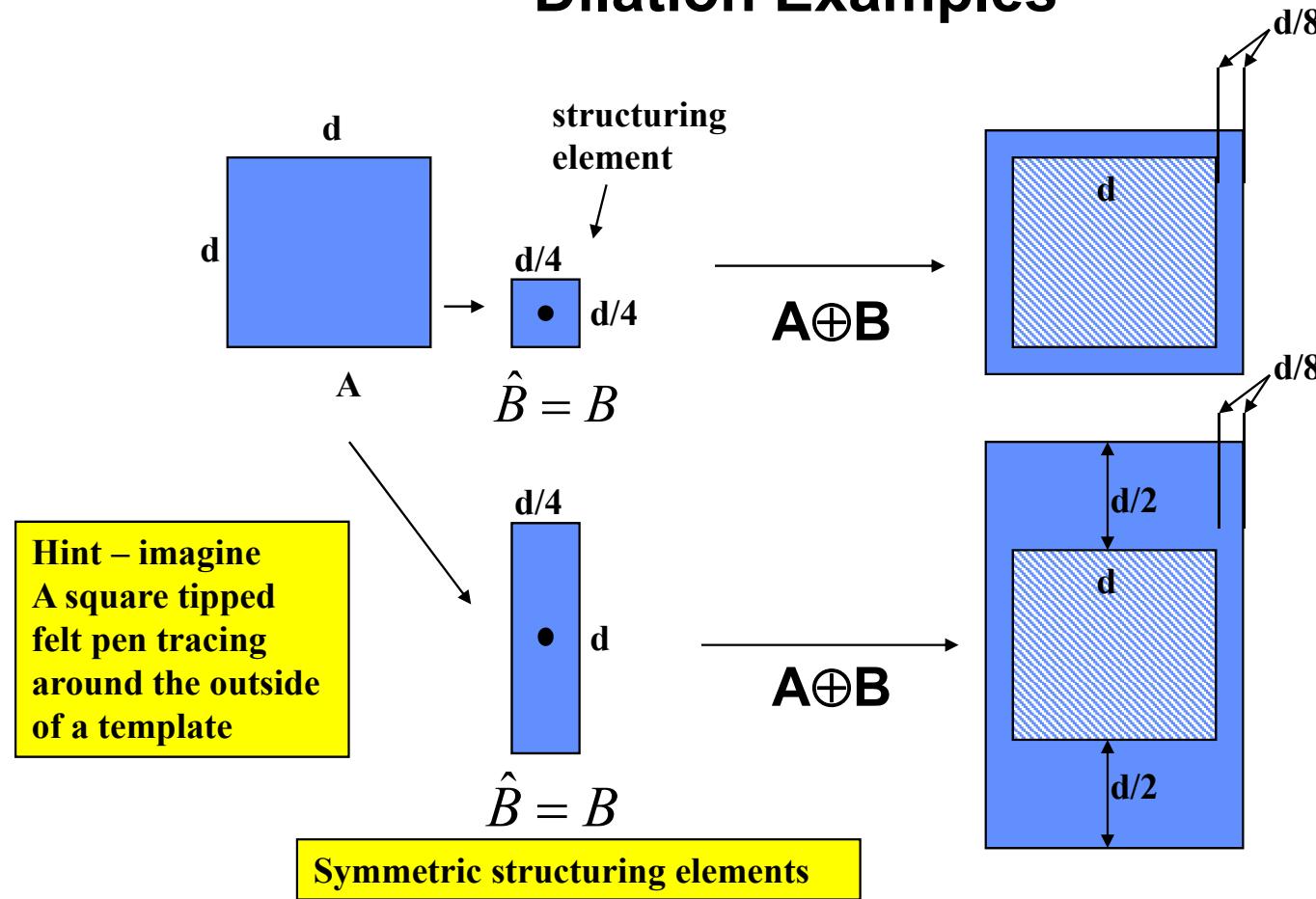


## Dilation (cont)

- Set B is commonly referred to as the *structuring element* in dilation , as well as in other morphological operations.
- Often the set B is viewed as a convolution mask
- The basic process of “flipping” B about its origin and then successively displacing it so that it slides over set (image) A is analogous to the convolution process.



## Dilation Examples





## Erosion

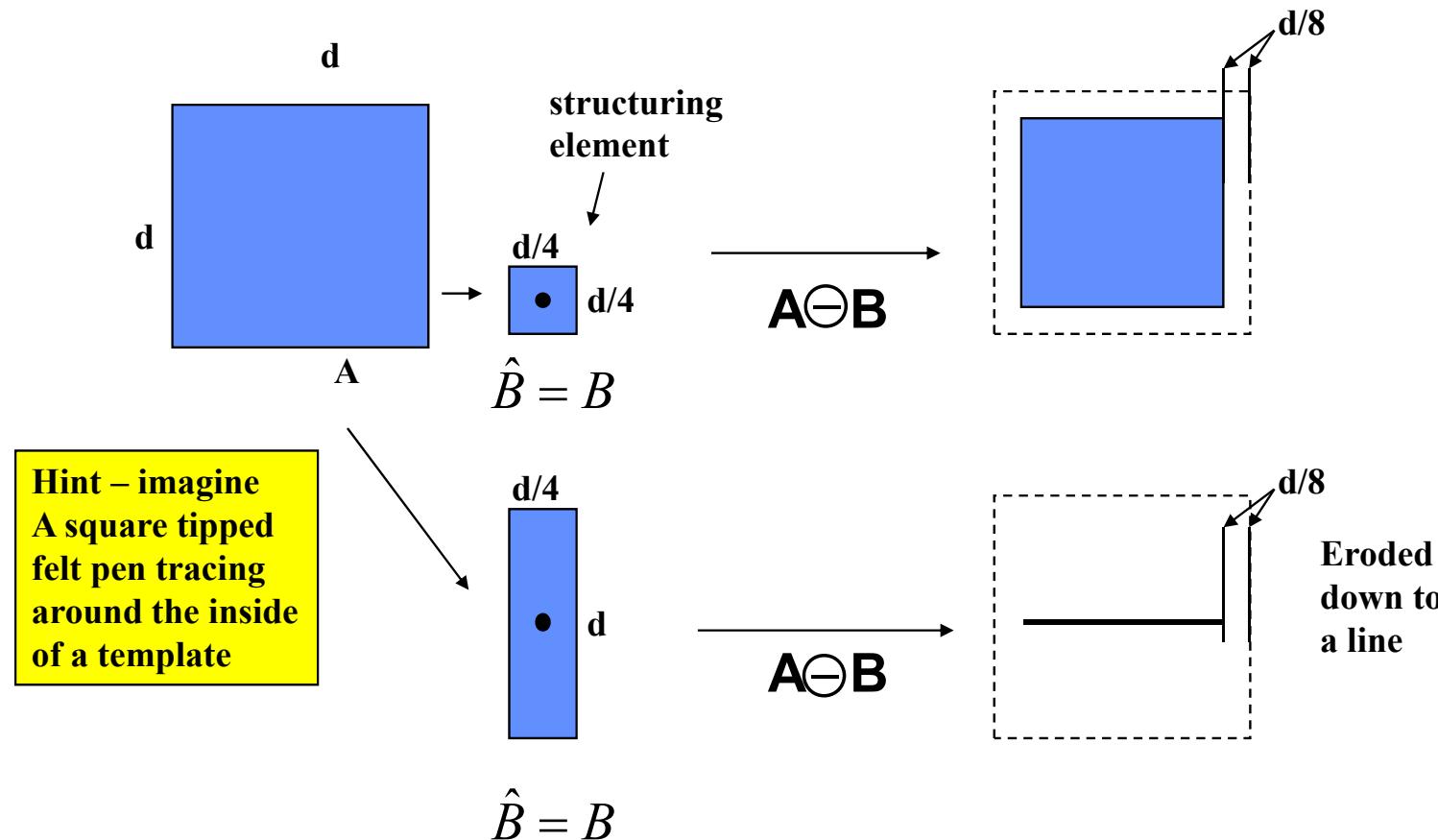
- For sets A and B in  $\mathbb{Z}^2$  the erosion of A by B denoted  $A \ominus B$  is defined by

$$A \ominus B = \{x \mid (\hat{B})_x \subseteq A\}$$

- In other words, the erosion of A by B is the set of all points x such that B, translated by x, is completely contained in A



## Erosion Examples





## Comment

- Dilation and erosion are duals of each other with respect to set complementation and reflection. That is

$$(A \ominus B)^c = A^c \oplus B$$

complement

erosion                                  dilation



## Opening and Closing

- Dilation expands an image and erosion shrinks it
- Opening generally smoothes the contour of an image, breaks narrow isthmuses, and eliminates thin protrusions
- Closing also tends to smooth sections of contours, but generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.
- The *Opening* of set A by structuring element B is

$$A \circ B = (A \ominus B) \oplus B$$

- The *Closing* of set A by structuring element B is

$$A \bullet B = (A \oplus B) \ominus B$$

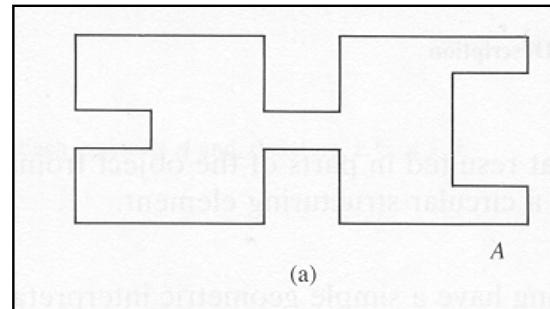


## Comments

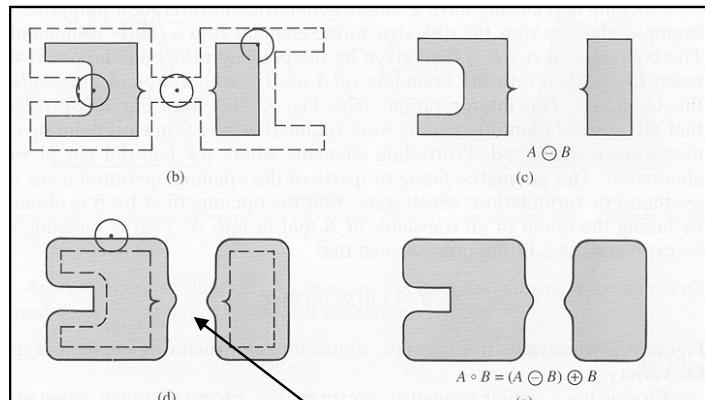
- In other words
  - Opening is simply the erosion of A by B, followed by the dilation by B
  - Closing is simply the dilation of A by B, followed by the erosion by B
- The operations are extensively used to preprocess images after thresholding operations to “clean up” the images prior to further processing



## Examples of Opening and Closing

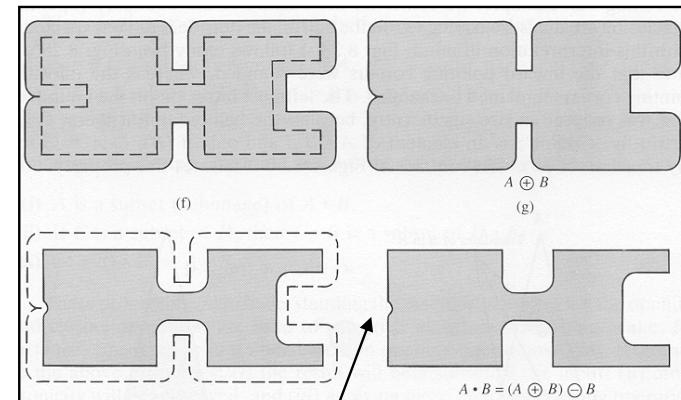


### Opening



Elimination of bridge

### Closing



Elimination of notch



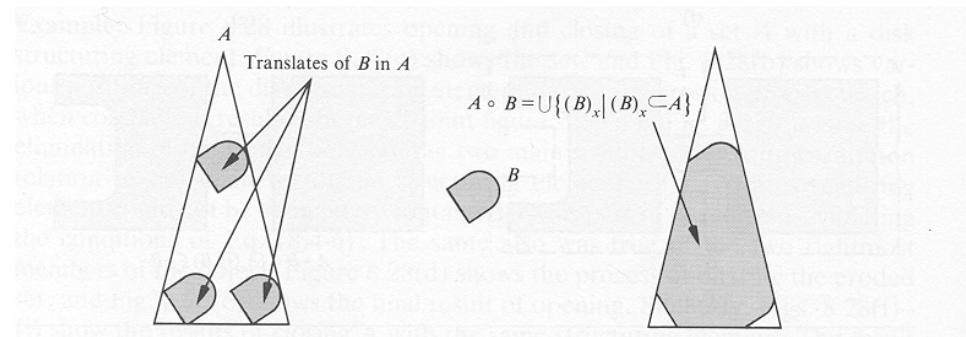
## Interpretation

- Opening and closing have simple geometrical interpretations, if we view a disk structuring element as a rolling ball
  - The boundary of the opening is the points on the boundary of the ball (structuring element) that are closest to the boundary of A as B is rolled on the inside of this boundary
  - The boundary of the closing is the points on the boundary of the ball (structuring element) that are closest to the boundary of A as B is rolled on the outside of this boundary

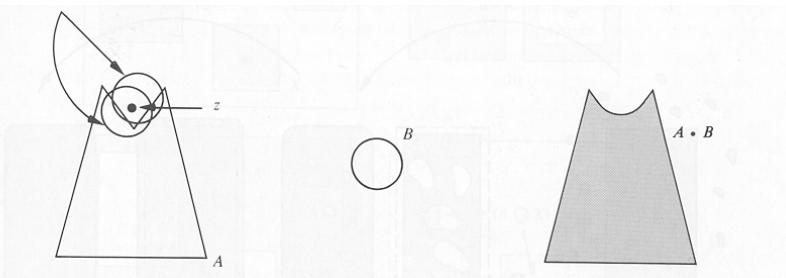
**Note: structuring elements  
do not roll (rotate) in  
practice**



## Illustration



“Fitting” Characterization of Opening



“Fitting” Characterization of Closing



## Properties of Openings

- Opening operation satisfies:
  1.  $A \circ B$  is a subset (subimage) of A
  2. If C is a subset of D, then  $C \circ B$  is a subset of  $D \circ B$
  3.  $(A \circ B) \circ B = A \circ B$
- In other words:
  1. The opening is a subset of the input
  2. Monotonicity is preserved
  3. Applying more than one opening has no effect on the result



## Properties of Closings

- Similarly, closing operation satisfies:
  - 1.A is a subset (subimage) of  $A \bullet B$
  - 2.If  $C$  is a subset of  $D$ , then  $C \bullet B$  is a subset of  $D \bullet B$
  3. $(A \bullet B) \bullet B = A \bullet B$
- In other words:
  - 1.The input is a subset of the closing
  - 2.Monotonicity is preserved
  - 3.Applying more than one opening has no effect on the result



## Example of Opening/Closing

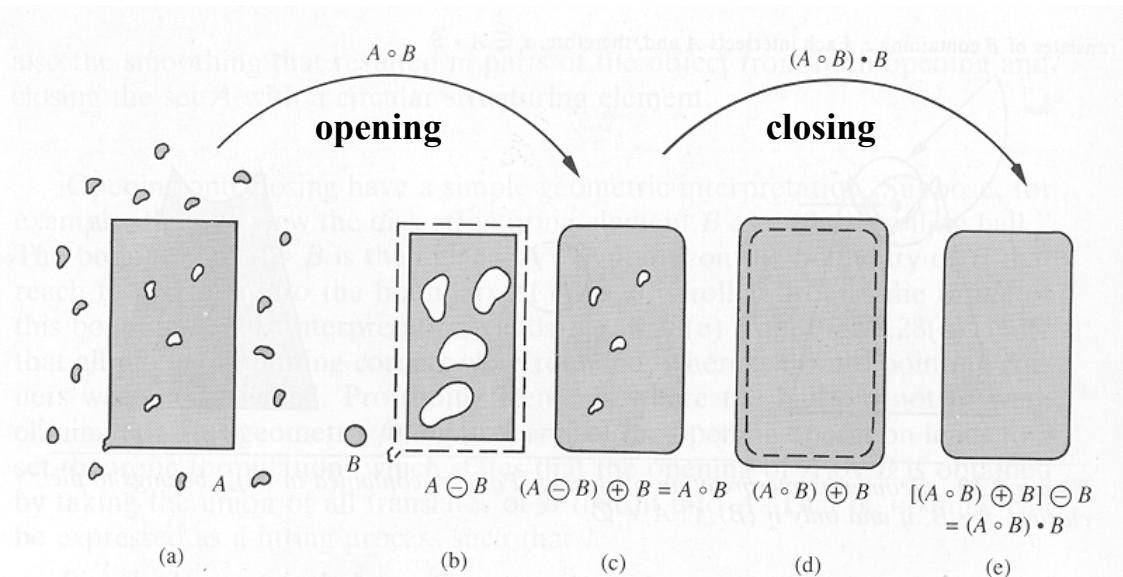


Image consists of a thresholded object with noise both within and without the object boundary. Opening using a structuring element larger than the noise artefacts removes noise outside the object. Closing fills in the voids.



# Hit or Miss Transform

## Morphological Template Matching

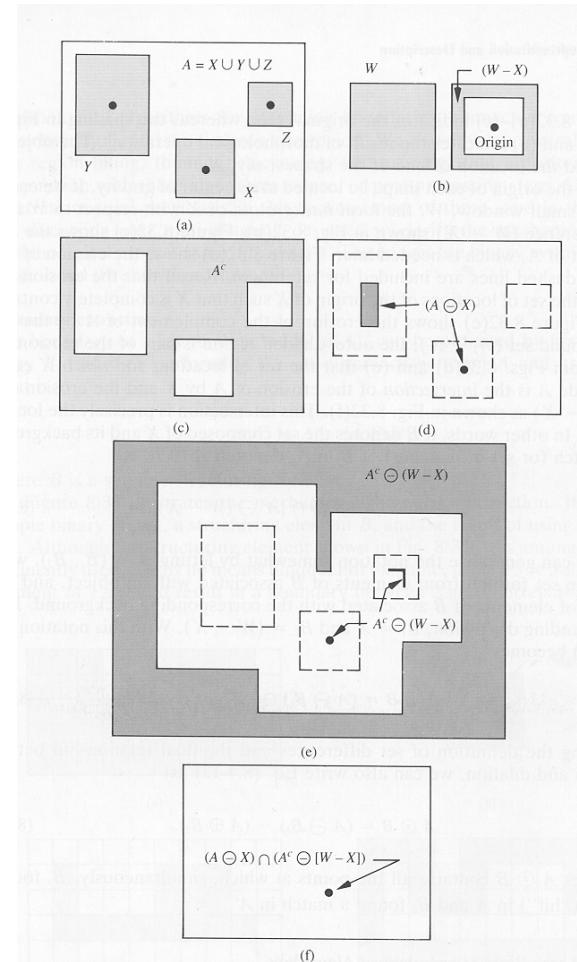
- This can be used as a basic tool for shape detection
- As an example we will use a set A consisting of three shapes: X, Y, and Z (see next slide)
- If we enclose X by a small window, W, the local background of X wrt W is the set difference ( $W-X$ )
- Recall that the erosion of Z by X is the set of locations of the origin of X such that X is completely contained in A
- Now the set of locations for which X exactly fits inside A is the intersection of the erosion of A by X and the erosion of  $A^c$  by ( $W-X$ )
- This intersection is precisely the location sought.



## Hit or Miss Example

We can generalize the notation somewhat by letting  $B = (B_1, B_2)$  where  $B_1$  represents the object and  $B_2$  represents the local background.

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

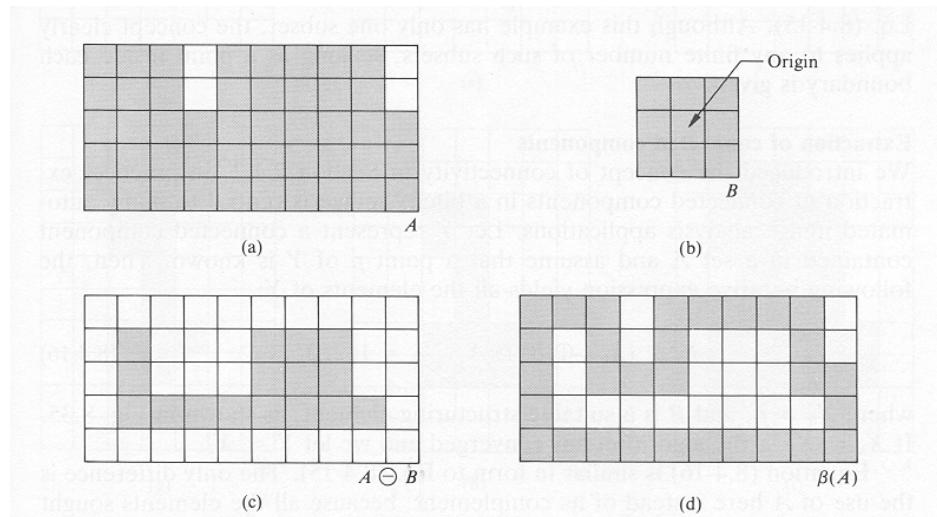




# Practical Uses

- **Boundary Extraction**

- Erode A by B and then perform set difference with A
    - $\text{boundary}(A) = A - (A \ominus B)$

 $\ominus$ 



## Region Filling

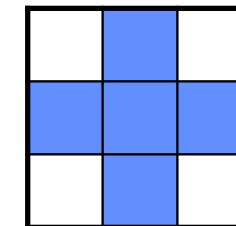
- Conditional Dilation

- Given the 8-connected boundary of a region, select a point p inside the boundary and conditionally dilate with structuring element B until region is filled using

$$X_k = (X_{k-1} \oplus B) \cap A^c, \quad k=1,2,3,\dots$$

read as AND

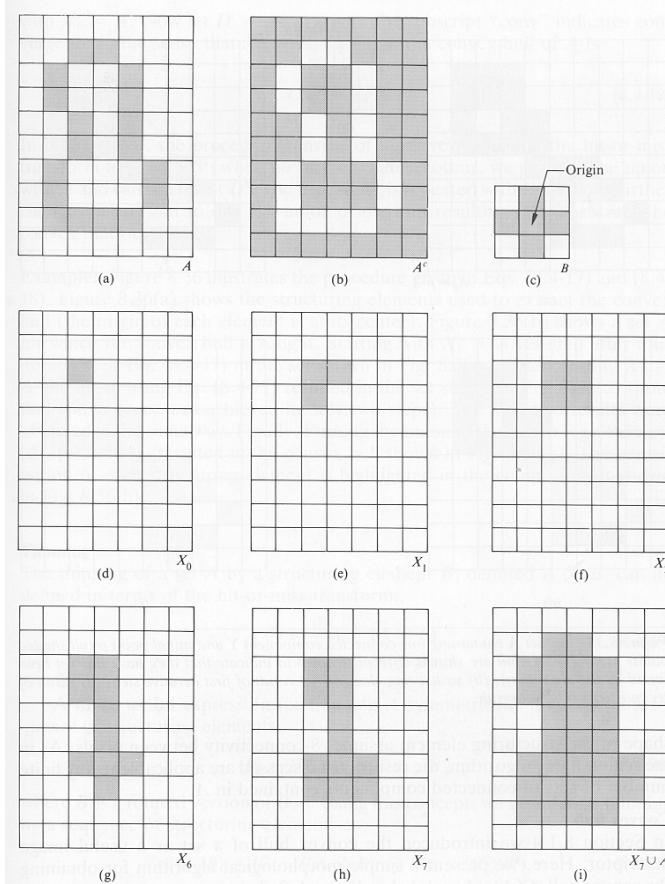
- Continue until  $X_k = X_{k-1}$
  - B must be of the form





## Example

**Morphological flood filling using conditional dilation**



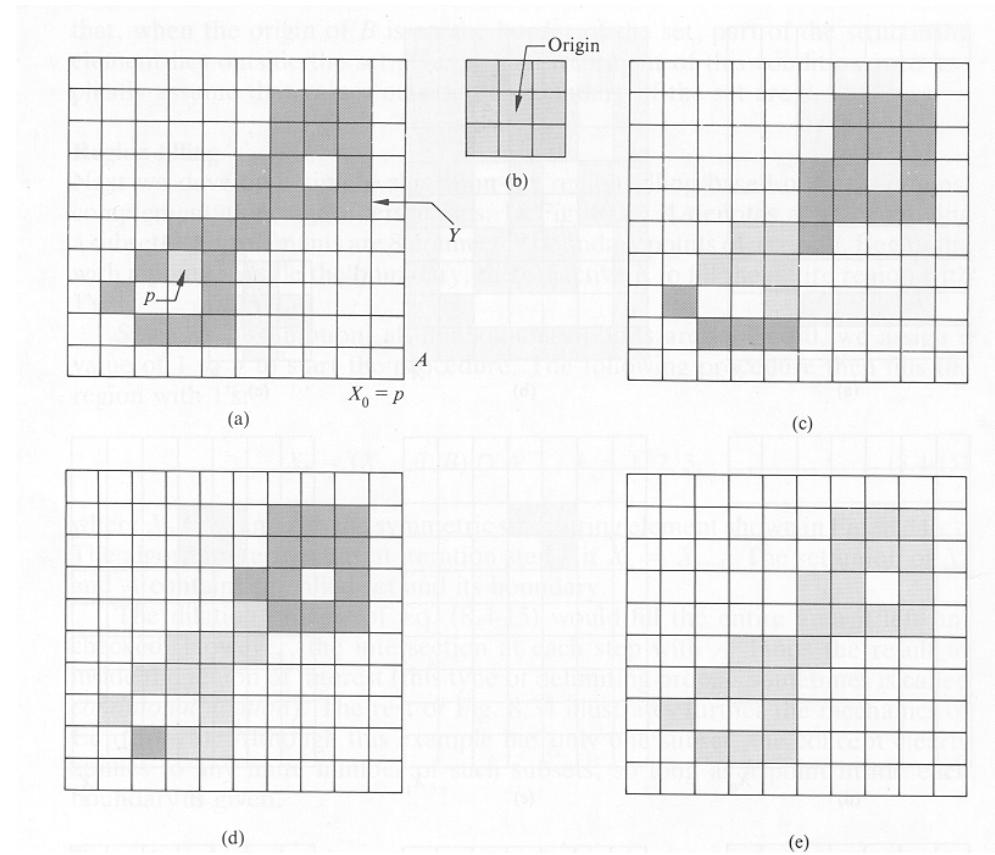


## Extraction of Connected Components

- This process is central to many image processing applications
- Let  $Y$  represent a connected component and assume that a point  $p$  of  $Y$  is known
- Setting  $X_0$  to  $p$ , this conditional dilation will yield all points in  $Y$ 
  - $X_k = (X_{k-1} \oplus B) \cap A$ ,  $k=1,2,3,\dots$
  - Continue until  $X_k = X_{k-1}$
  - Structuring element  $B$  should be say a  $3 \times 3$  pixel group



## Example





## Convex Hull

- Let  $B^i$ ,  $i=1,2,3,4$ , represent 4 structuring elements.
- The procedure consists of implementing the equation

$$X_k^i = (X \circledast B^i) \cup A \quad i = 1,2,3,4 \text{ and } k = 1,2,3,4.$$

with  $X_0^i = A$

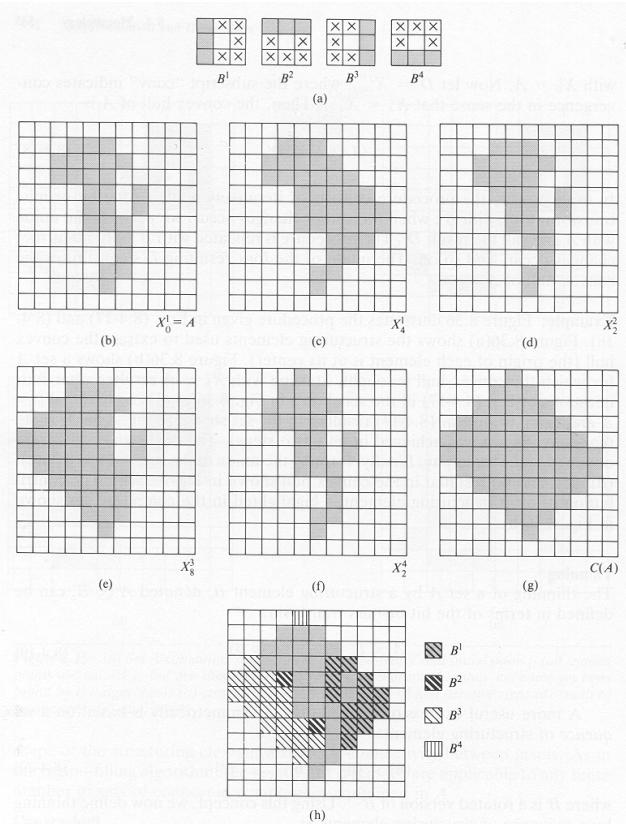
- When each of these has converged, the union is the convex hull.



## Example

### Convex Hull of Set A

Note: X means “don’t care”





## Thinning

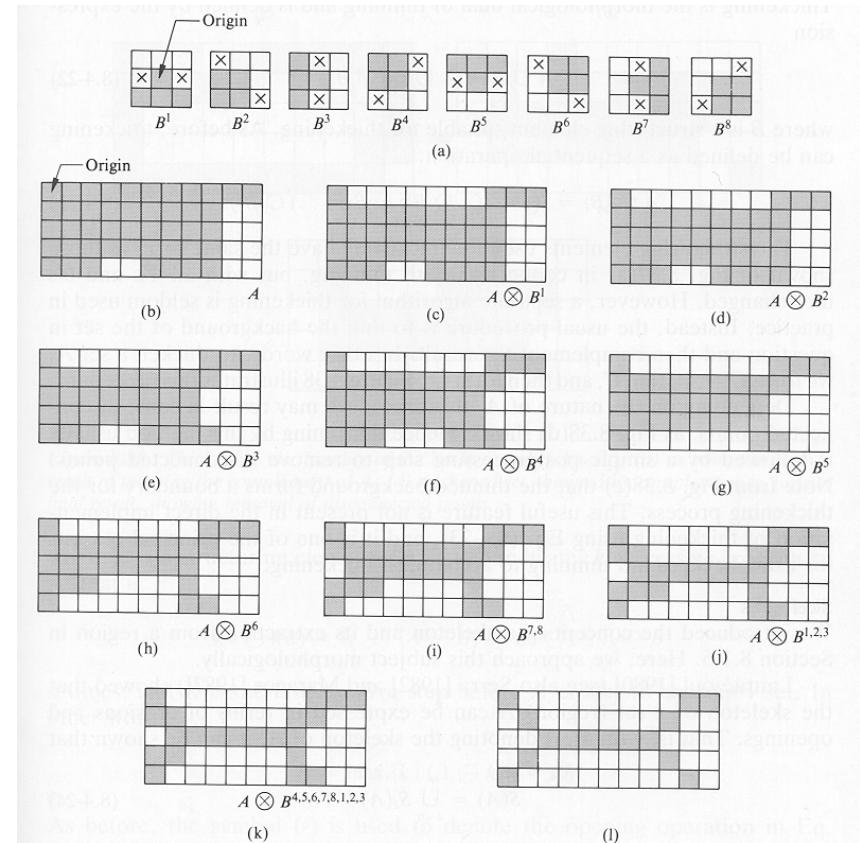
- The thinning of a set  $A$  by structuring element  $B$ , denoted  $A \otimes B$ , can be defined in terms of the hit-or-miss transform

$$\begin{aligned} A \otimes B &= A - (A \oplus B) \\ &= A \cap (A \oplus B)^c \end{aligned}$$

- The usual process is to thin  $A$  using a sequence of structuring elements  $B^1, \dots, B^n$
- In other words,  $A$  is thinned by successive passes of structuring elements  $B^1, B^2, \dots$
- The entire process is repeated until no further change occurs



# Thinning Example





## Other Morphological Operations

- We also have definitions for the following operations
  - Thickening
    - make lines thicker
  - Skeletonization
    - extract morphological skeleton
  - Pruning
    - extract parasitic components after skeletonization
- See Gonzalez and Woods, “Digital Image Processing,” pp 518-545 for details



# Application

## Postcode processing

Mr. John P. Rose  
1684 M. Meadow PL  
Knoxville, TN 37919-2100

(a)

adaw PL  
1 37919-2100

(b)

dilation  
to bridge  
breaks in  
numbers

Erosion to separate  
numbers

adaw PL  
1 [37919-2100]

(c)

adaw PL  
1 [37919-2100]

(d)

skeleton

adaw PL  
1 [37919-2100]

(e)

adaw PL  
1 [37919-2100]

(f)

pruning



## Grayscale Morphology

- Many binary morphological operations are simply extended to grayscale images
- Here we regard the grayscale image as a surface that is eroded and dilated
- Often it is simpler to illustrate the process with 1-D functions, since the extension to 2-D is trivial.



## Grayscale Dilation

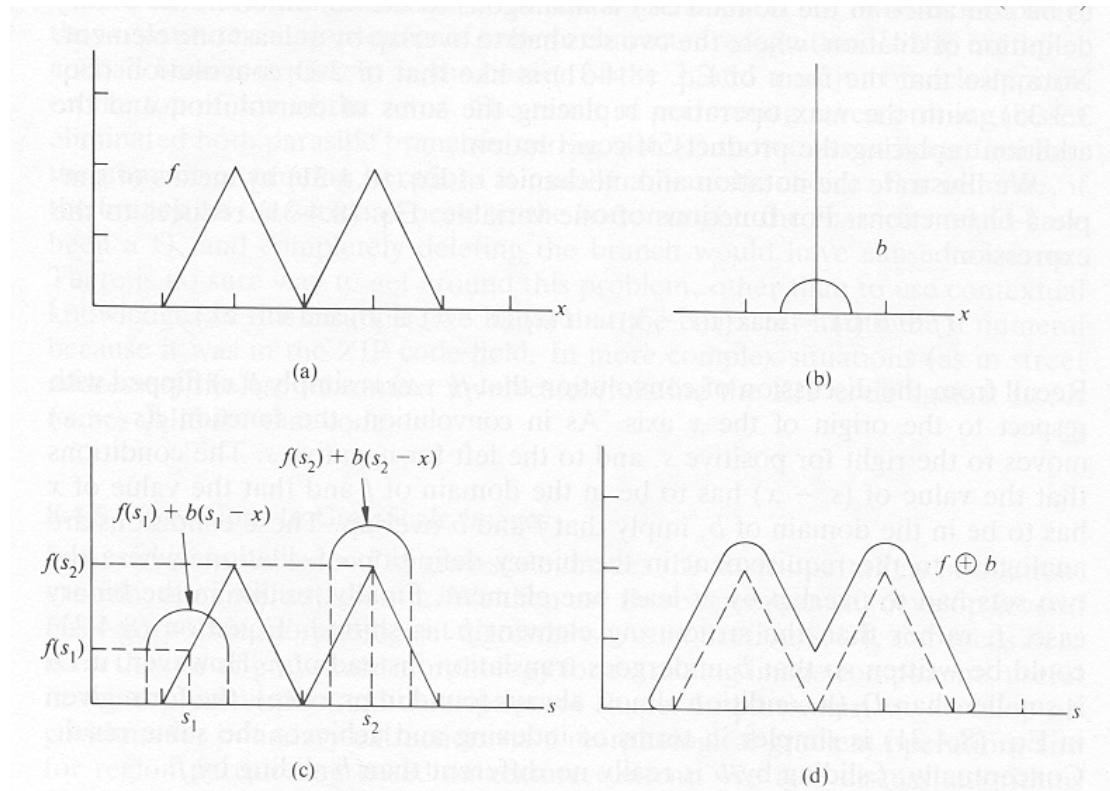
- Grayscale dilation of  $f$  by  $b$ , denoted  $f \oplus b$ , is defined by

$$(f \oplus b)(s, t) = \max \{f(s - x, t - y) + b(x, y) \mid (s - x), (t - y) \in D_f; (x, y) \in D_b\}$$

- In other words, we find the maximum of the function  $f+b$  in a neighborhood defined by the structuring element  $b$  as we slide  $b$  over  $f$ .
- This is illustrated graphically on the next slide for a 1-D function



## Dilation Example





## Grayscale Erosion

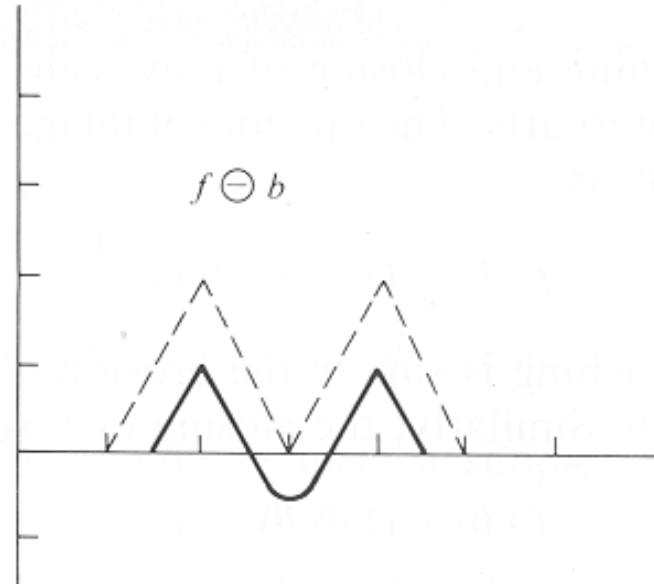
- Similarly, Grayscale erosion of  $f$  by  $b$ , denoted  $f \ominus b$ , is defined by

$$(f \ominus b)(s, t) = \min\{f(s - x, t - y) + b(x, y) \mid (s - x), (t - y) \in D_f; (x, y) \in D_b\}$$

- In other words, we find the minimum of the function  $f+b$  in a neighbourhood defined by the structuring element  $b$  as we slide  $b$  over  $f$ .



## Erosion Example



Same example as before



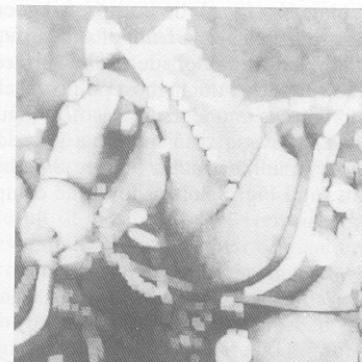
## Image Example

Original



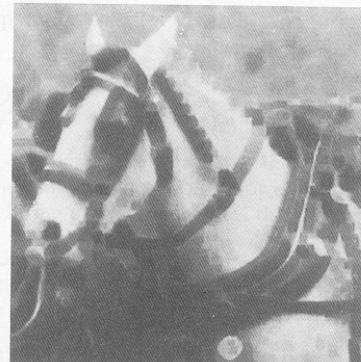
(a)

Dilated



(b)

Eroded



(c)



## Grayscale Opening and Closing

- The expression for grayscale opening and closing is the same as for binary
- The expression for opening is

$$f \circ b = (f \ominus b) \oplus b$$

which is simply erosion followed by dilation

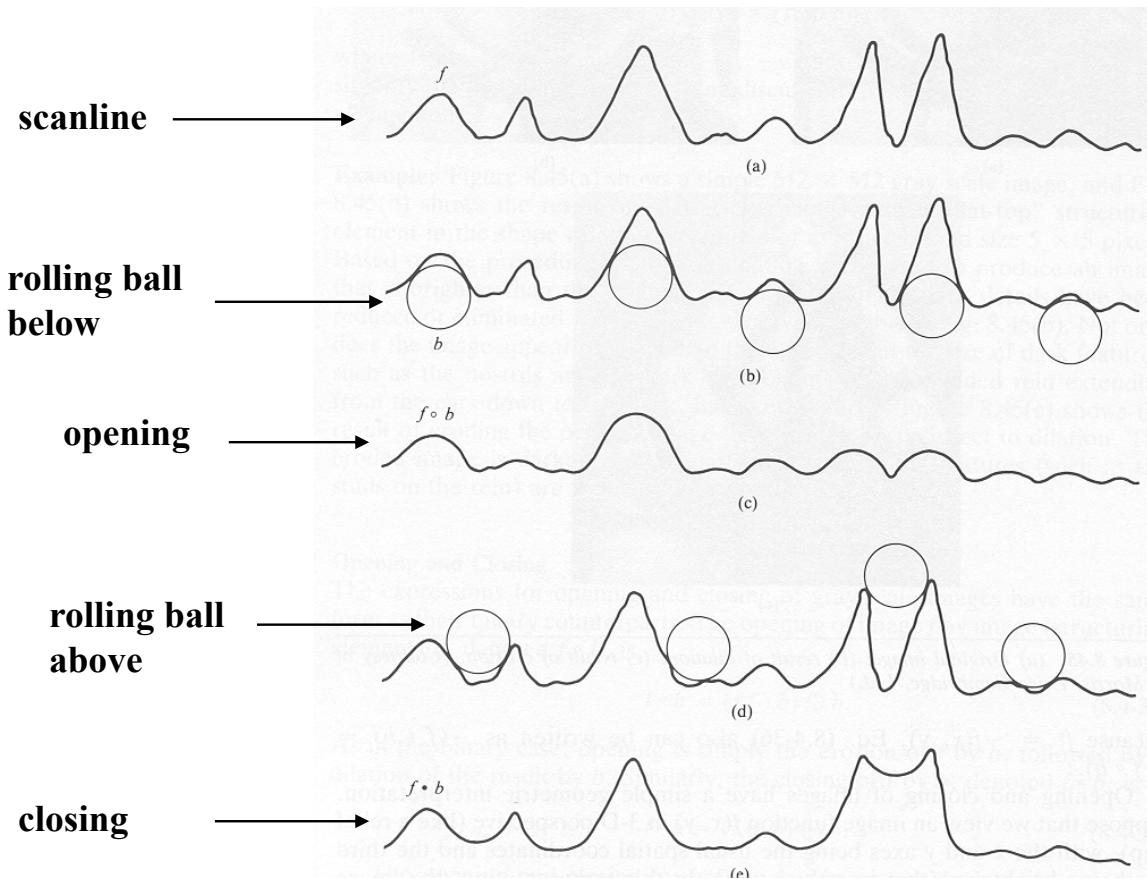
- The expression for closing is

$$f \bullet b = (f \oplus b) \ominus b$$

which is simply dilation followed by erosion



## Geometric Interpretation



**Rolling  
Ball  
Interpretation**



## Comments

- Openings are used to remove small light details, while leaving the overall gray levels and larger bright features relatively undisturbed
- Closing is generally used to remove dark details from an image while leaving bright features relatively undisturbed



## Applications

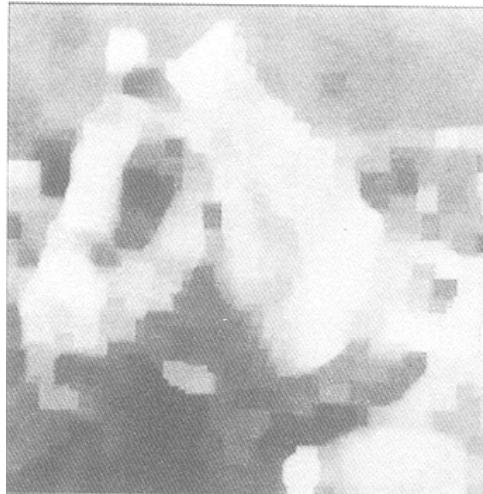
- Morphological smoothing
  - Opening followed by a closing
  - removes or attenuates both bright and dark artifacts or noise
- Morphological gradient
  - This is the difference between dilation and erosion

$$g = (f \oplus b) - (f \ominus b)$$

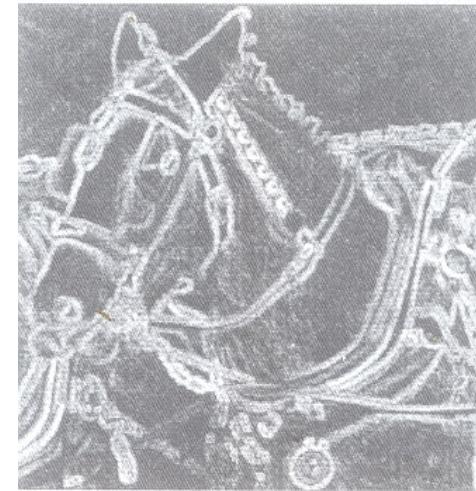
- highlights sharp gray level transitions in the image
- depends less on edge direction than Sobel etc



## Examples



**Morphological  
Smoothing**



**Morphological  
Gradient**



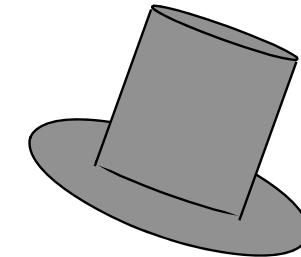
## Applications

- Top Hat Transformation

- The Morphological top-hat transformation is defined by

$$h = f - (f \circ b)$$

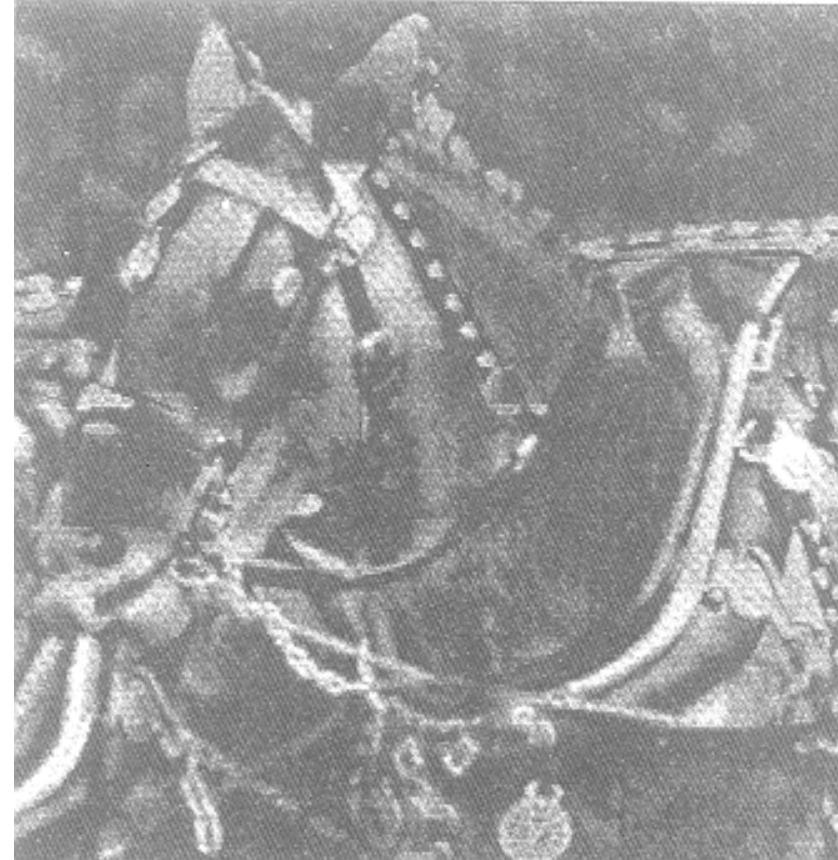
- That is the image  $f$  minus its opening with a structuring element  $b$ , which is often of the form of a “top-hat.” That is, a cylinder attached to a disk.
  - This transformation is useful for enhancing detail in the presence of shading
  - Also good in 1D for finding peaks that are, say, greater than a certain width and more than a certain depth (significant peaks)





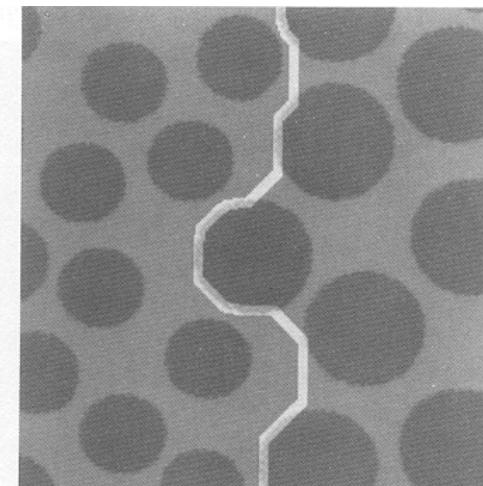
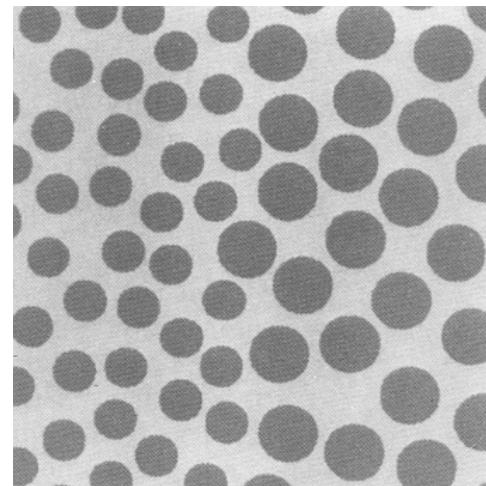
## Example

**Top hat transformation  
of image  
Note the enhanced  
detail**





## Texture Segmentation



**Method:** Close with successively larger structuring elements until small dots disappear.  
Open remaining image with large structuring element and then threshold to determine textural boundary.

Consider interpretation with rolling ball – process resembles coin sorter

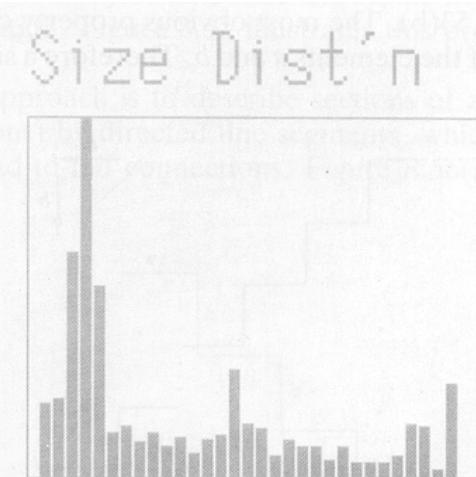
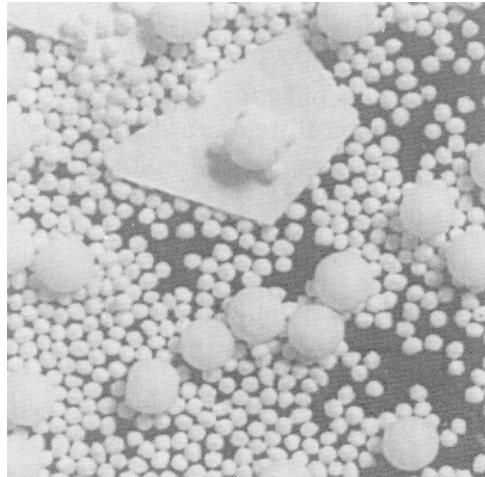


## Granulometry

- Granulometry is a field that deals with the size distribution of particles
- In the example image (next slide), there are light objects of three different sizes
- The objects are overlapping and are too cluttered to detect individual objects



## Example



### Method:

- opening operations with structuring elements of increasing size
- the difference between the original image and its opening is computed on each pass
- at the end of the process the differences are normalized and used to construct a histogram of particle size distribution

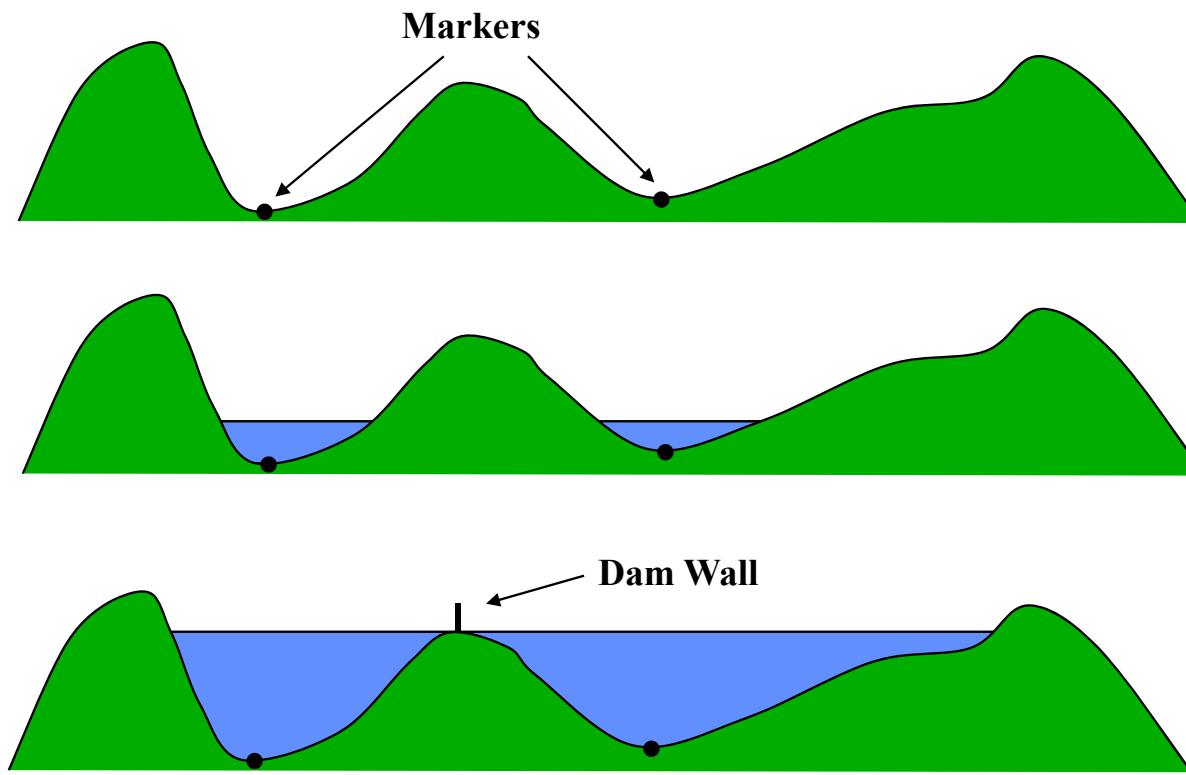


## Watersheds

- Create markers on image (usually image gradient) to indicate regions of interest
- Visualize image as a topological surface (mountains and valleys)
- Flood object with a deluge of rain
- When waters from different regions meet, construct dams
- Once surface is completely flooded, the dam walls are our watershed segmentation.



## Watershed Example





## Problems with Watersheds

- Usually operates on the gradient image rather than the image itself
  - gradient accentuates noise
- Often difficult to determine appropriate markers in many applications
  - may lead to poor segmentation
- Found to be unsuitable for cell image segmentation application



## Case Study: Problem Overview

Aim:

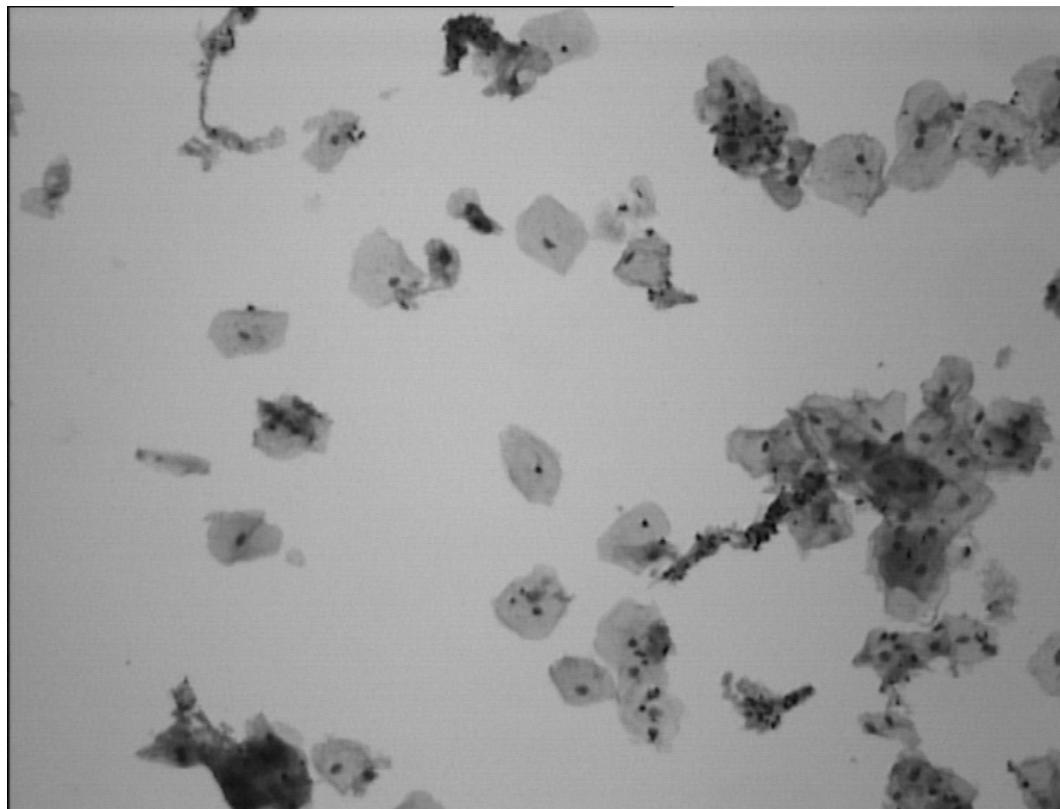
To develop a robust segmentation scheme, optimised for Pap. smear slides, for the application of an automatic cervical cancer pre-screener.

Present focus:

Evaluation of a water immersion algorithm developed by Jeacock and Bamford.



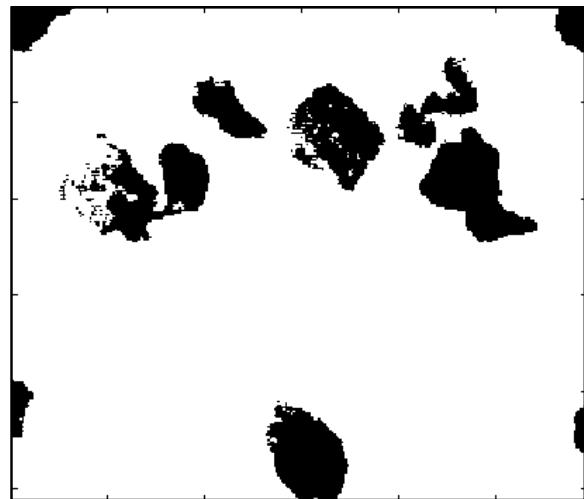
## Example Scene



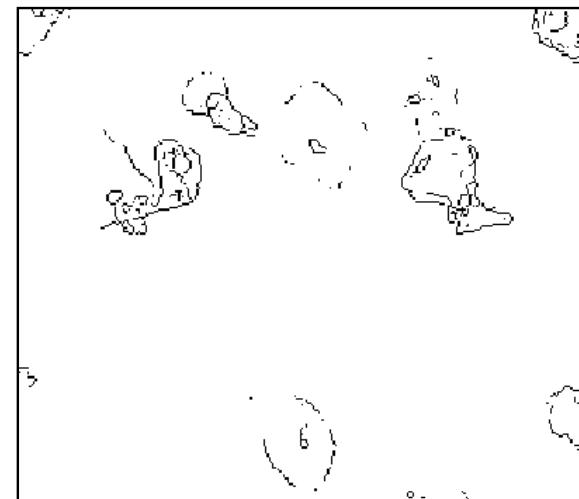


# Motivation

**Thresholding difficulties**



**Edge Detection Difficulties**



**Voids**

**Incomplete Boundaries**



## Algorithm Outline

- The algorithm consists of three stages:
  - Quadtree smoothing (Multiresolution technique, Pyramids)
  - Water immersion for lowest level classification
  - Boundary re-estimation

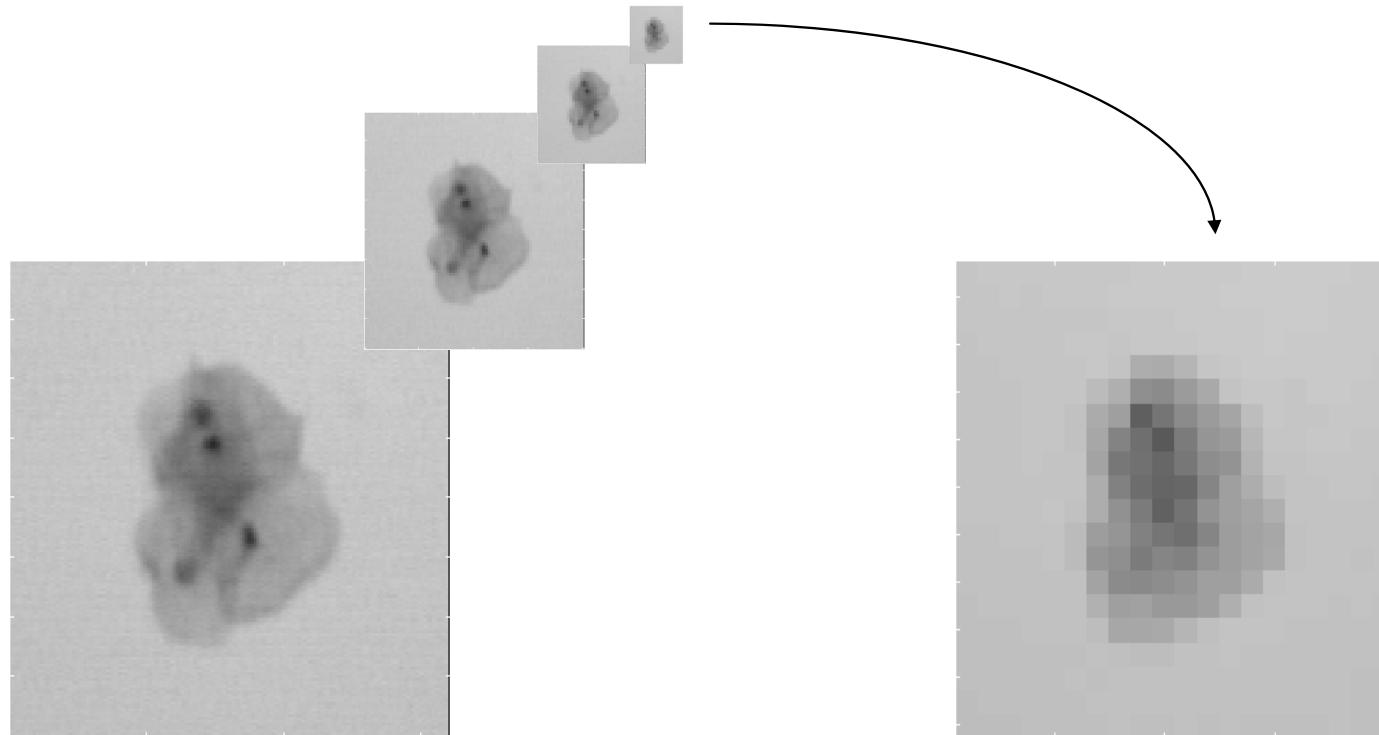


The image is first reduced by quadtree smoothing. This has the advantages of:

- Decreasing the number of pixels that require processing, increasing the speed of the algorithm.
- Causing background artefacts, such as blood cells, to become less significant compared to the objects of interest.
- Producing a series of images, each of a lower resolution than the previous.



## Quadtree Smoothing (Pyramid Processing)



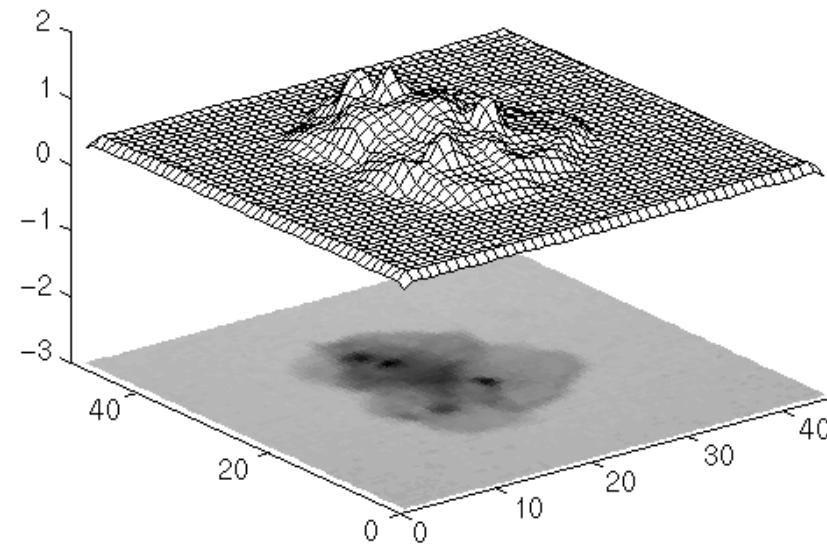


## Visualization of Water Immersion

- The gray-level image is treated as a topographical map.
- The surface can be imagined to be lowered into water, causing air to be trapped in the elevated areas.
- The air pockets mark the position of the cells.



## Gray-level image as a topographical map



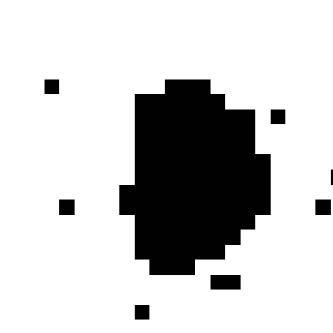
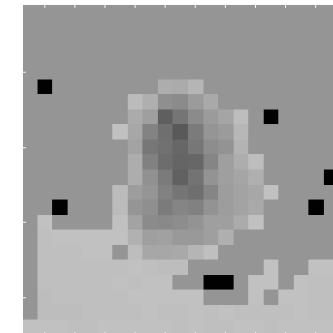
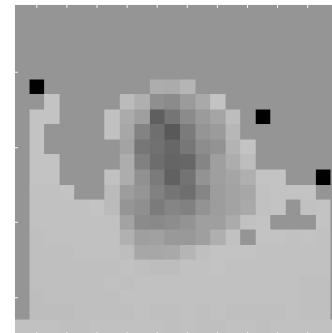
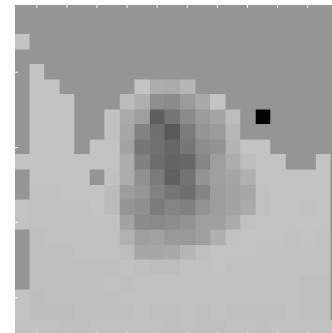
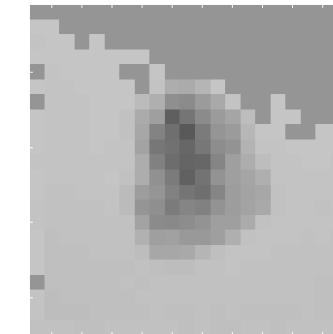
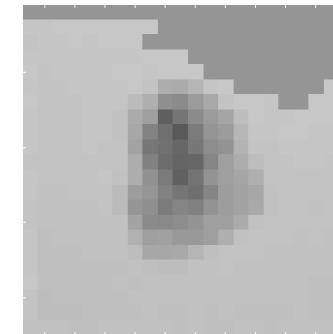
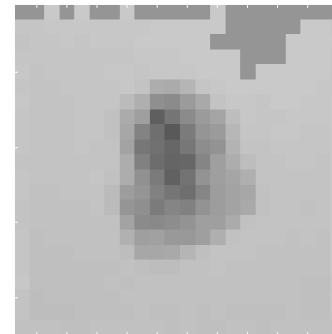
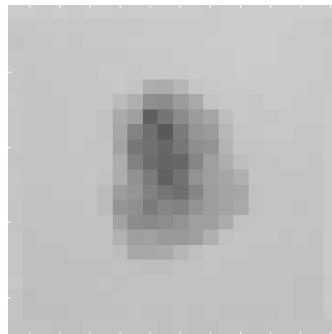


## Method

- The subsampled image is now flooded.
- The lighter pixels, being of lower elevation, become flooded first.
- Areas that become completely surrounded by water are considered to be possible cells and are marked (black in diagram).
- Occasionally, rogue pixels will be marked (as seen). These are possibly the remains of smoothed artifacts and can be removed later. Artifacts tend to be much smaller than cells.



## Water Immersion



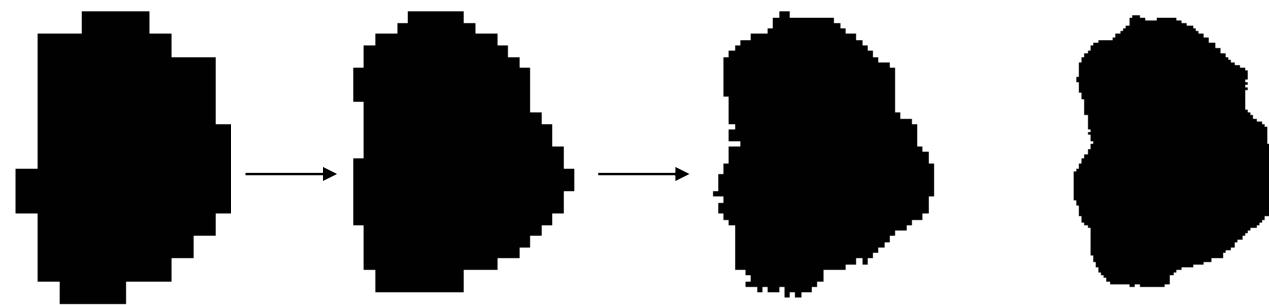


## Boundary Re-estimation

- The boundary now requires refining.
- The border pixels are selected for processing and are re-flooded as before at the next resolution (more detail).
- This is repeated until the original image resolution is reached.

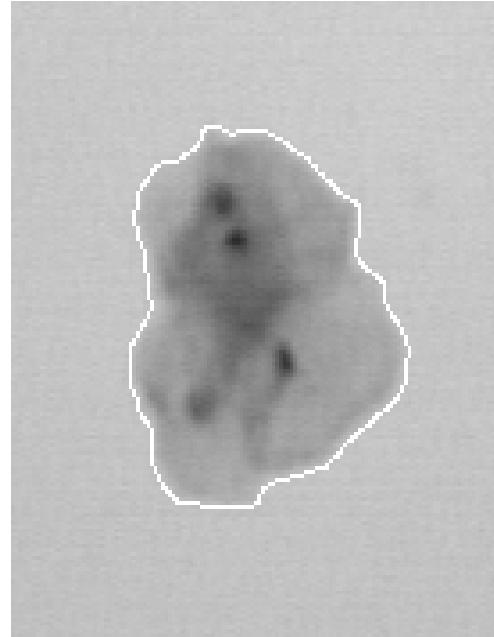


## Boundary Re-estimation



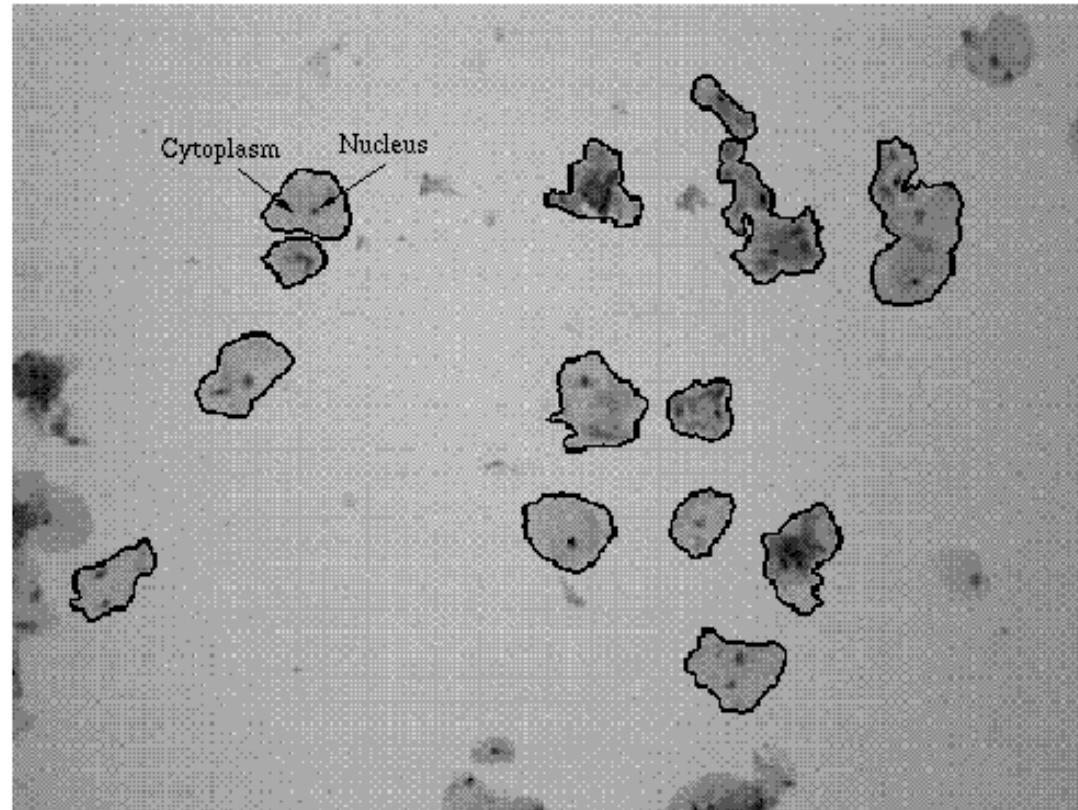


## Final Segmentation





## Full Scene Segmentation





# 3D Reconstruction from Multiple View Images

Image Processing and Computer Vision

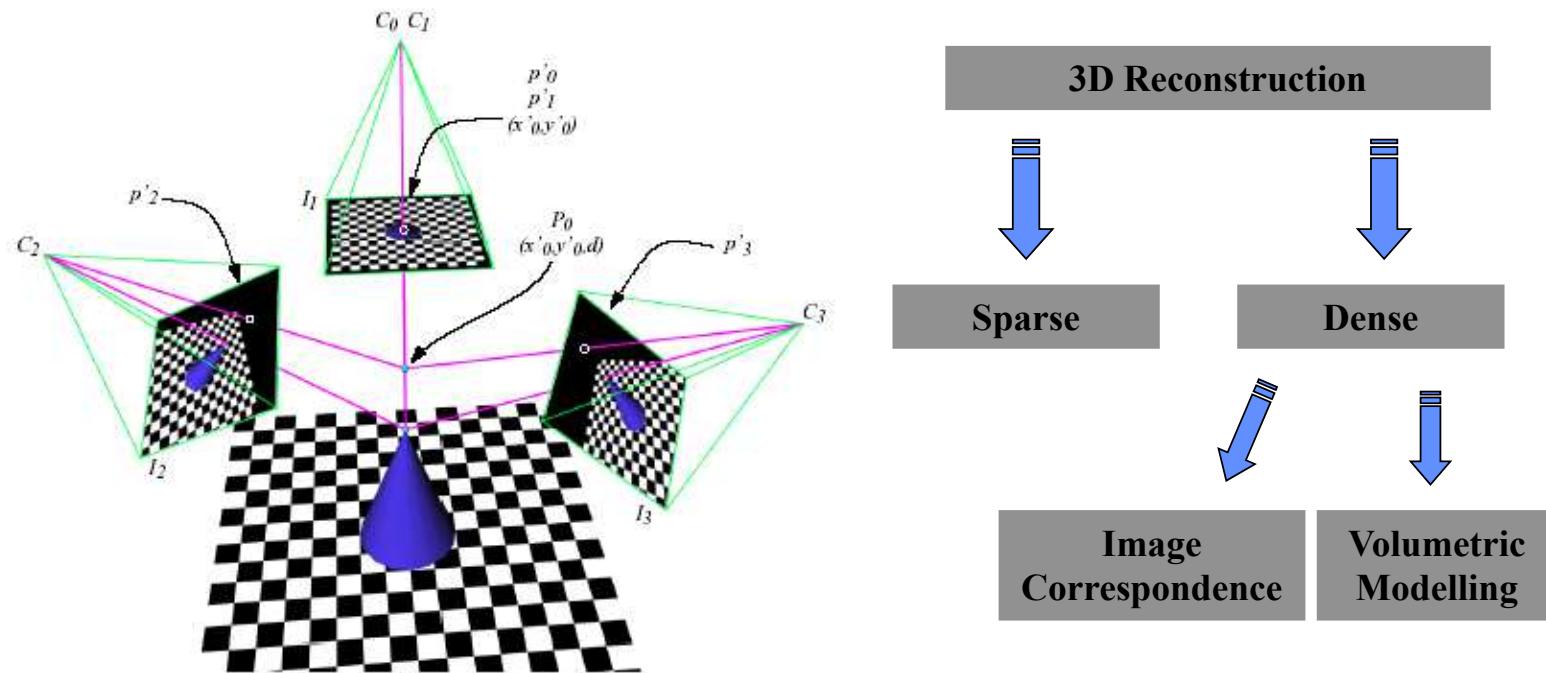


# 3D Reconstruction from Multiple View Images

- Review of 3D Reconstruction techniques
- Projective Geometry
- Volumetric Scene Modelling
  - Shape from Silhouette
  - Voxel Colouring
- Embedded Voxel Colouring
- Stereo Matching
  - Improving Speed
  - Improving Quality
- 4D Reconstruction from Image Sequences



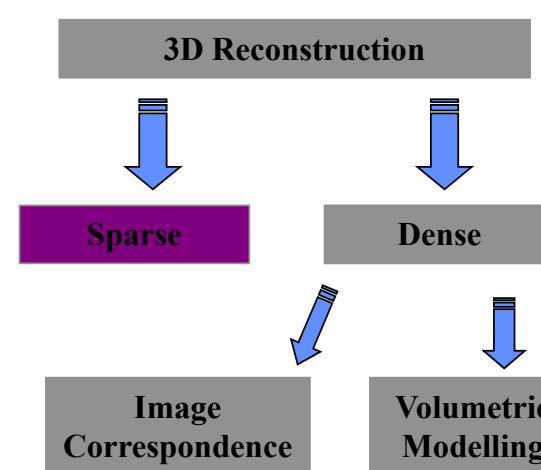
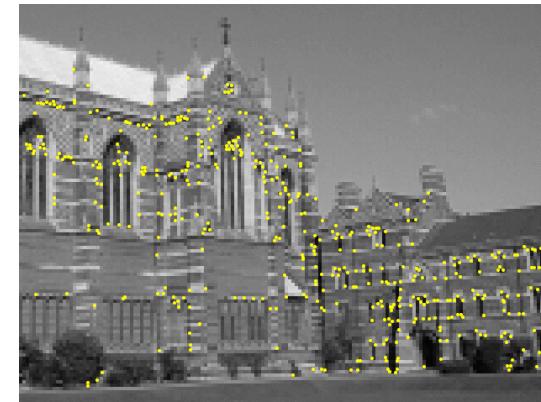
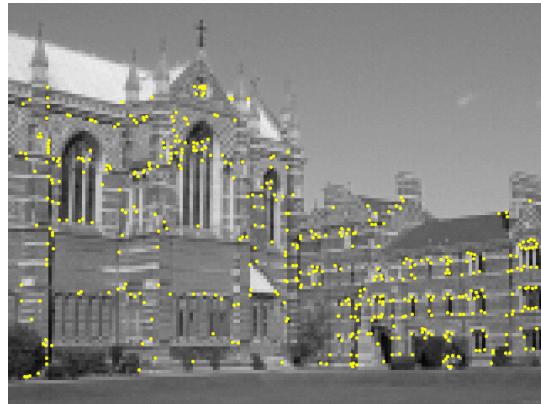
# 3D Reconstruction from Images



**Aim: Recover the lost third dimension – Depth – from images alone**



# Sparse Reconstruction





# Dense Reconstruction : Feature Correspondence Problem



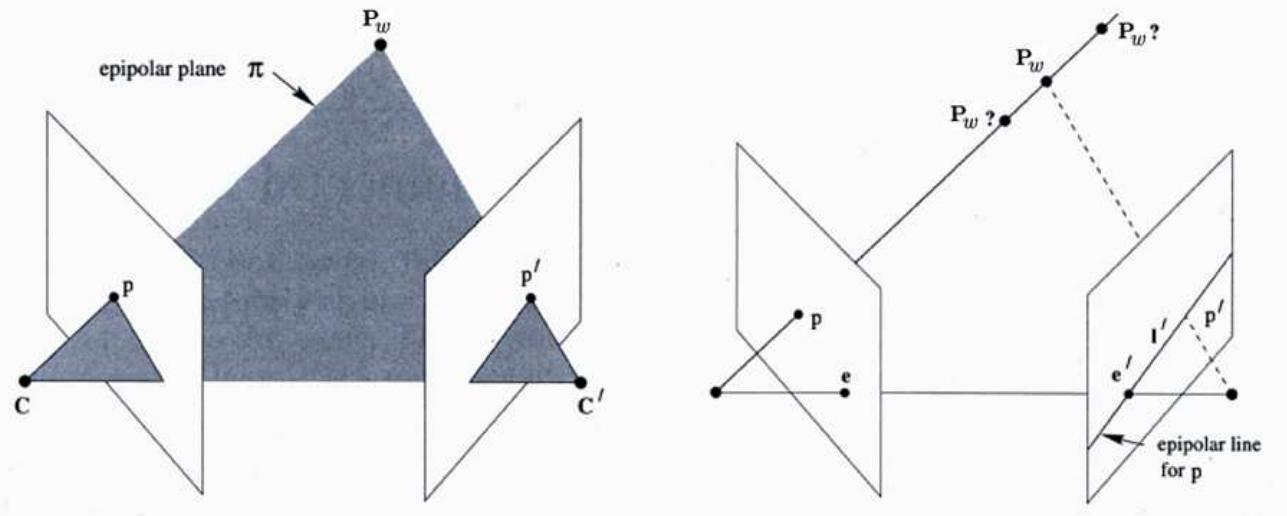


## Stereo Matching





# Epipolar Geometry

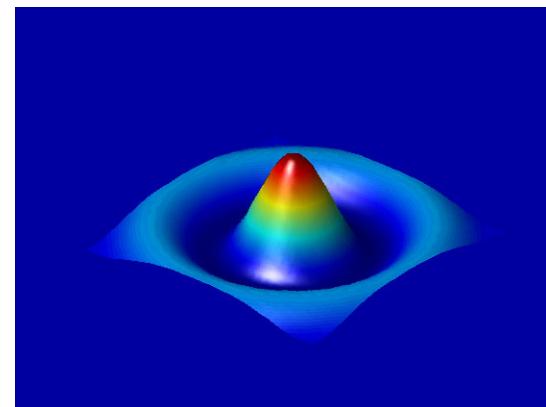
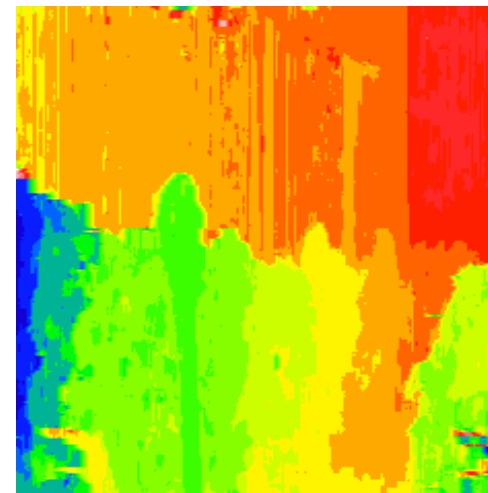




# 2.5D Sketch



$$z = f(x, y)$$



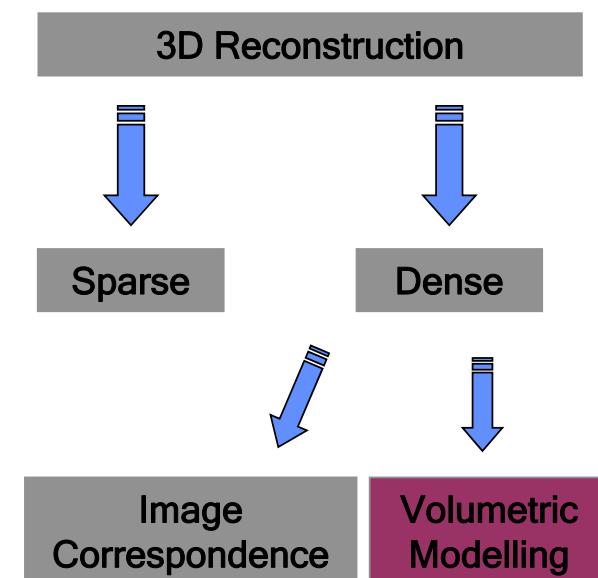
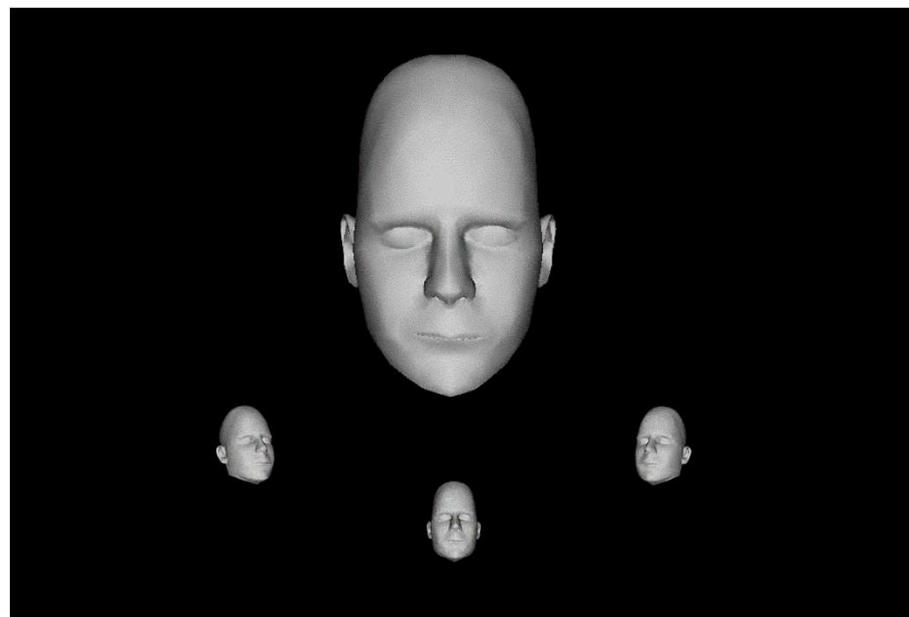


# Stereo Matching





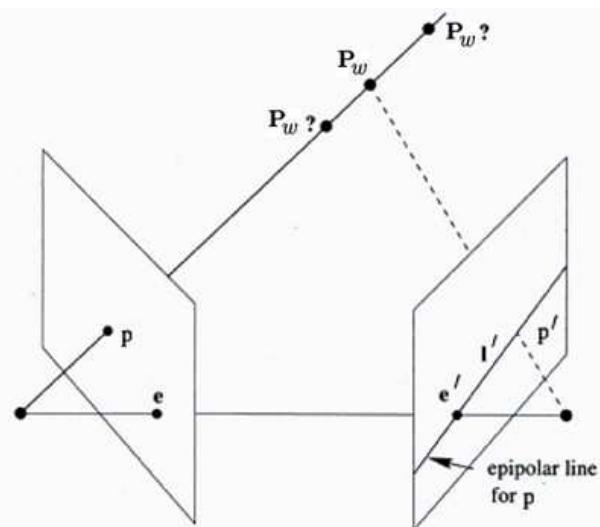
# 3D Reconstruction from Multiple Views





# Projective Geometry

## ■ Projective Coordinates



$$(x, y, w) \rightarrow \left( \frac{x}{w}, \frac{y}{w} \right)$$

$$(x, y, z, w) \rightarrow \left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w} \right)$$

Epipolar Constraint:

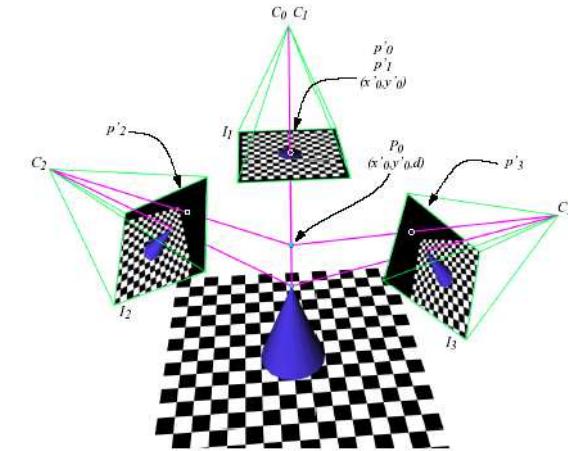
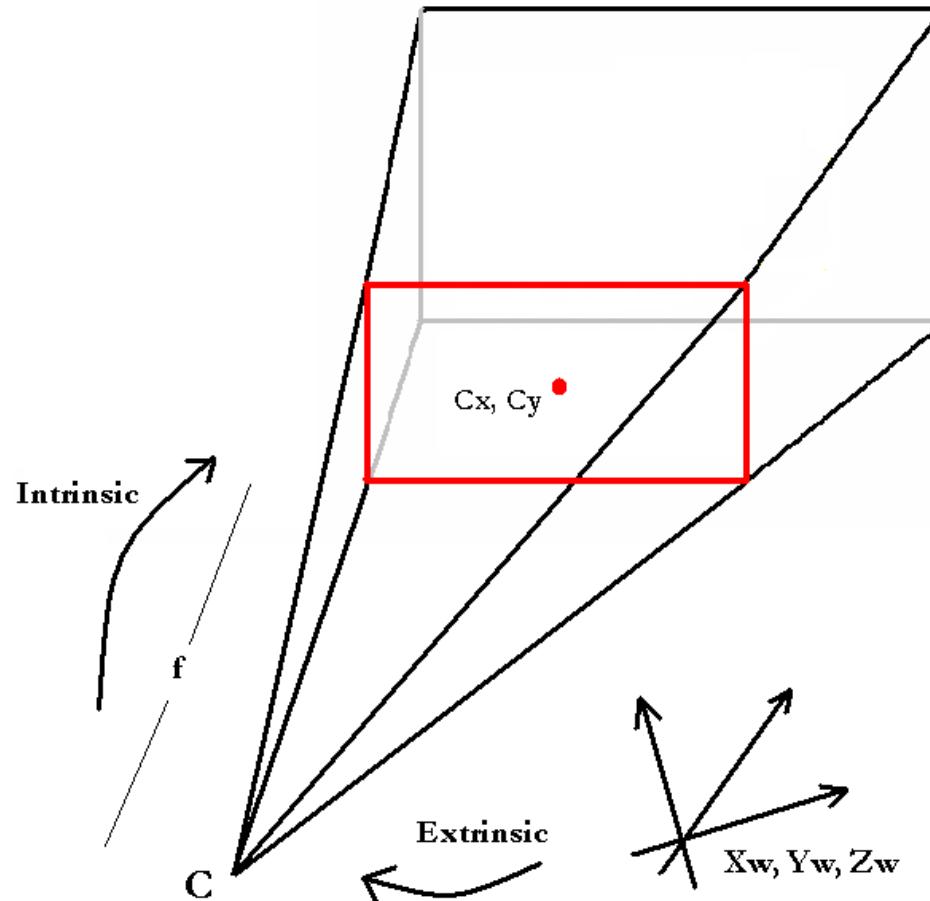
$$p'^T F p = 0$$

$F$  is a  $3 \times 3$  Matrix

Calibration = estimate  $F$



# Projective Geometry



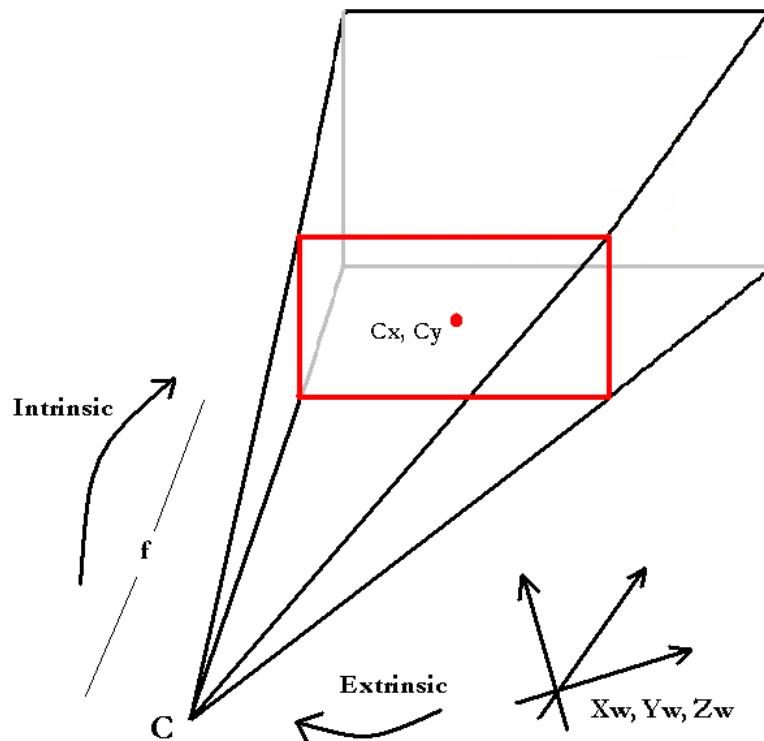
- Calibration is to find relationship:

$$(x, y, z, w) \leftrightarrow (x, y, w)$$

computing the  
Projection Matrix



# Projective Geometry



Step 1: Compute  
Extrinsic Transformation

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \\ w_w \end{pmatrix} + T \quad \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix}$$

$$\begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

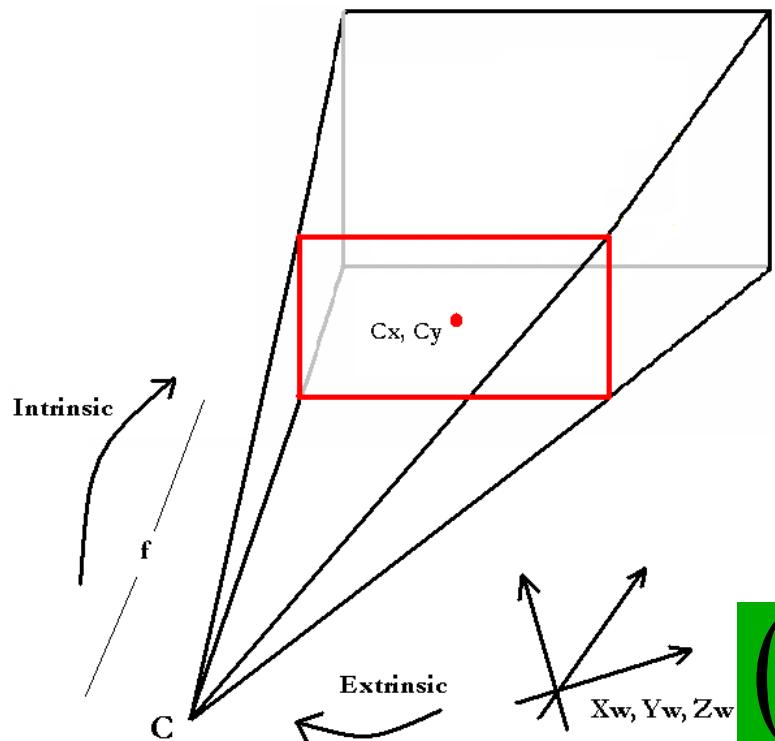
$$\begin{bmatrix} R & T \\ v^T & v \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Euclidean

Projective



# Projective Geometry



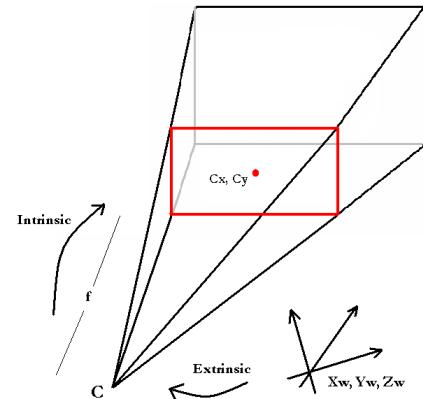
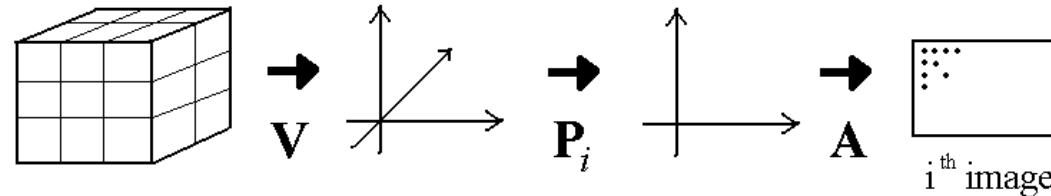
Step 2: Compute  
Projective Matrix

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix}$$

$$(x, y, z, w) \leftrightarrow (x, y, w)$$



# Projective Geometry

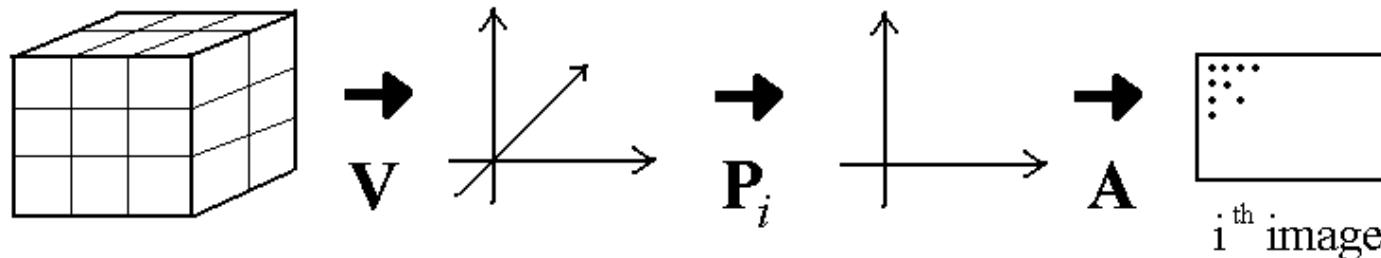


$$A = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

**Step 3: Add in  
Intrinsic Transformation**



## Projective Geometry



$$\mathbf{p}_i = \mathbf{A} \mathbf{P}_i \mathbf{V} \mathbf{v}_m$$

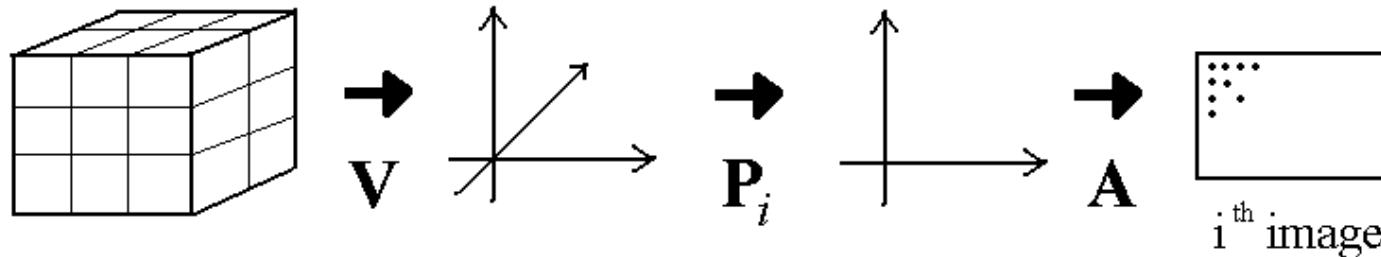
$\mathbf{A} \mathbf{P}_i$  = Projection Matrix,  $\mathbf{P}$

$$\mathbf{P} = \mathbf{A} [ \mathbf{R} \mid -\mathbf{R}\mathbf{T} ]$$

$$P \begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix}$$



## Projective Geometry



$$p_i = A \ P_i \ V \ v_m$$

- Estimating the 12 parameters of the Projection Matrix is a non-trivial task
- If you are given the Projection Matrices =  $A \ P_i$
- Design  $V$  matrix to compute 3D coordinate of each voxel
- Obtain the Region of Interest in world coordinates



## Volumetric Modelling





## Shape from Silhouette





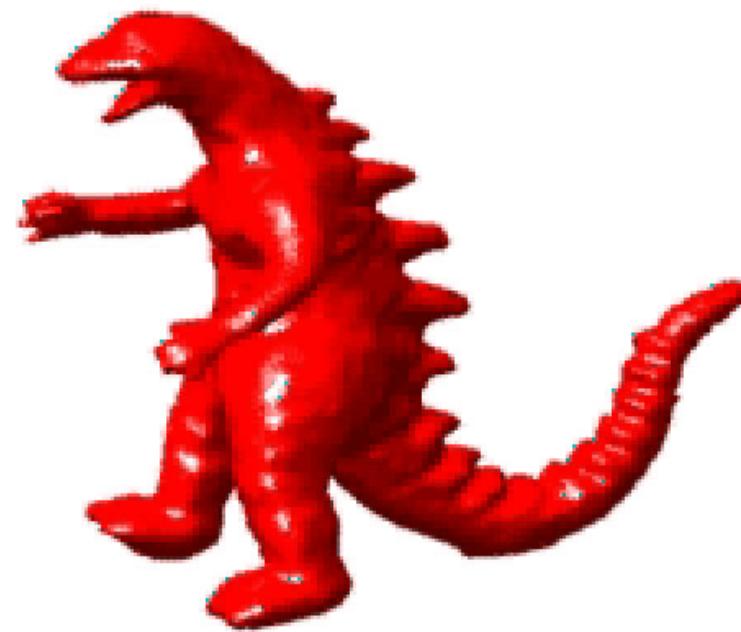
## Shape from Silhouette



- Project the frustum of each silhouette and compute intersections
- Back-Project each voxel into all images and CARVE away non-dinosaur voxels

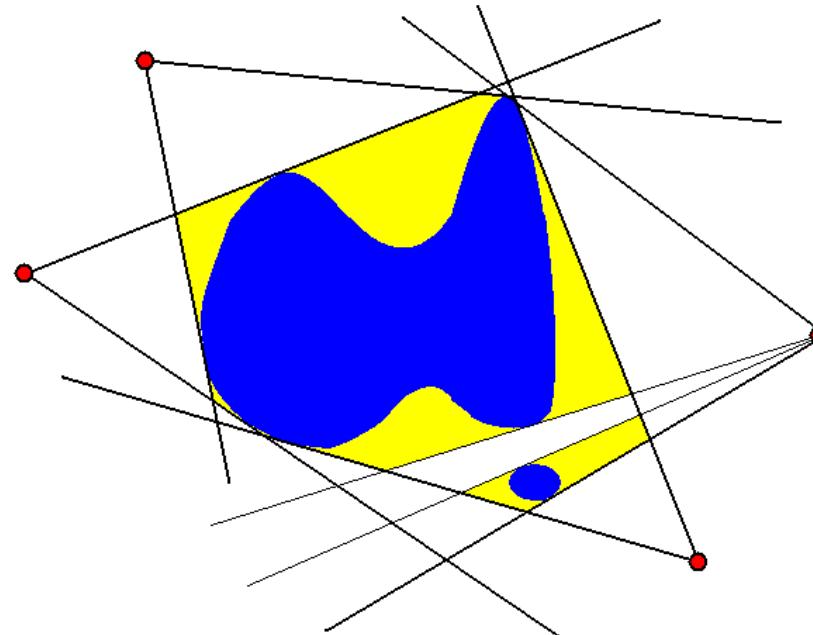


# Shape from Silhouette





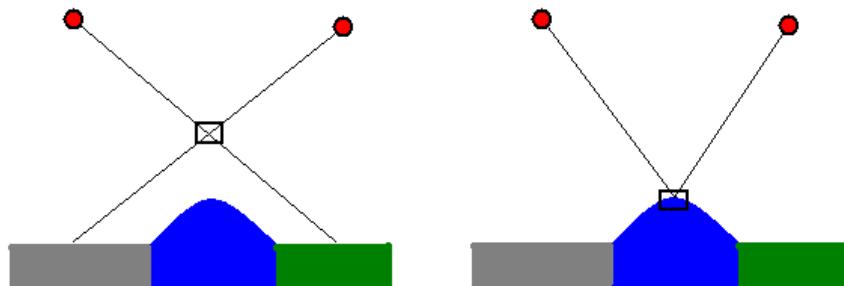
## Shape from Silhouette



- Sensitive to Segmentation Errors (eg. Table extraction)
- Reconstruction by geometric intersection → Visual Hull



## Shape from Photo-Consistency



Inconsistent voxels are *carved*

Space Carving or Voxel Colouring

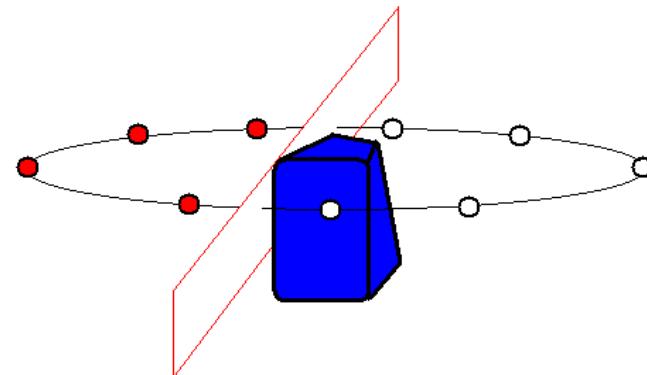
- Metric :
  - difference measure
  - variance
  - probability density function
  - histogram

- S. Seitz and C. Dyer, “Photorealistic Scene Reconstruction by Voxel Coloring”, IJCV, Vol. 35, No. 2, 1999, pp. 151-173.



## Occlusion Modelling

- Voxel Colouring
  - Ordinal Visibility Constraint - near to far traversal ordering
  - Camera location restricted
- Space Carving
  - Iterated voxel colouring
- Generalized Voxel Coloring
  - Arbitrary camera placement
  - Single sweep



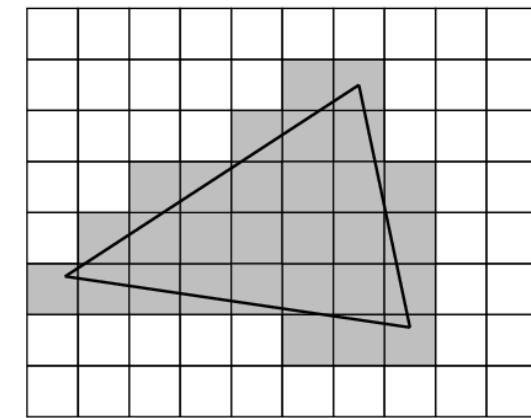
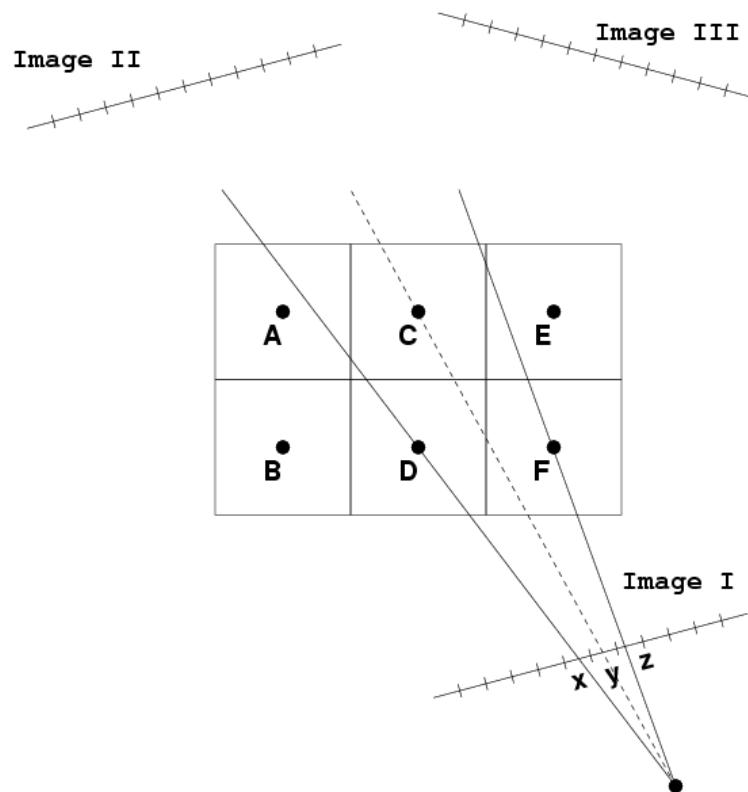


## Embedded Voxel Colouring

- C. Leung, B. Appleton, C. Sun, “*Embedded Voxel Colouring*”, Digital Image Computing: Techniques and Applications, Vol. 2, pp. 623-632, December.
- Properties of Carving
  - Water-Tight Surface Model
  - Monotonicity Carving Order
  - Causality



## Water-Tight Surface Model



- Many voxels to many pixels relationship
- Water-Tight Voxels
- Water-Tight Pixels



## Monotonic Carving Order

- Consider two carvings,  $S_A$  and  $S_B$ , computed at thresholds A and B. *Monotonicity of carving* dictates:

$$A \leq B \rightarrow S_A \subseteq S_B$$

- Therefore these sets may be embedded into a function!

$$S_A = \{\mathbf{x} | f(\mathbf{x}) \leq A\}$$

- Compute  $f$  in a single sweep
- All carvings may be obtained by thresholding



## Causality

- Monotonic Carving Order + Water-tightness → Causality
- Under a water-tight surface model, only surface voxels get carved
- Every new surface voxel must have a neighbour who has been carved
- Every voxel has a neighbour of equal or higher consistency threshold
- No local maxima in the function  $f$

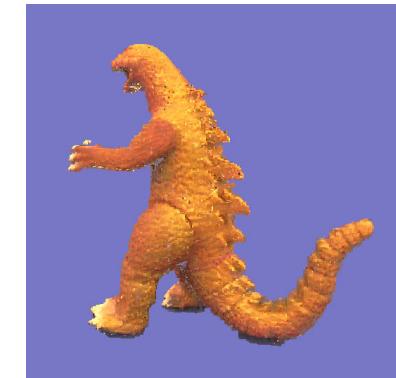


## Volumetric Modelling





## Results





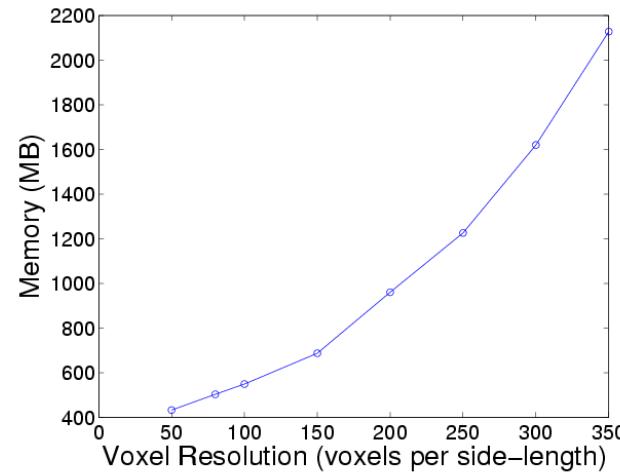
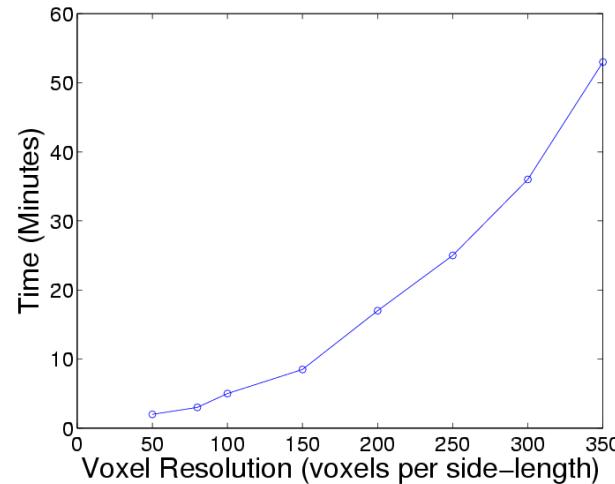
## Embedded Voxel Colouring



- Embed carvings for all possible consistency threshold into one volume



## Results



- **Embedded VC :**
  - 36 images (720x576)
  - 350x350x350 volume
  - 53 minutes (450MHz Ultra Sparc II)

- **Generalised VC :**  
(Culbertson et al.)
  - 17 images (800x600)
  - 167x121x101 volume
  - 40 minutes (440MHz HP J5000)



## Stereo Matching



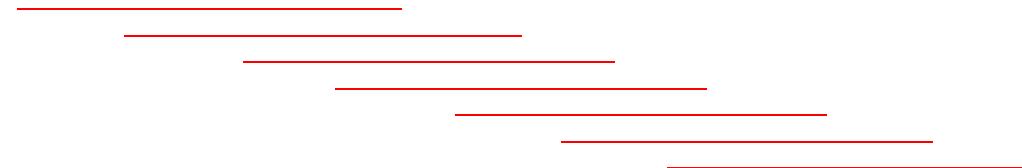


## Multiscale





## Box Filtering

$$[3 \quad 7 \quad 4 \quad 9 \quad 2 \quad 1 \quad 0 \quad 5 \quad 4 \quad 6]$$
$$[3 \quad 7 \quad 4 \quad 9 \quad 2 \quad 1 \quad 0 \quad 5 \quad 4 \quad 6]$$


Summing window of size 4 -  
7 additions of a window size of 4

$$[23 \quad 22 \quad 16 \quad 12 \quad 8 \quad 10 \quad 15]$$



## Box Filtering

$$[3 \ 7 \ 4 \ 9 \ 2 \ 1 \ 0 \ 5 \ 4 \ 6]$$

Compute Accumulated Sum -

$$[3 \ 10 \ 14 \ 23 \ 25 \ 26 \ 26 \ 31 \ 35 \ 41]$$


Take Differences to obtain same result



$$[23 \ 22 \ 16 \ 12 \ 8 \ 10 \ 15]$$

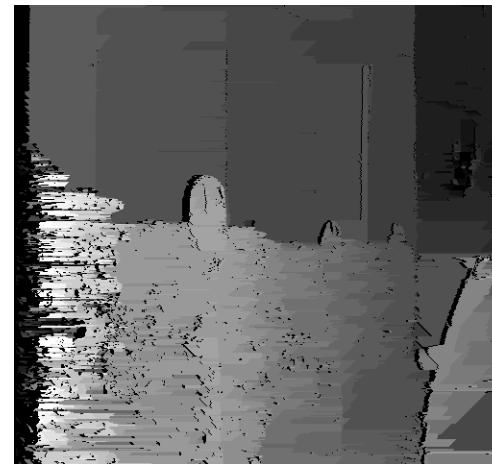


## Smoothness Constraint

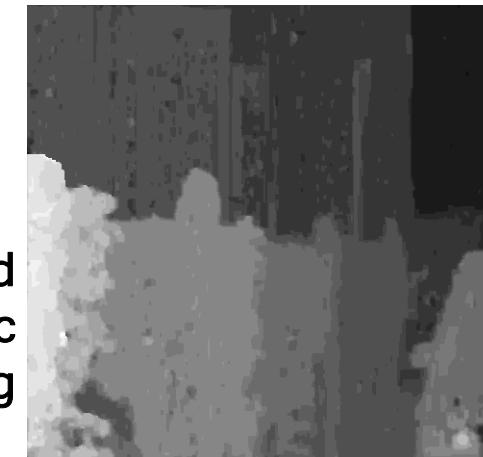


Greedy

Dynamic  
Programming



Iterated  
Dynamic  
Programming





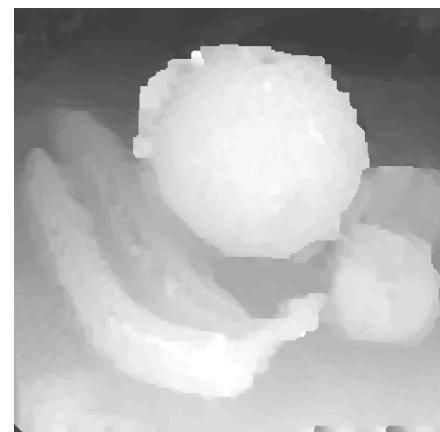
# Stereo Reconstruction using Iterated Dynamic Programming



Ground Truth



IDP



IDP



# Stereo Reconstruction using Iterated Dynamic Programming and Quadtree Subregioning

Image	Size	Scales	Disparity range	Window size	Time (seconds)
	512×480	3	-30, 0	5×5	3.28
	284×216	1	-30, 0	3×3	1.1
	512×512	3	-25, 20	9×9	5.9



# Stereo-Temporal Reconstruction (3.5D Reconstruction)

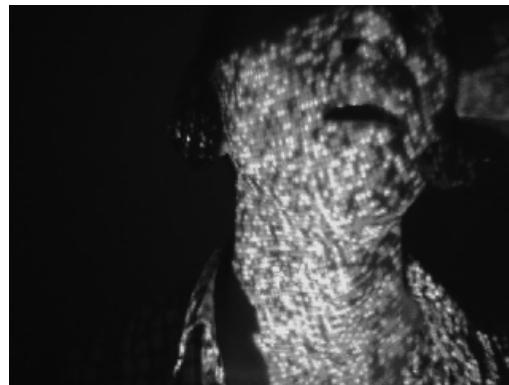


Without Temporal Coherence

With Temporal Coherence



# Stereo-Temporal Reconstruction

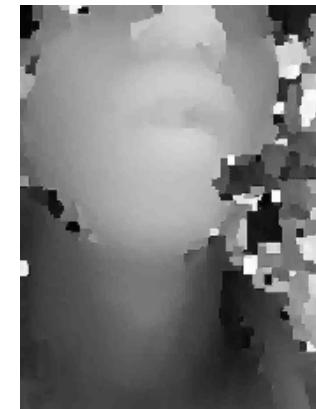


Without Temporal

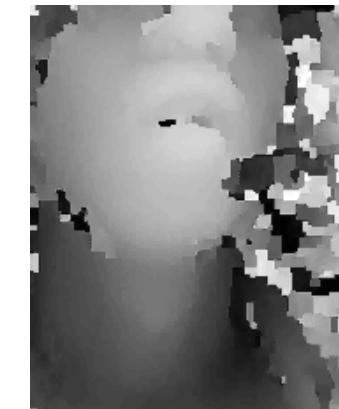


With Temporal

$5 \times 5$  window,  $K_2 \approx K_1$



Without Temporal



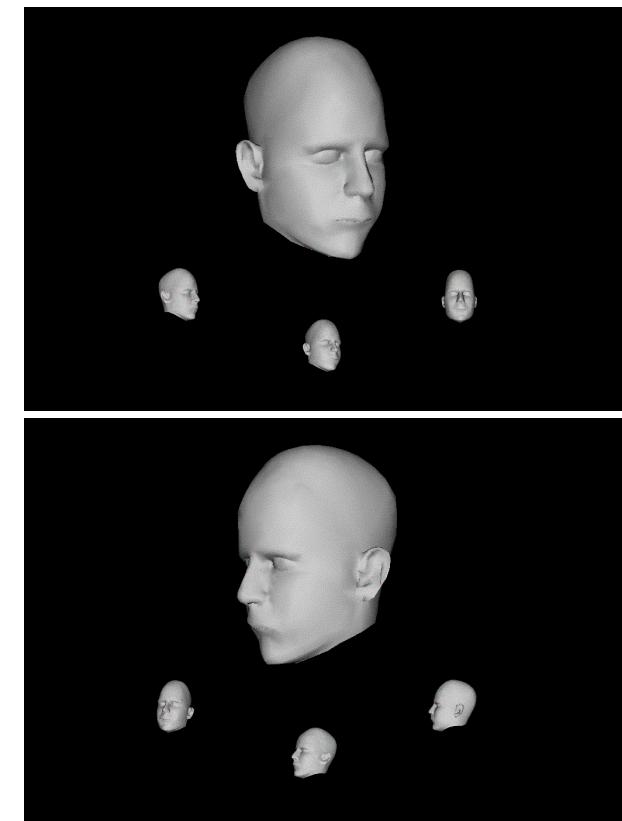
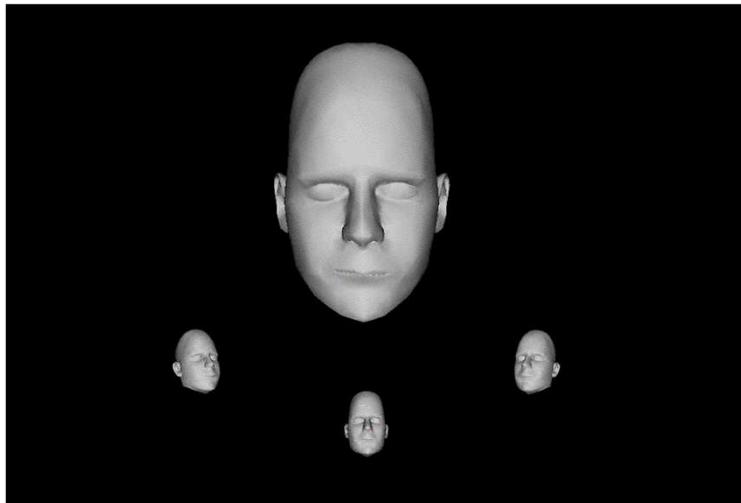
With Temporal

$3 \times 3$  window,  $K_2 > K_1$



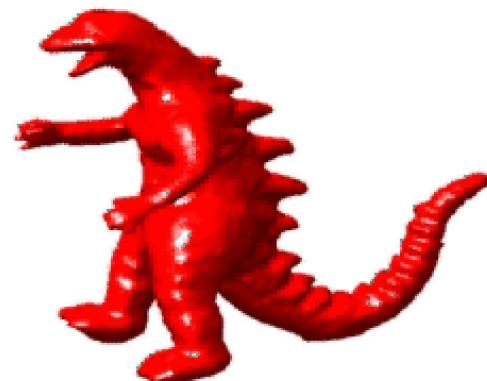
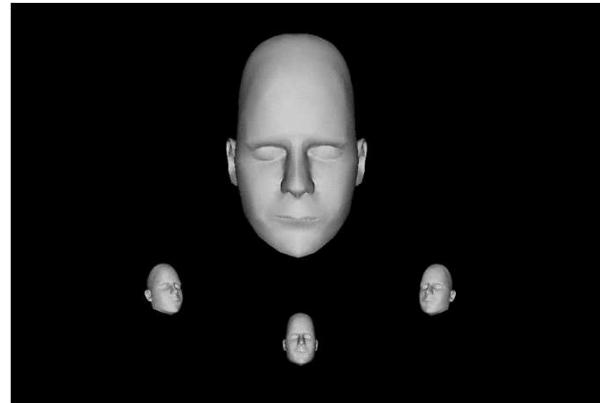
# 3D Dynamic Scene Reconstruction from Multiple View Image Sequences

## (4D Reconstruction)





# 3D Reconstruction from Multiple View Images





# **Discrete Energy Minimisation**

Applications in Image Processing

Credit: Ben Appleton



## What's it all about?

- Many methods in image analysis are task-oriented
  - Like a recipe: perform operations a, b, c on the image to produce the desired output
  - Example: Perform a thresholding then an opening to find the left ventricle
- Energy minimisation is goal-oriented
  - State your aim, and then apply an algorithm to find it optimally
  - Your answer will always make sense, as the solution will be the best according to your goal



## What we'll be talking about

- Dynamic Programming
- Shortest Path Algorithms: Dijkstra and Fast Marching
- Evolving Contours and Surfaces: Level Sets
- Applications



## Dynamic Programming/Viterbi

- Motivating Example: String Matching
  - Given two strings, where one has had letters deleted, inserted or mistyped, find the minimum number of changes to make them match.
  - Used in some word processors to autocorrect a spelling error to the nearest word from a dictionary
  - One of the main techniques to perform stereo matching (with suitable alteration)



## Fundamental Operations

Operation	Cost
Delete a letter from the 'left' word	1
Delete a letter from the 'top' word	1
Match two letters	0 if they match, 1 otherwise



## Distance function

- The value of an element  $(x,y)$  in the matrix is the cost of matching the corresponding sub words.
- For example,  $d(5, 4)$  is the cost of matching ‘ELEPH’ to ‘EALE’
- $d(8, 8)$  will then be the cost of matching ‘ELEPHANT’ to ‘EALEPANT’

Matching ‘ELEPHANT’  
to ‘EALEPANT’

	E	L	E	P	H	A	N	T
E								
A								
L								
E								
P								
A								
N								
T								



## Recursive Problem Definition

- Recursive definition of matching distance:

$$d(x, y) = \min \left\{ \begin{array}{l} d(x - dx, y - dy) + c(x, y, dx, dy) \\ 0 \leq dx, dy \leq 1 \end{array} \right\}$$

where  $c(x, y, dx, dy)$  is the transition cost and  $d(x, y)$  is the cumulative matching cost

Operation	Corresponding subwords	Additional Cost
Delete left	ELEPHANT EALEPAN <span style="background-color: yellow;">T</span>	1
Delete top	ELEPHAN <span style="background-color: yellow;">T</span> EALEPANT	1
Match	ELEPHANT EALEPANT	0



# Dynamic Programming

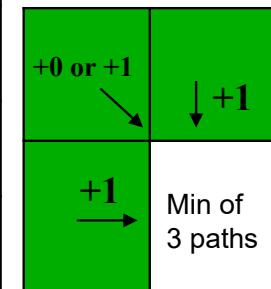
- Solve the recursion by Dynamic Programming
- A clever caching scheme
  - Express your problem recursively
  - When calling the recursive function, save the output (indexed by the input parameters) and never make the same call twice



# Matching Algorithm

## Distance function

	E	L	E	P	H	A	N	T
E	0							
A								
L								
E								
P								
A								
N								
T								



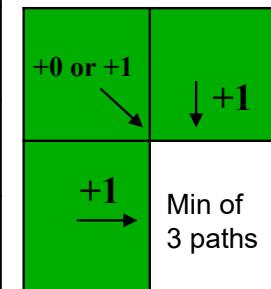
0 if match  
1 if no match



# Matching Algorithm

## Distance function

	E	L	E	P	H	A	N	T
E	0							
A	1							
L	2							
E	3							
P	4							
A	5							
N	6							
T	7							



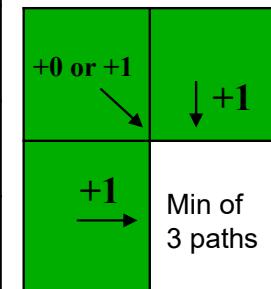
0 if match  
1 if no match



## Matching Algorithm

### Distance function

	E	L	E	P	H	A	N	T
E	0	1						
A	1	1						
L	2	1						
E	3	2						
P	4	3						
A	5	4						
N	6	5						
T	7	6						



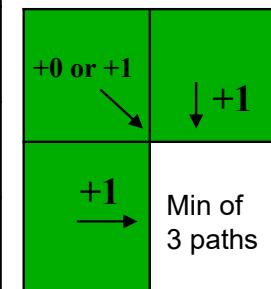
0 if match  
1 if no match



# Matching Algorithm

## Distance function

	E	L	E	P	H	A	N	T
E	0	1	2	3	4	5	6	7
A	1	1	2	3	4	4	5	6
L	2	1	2	3	4	5	5	6
E	3	2	1	2	3	4	5	6
P	4	3	2	1	2	3	4	5
A	5	4	3	2	2	2	3	4
N	6	5	4	3	3	3	2	3
T	7	6	5	4	4	4	3	2



0 if match  
1 if no match



# Matching Algorithm

## Matching path

	E	L	E	P	H	A	N	T
E	0	1	2	3	4	5	6	7
A	1	1	2	3	4	4	5	6
L	2	1	2	3	4	5	5	6
E	3	2	1	2	3	4	5	6
P	4	3	2	1	2	3	4	5
A	5	4	3	2	2	2	3	4
N	6	5	4	3	3	3	2	3
T	7	6	5	4	4	4	3	2



# Applications of Dynamic Programming/Viterbi

- String matching, stereo matching
- Hidden Markov Models (recognition)
  - Speech recognition - finding the *most likely* word
- Optimal control in State Space
  - Least time path for *speed*
  - Least energy path for *efficiency*
- General 1D sequence optimisation



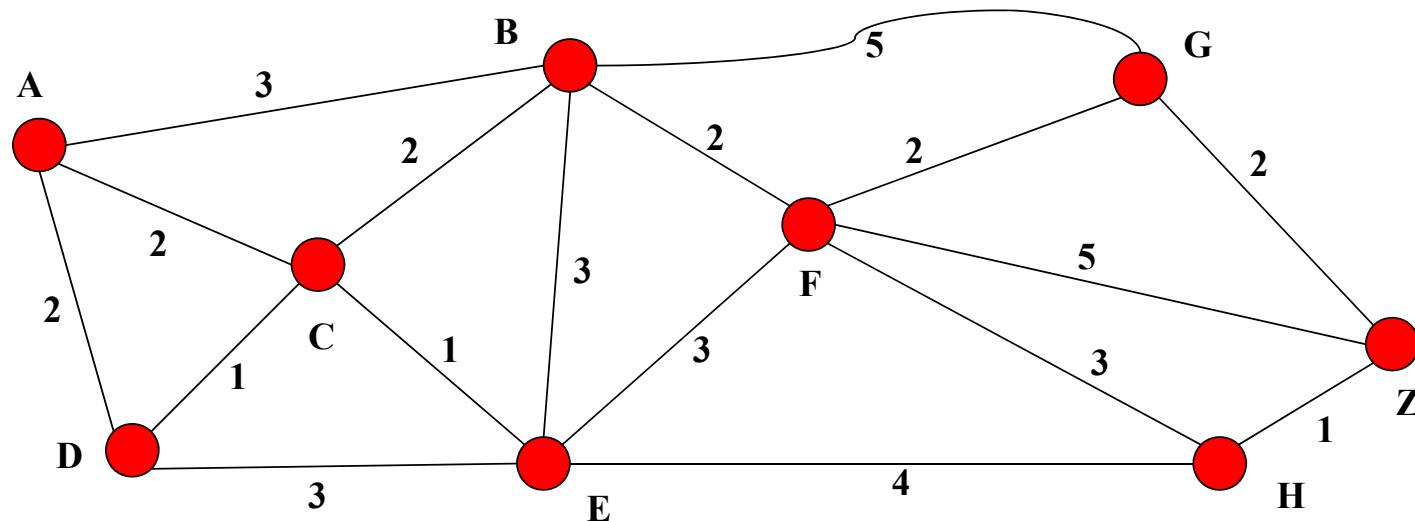
# Dijkstra's Algorithm and Fast Marching

- DP/Viterbi is good for problems that can be solved ‘one column at a time’
  - Not good for general path finding
- Dijkstra and Fast Marching are general shortest path algorithms



## Dijkstra's Algorithm

- Motivating Example:
  - Given a map, with a list of cities and the distances between them, find the shortest path from A to Z





## Dijkstra's Algorithm

- Express shortest distance recursively:

$$d(v) = \min_{\forall u \in V} \{d(u) + c(u, v)\}$$

where  $d(v)$  is the distance to vertex  $v$

and  $c(u, v)$  is the cost of travelling from  $u$  to  $v$

### ● Solve for $d(v)$ in order of distance

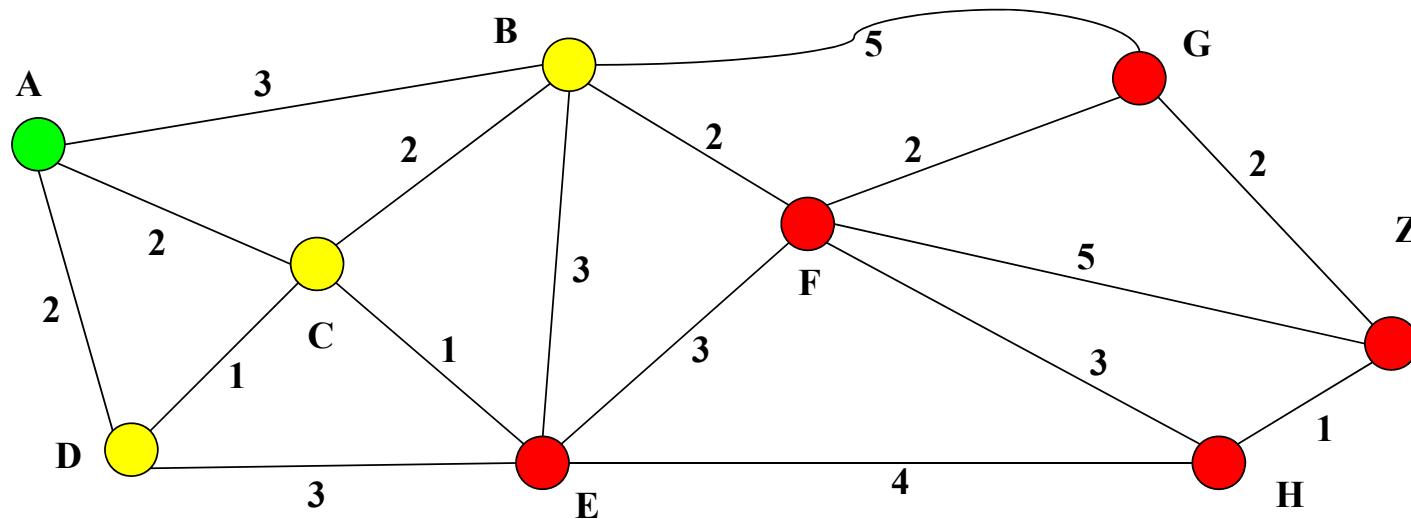
- ‘Expanding waveform’ view of shortest path finding
- Guarantees that we don’t miss any shorter paths



## Example: Path planning

- ‘Known’ vertex – distance finalised
- ● ‘Trial’ vertex – intermediate value uncertain
- ● ● ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	?	?	?	?	?

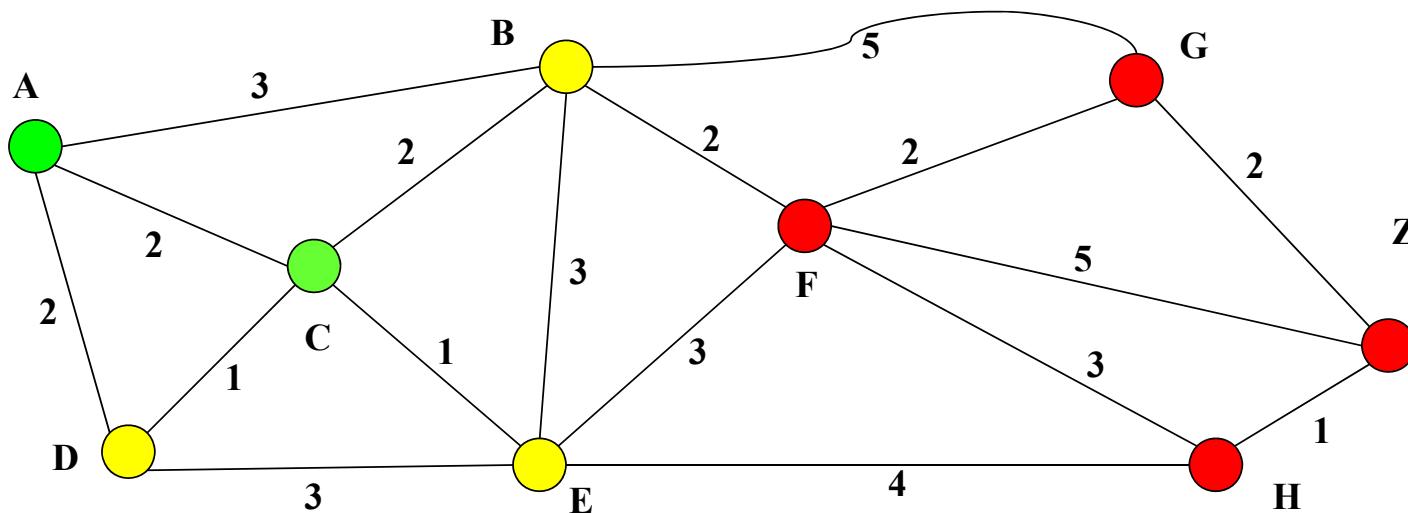




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ‘Trial’ vertex – intermediate value uncertain
- ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	?	?	?	?

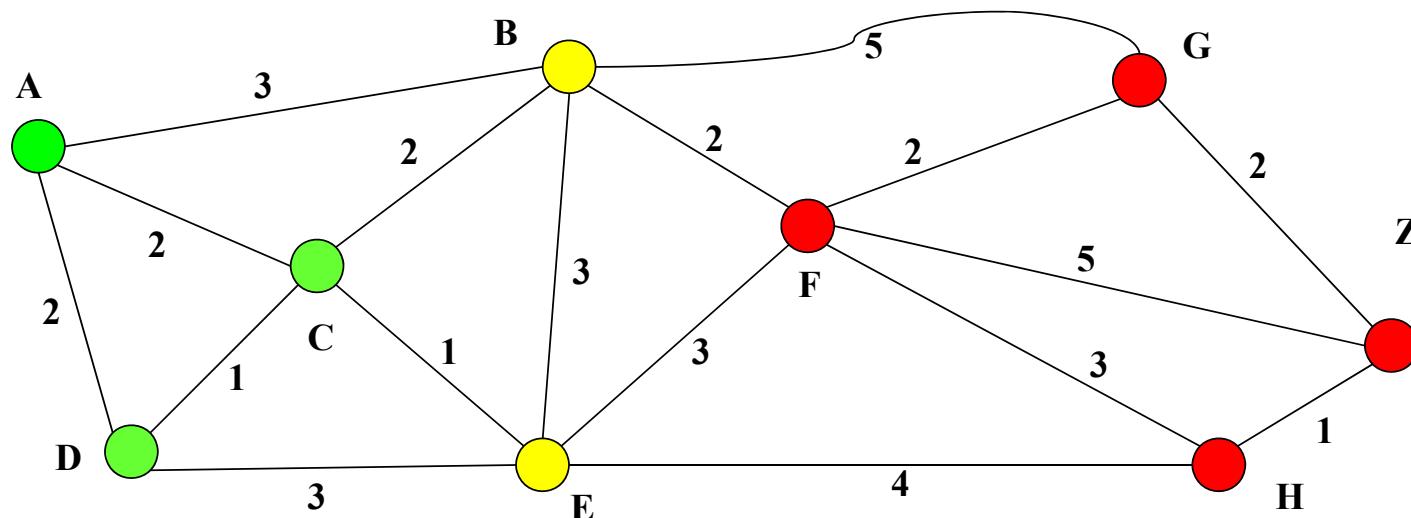




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ● ‘Trial’ vertex – intermediate value uncertain
- ● ● ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	?	?	?	?

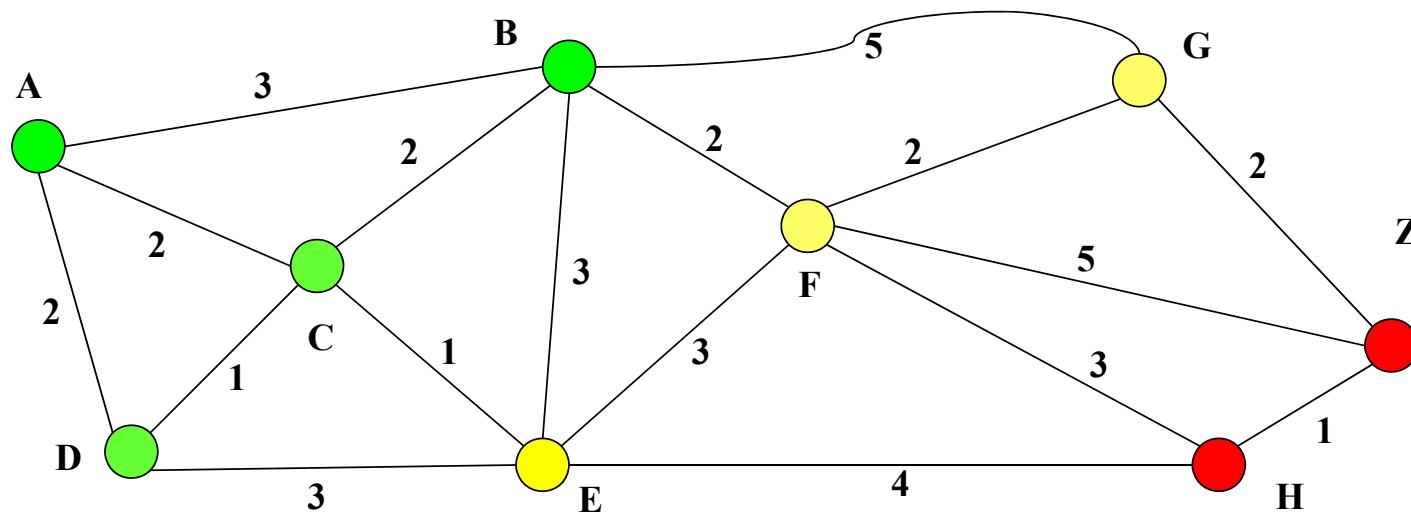




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ‘Trial’ vertex – intermediate value uncertain
- ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	8	?	?

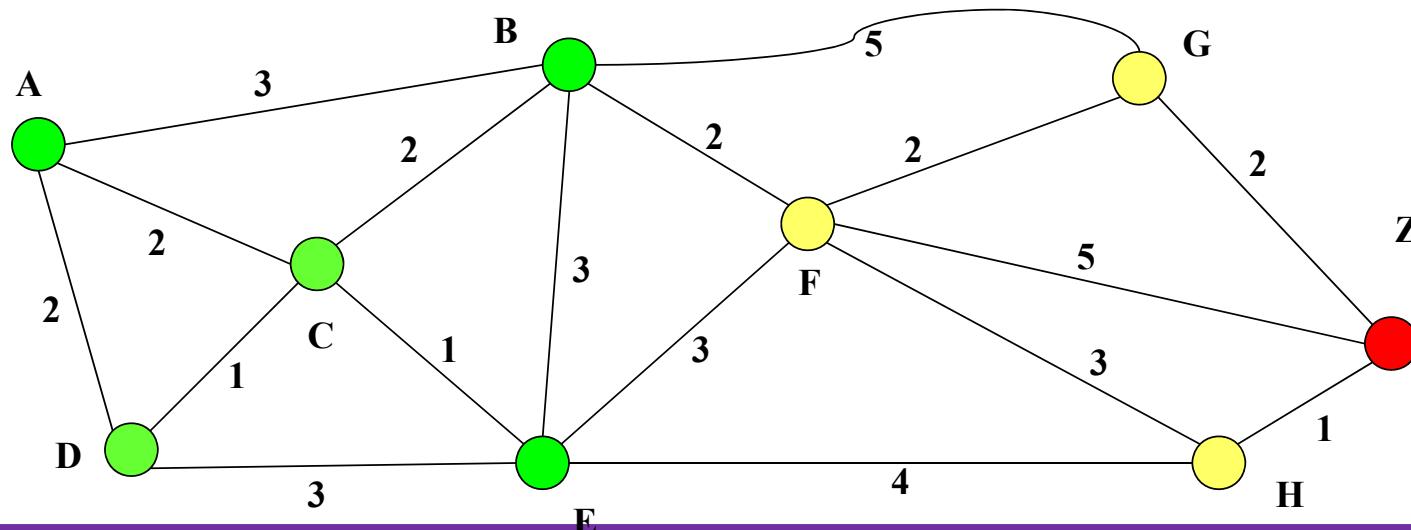




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ‘Trial’ vertex – intermediate value uncertain
- ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	8	7	?

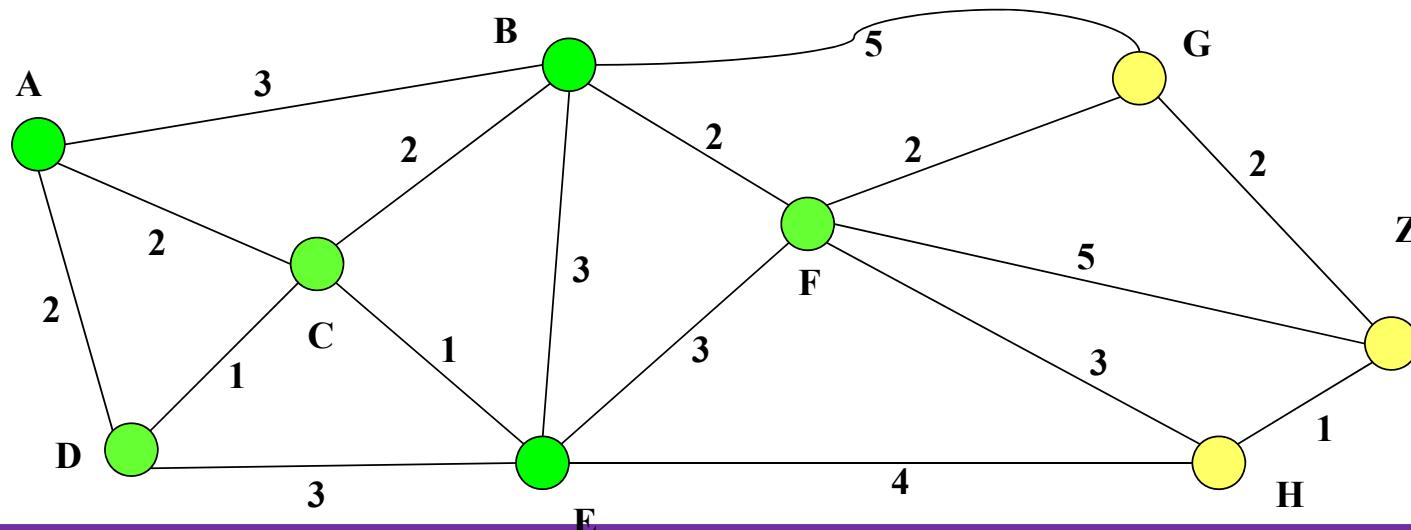




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ‘Trial’ vertex – intermediate value uncertain
- ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	7	7	10

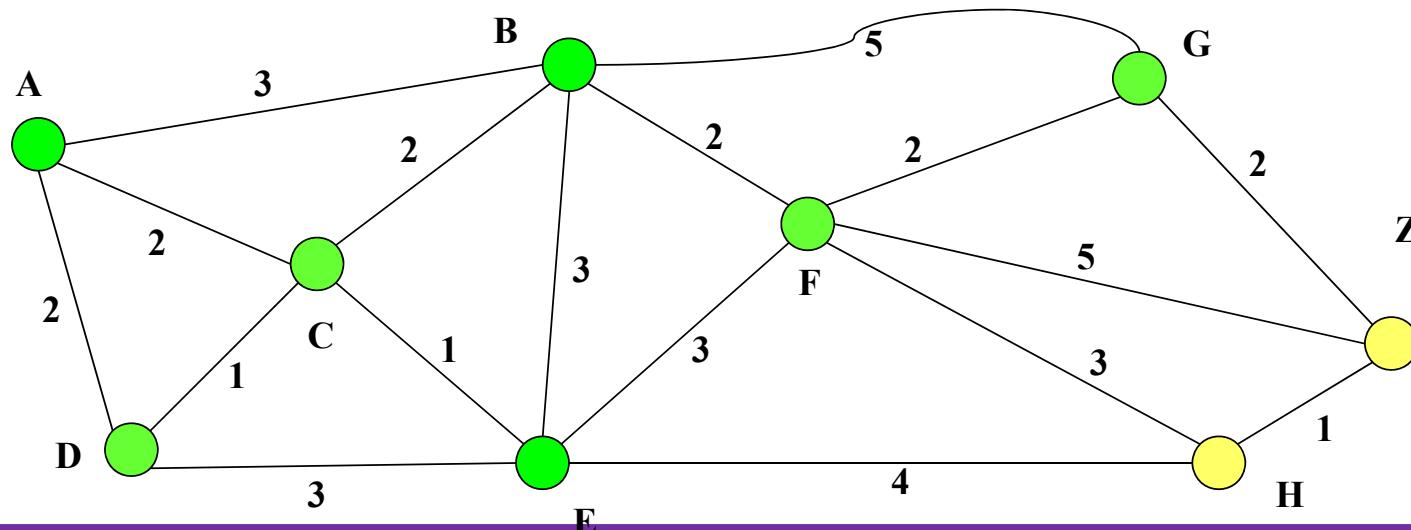




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ‘Trial’ vertex – intermediate value uncertain
- ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	7	7	9

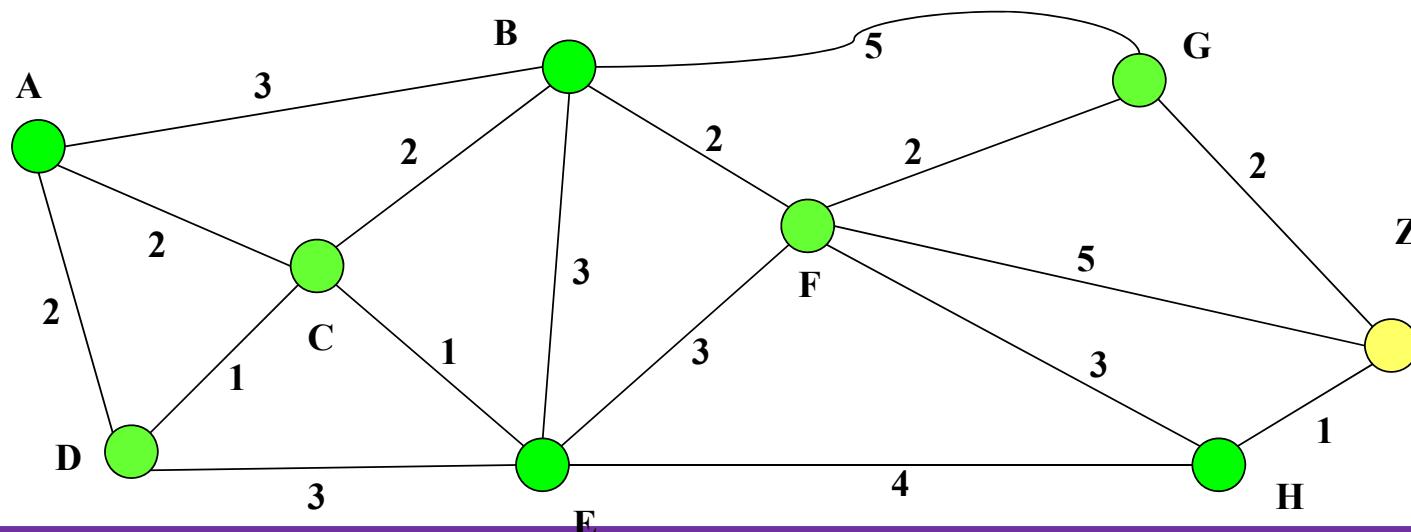




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ● ‘Trial’ vertex – intermediate value uncertain
- ● ● ‘Far’ vertex – not yet reached by wavefront propagation

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	7	7	8

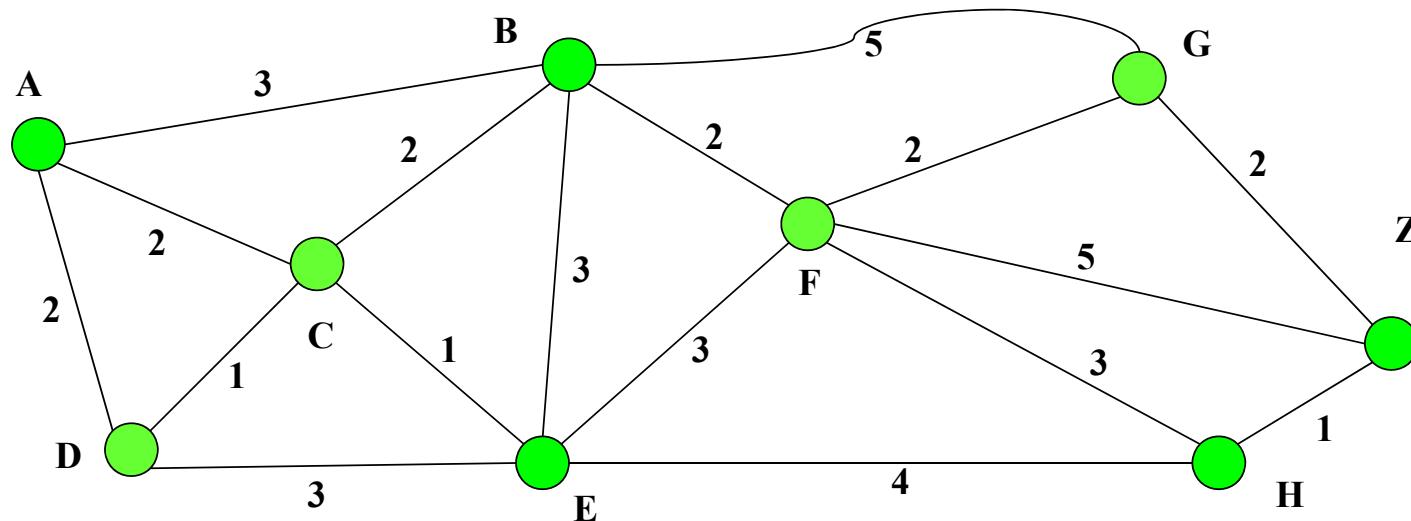




## Example: Path planning

- ‘Known’ vertex – distance finalised
- ‘Trial’ vertex – intermediate value uncertain
- ‘Far’ vertex – not yet reached by wavefront propagation

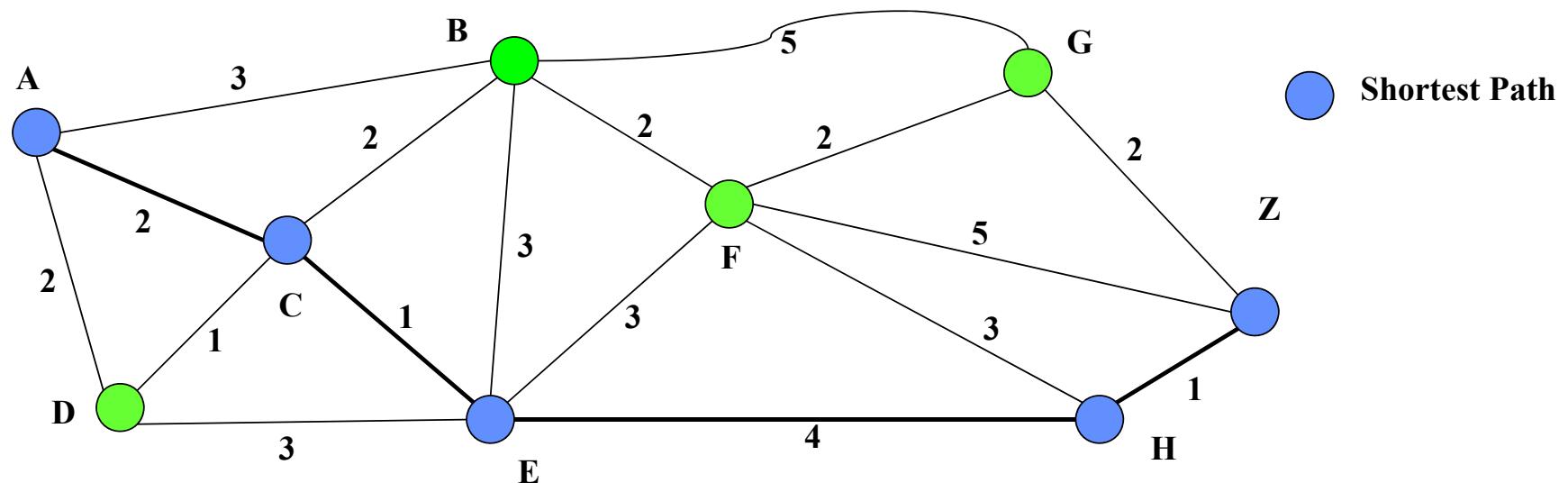
Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	7	7	8





## Example: Path planning

Vertex	A	B	C	D	E	F	G	H	Z
Distance	0	3	2	2	3	5	7	7	8
Backward Pointers	.	A	A	A	C	B	F	E	H

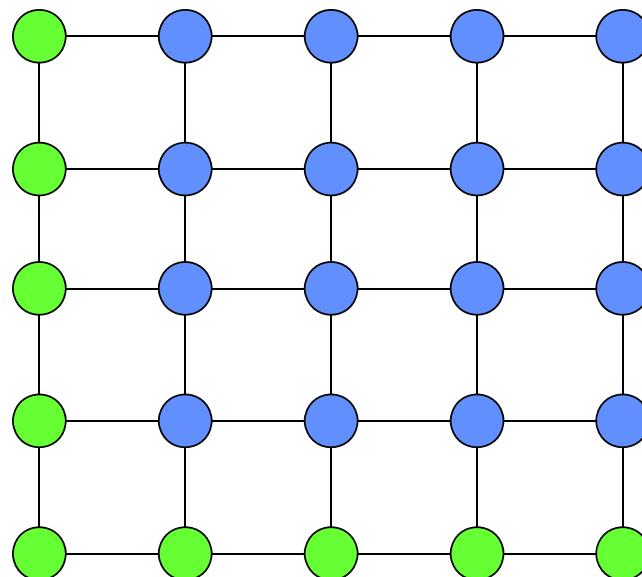




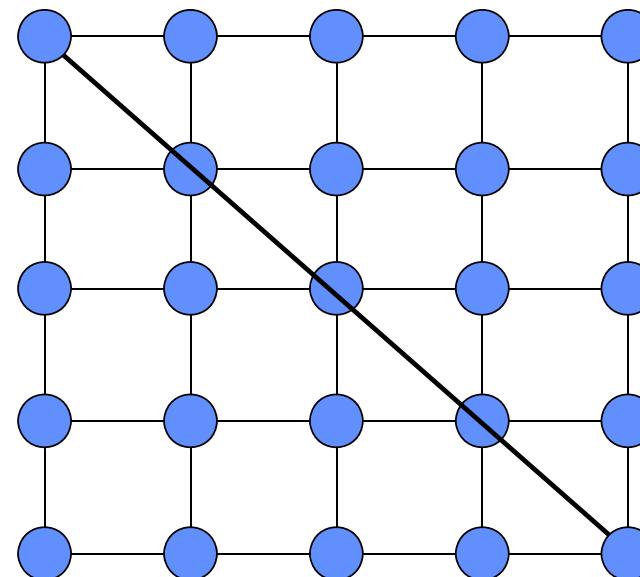
## Dijkstra's Algorithm - Anisotropy

- Dijkstra's Algorithm will find the shortest network path, but can't be extended to shortest Euclidean path

**Shortest Network Path (Length 8)**



**Shortest Euclidean Path (Length 5.66)**





## Fast Marching Method

- In Euclidean space, a distance function  $d(x, y)$  is defined by

$$|\nabla d| = 1, d(C) = 0$$

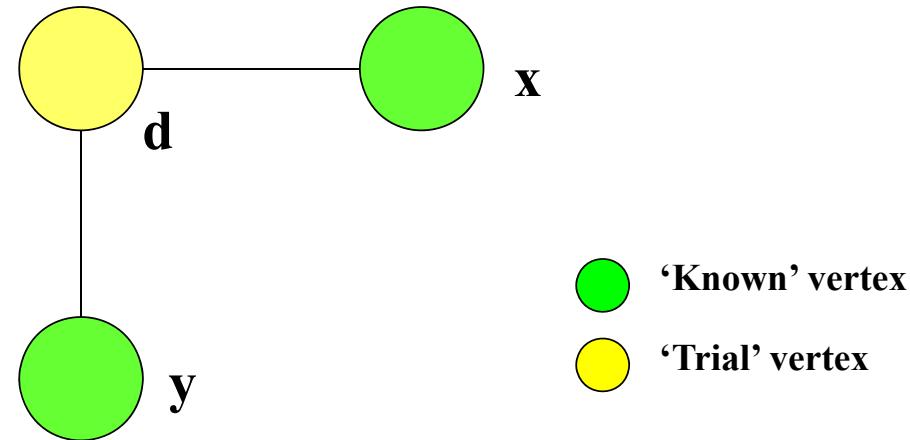
for some initial contour  $C$  of zero distance

- Solving the gradient equation for  $d$  at each point in the image grid will give an isotropic (Euclidean) distance function
- Very similar to Dijkstra's algorithm, except that all 'Known' neighbours (pixels) are used in computing the distance of a 'Trial' point.



## Fast Marching Method

Vertices labelled by distance



- Distance is computed by solving:

$$(d - x)^2 + (d - y)^2 = 1$$

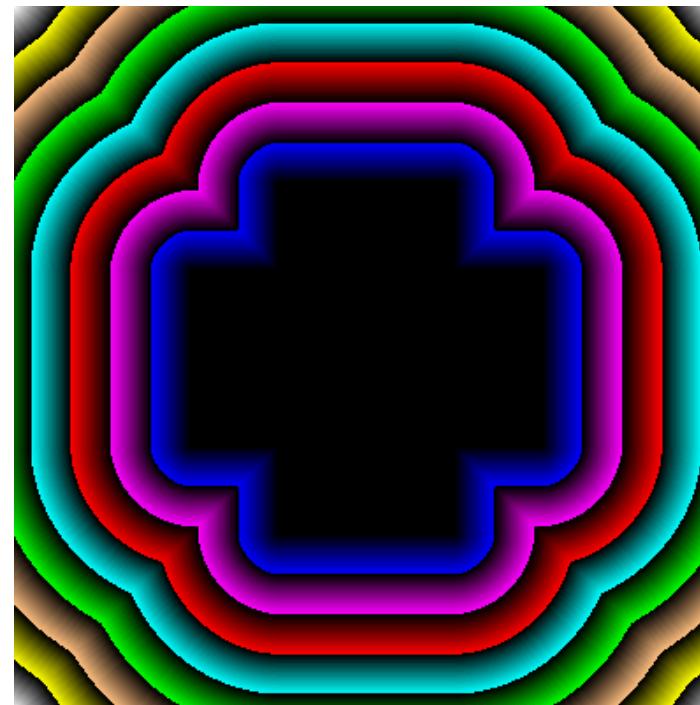


## Fast Marching Examples

A simple contour

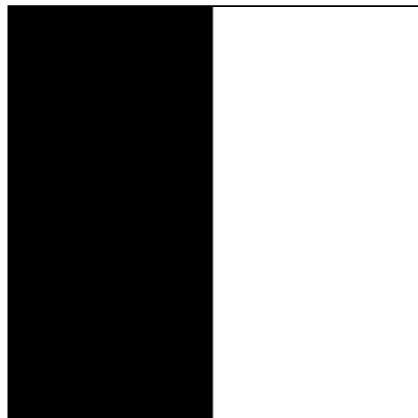


The distance function computed  
by Fast Marching



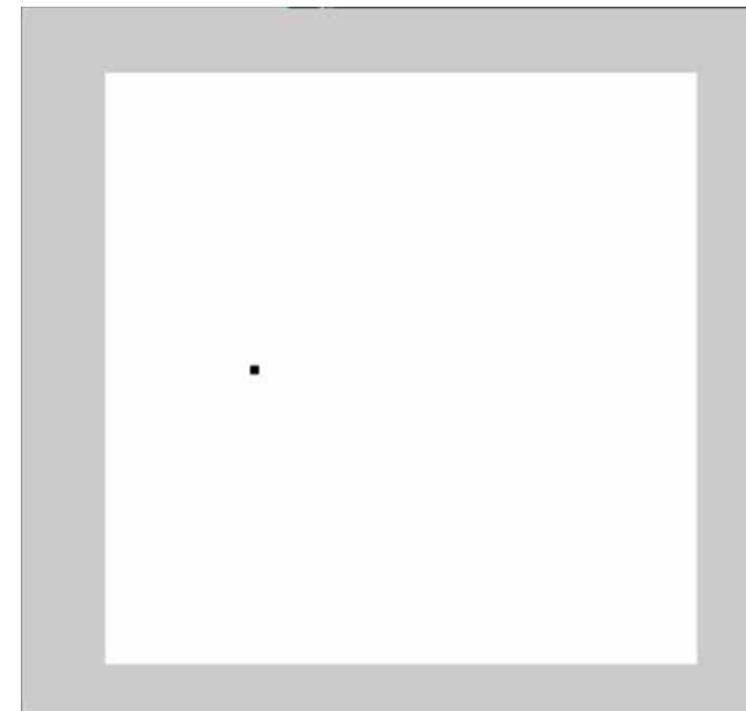


A weighted domain

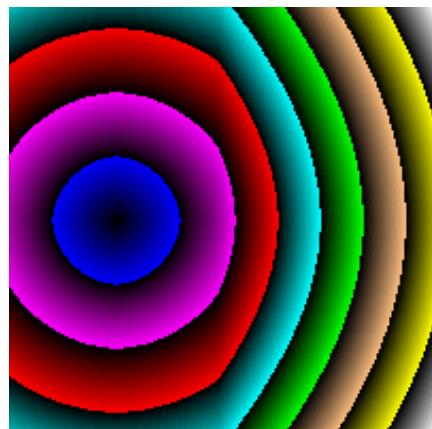


## Fast Marching Examples

The Fast Marching computation wavefront



The distance function  
from a point

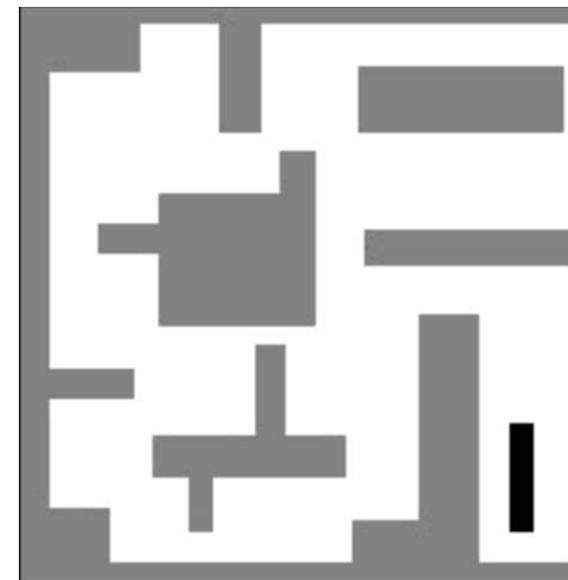




# Optimal Control by Shortest Paths

- By suitably phrasing a control theory problem, we may apply shortest path algorithms to solve it optimally
- Here we see a robot navigating through a complicated maze as fast as possible

Path planning example



Taken from <http://math.berkeley.edu/~sethian/> - an excellent reference!



## Questions?

?



## Active Contours and Surfaces

- Many problems in image analysis can be expressed as curve or surface finding
  - Segmentation in 2D searches for closed curves (1D) around each object
  - Segmentation in 3D searches for closed surfaces (2D) around each object
  - Stereo matching searches for a disparity surface (2D) in a 3D space of possible matches
  - Motion estimation searches for a velocity field, a 2D surface in a 4D space
- Some of these problems cannot be solved efficiently by global optimisation



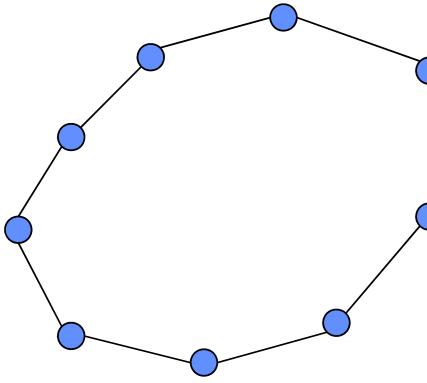
## **Snakes and Level Sets**

- Snakes and Level Sets are representations of curves and surfaces
- They allow for optimisation by curve and surface evolution towards the desired goal



## Snakes

- Snakes represent a curve or surface as a polygon or polyhedra
- Evolution is performed iteratively by moving contour points towards the goal
  - Eg. To segment an object, move the contour points toward the edges of the image. This is one way to perform edge linking.



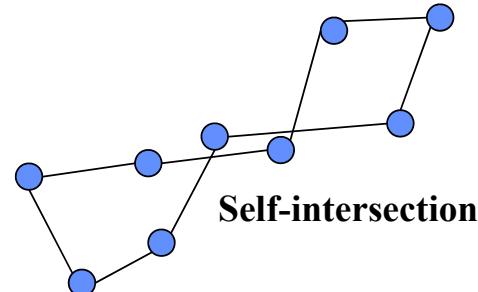
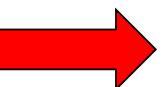
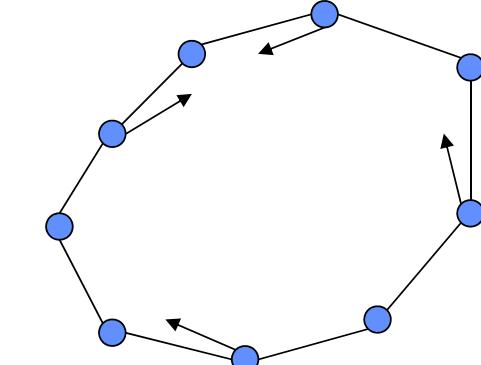
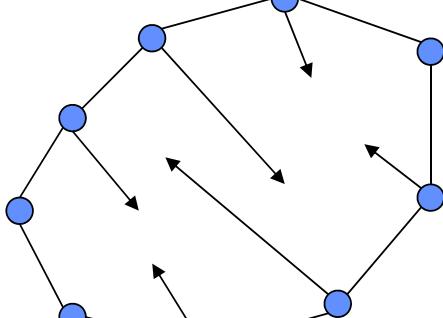


## Snakes

- Snakes have a few problems:
  - They cannot easily change topology to segment multiple objects
  - The contour points often bunch up or spread out, reducing the stability or accuracy of the solution
  - Self-intersections of the curve are difficult to detect
  - They are difficult to extend to higher dimensions

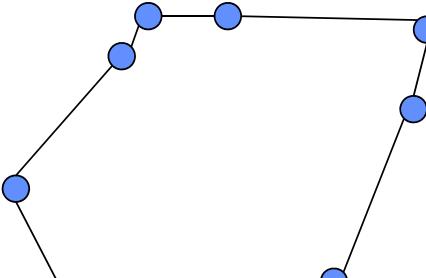


## Snakes



**Self-intersection**

**Bunching**



**Spreading**



## Level Sets

- Level Sets overcome these parameterisation problems
- Use an implicit representation of the contour  $C$  as the 0-level set of higher dimensional function  $\phi$

$$\phi(C) = 0$$



# The Level Set Evolution Equation

- Manipulate  $\phi$  to indirectly move  $C$ :

$$\begin{aligned}\phi(C) &= 0 \\ \frac{d\phi(C)}{dt} &= \frac{\partial C}{\partial t} \cdot \nabla \phi + \frac{\partial \phi}{\partial t} = 0 \\ \therefore \frac{\partial \phi}{\partial t} &= -F |\nabla \phi|\end{aligned}$$

where  $F$  is the speed function  
normal to the curve



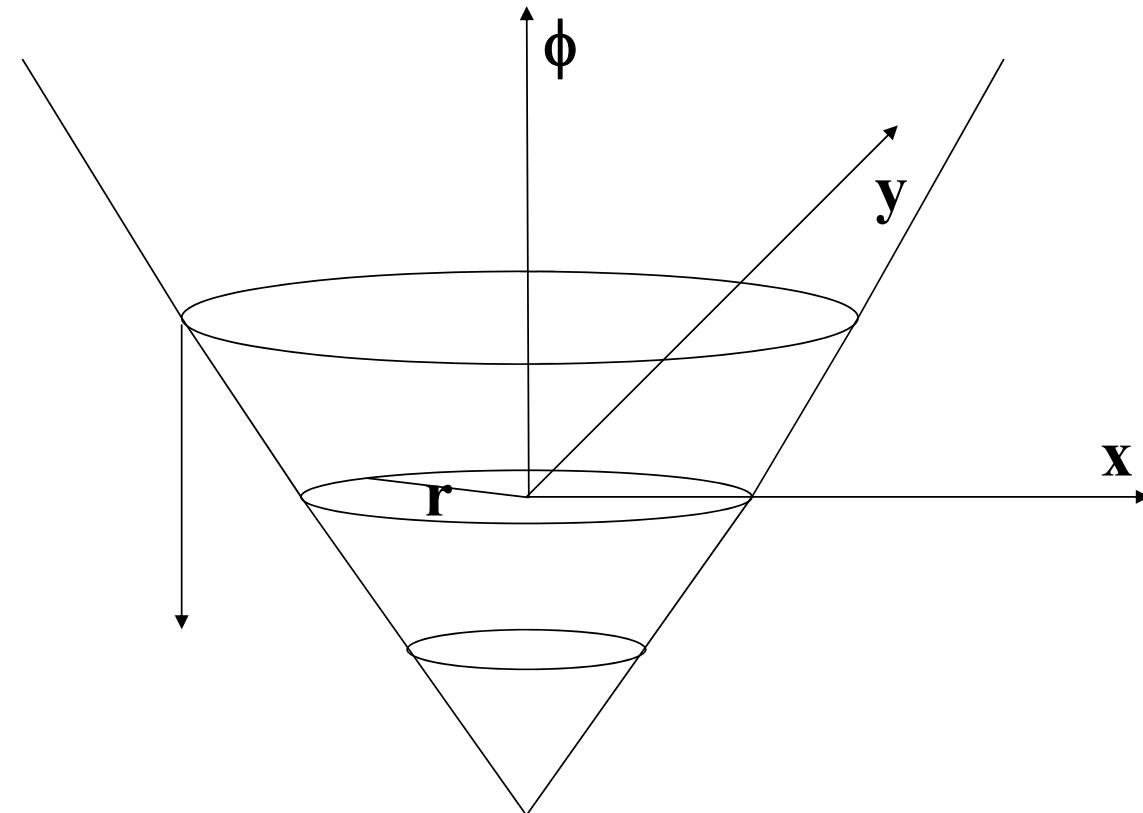
# Level Set example: An expanding circle

**Level Set representation of a circle:**

- Setting  $F = 1$  causes the circle to expand uniformly
- Observe that  $\nabla\phi = 1$  almost everywhere (by choice of representation), so we obtain the level set evolution equation:
- Explicit solution:  
which means that the circle has radius  $r + t$  at time  $t$ , as expected!

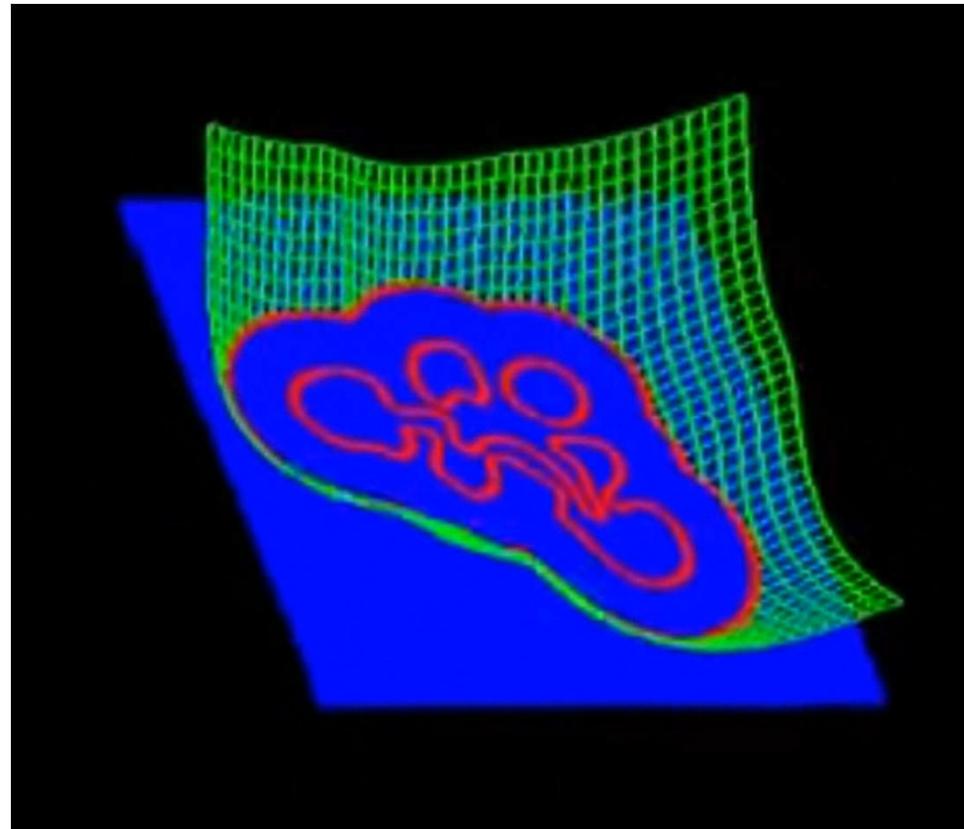


# Level Set example: An expanding circle





## Another Example





## Level Set Segmentation

- Since the choice of  $\phi$  is somewhat arbitrary, we choose a signed distance function from the contour.
  - This distance function is negative inside the curve and positive outside.
  - A distance function is chosen because it has unit gradient almost everywhere and so is smooth.
- By choosing a suitable speed function  $F$ , we may segment an object in an image



## Level Set Segmentation

- The standard level set segmentation speed function is:

$$F = 1 - \varepsilon\kappa + \beta(\nabla\phi \cdot \nabla|\nabla I|)$$

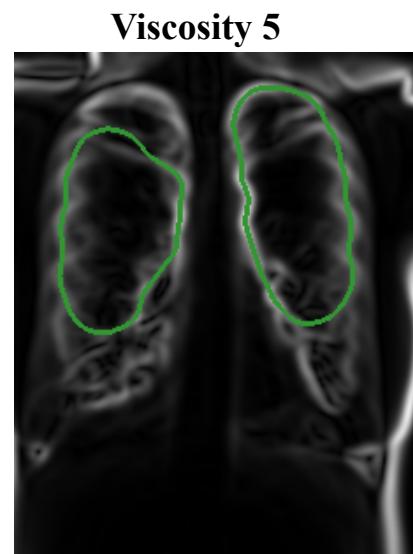
- The  **$I$**  causes the contour to inflate inside the object
- The  **$-\varepsilon\kappa$  (viscosity)** term reduces the curvature of the contour
- The final term (edge attraction) pulls the contour to the edges
- Imagine this speed function as a balloon inflating inside the object. The balloon is held back by its edges, and where there are holes in the boundary it bulges but is halted by the viscosity  $\varepsilon$ .



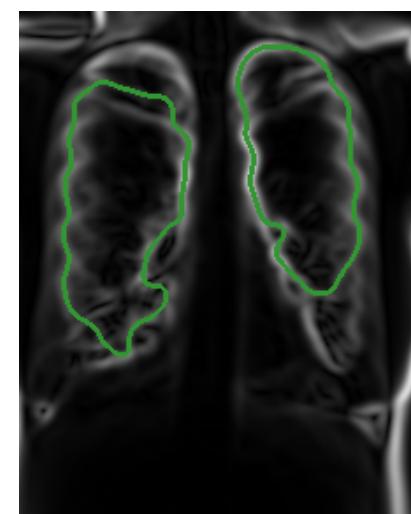
# Level Set Segmentation Example



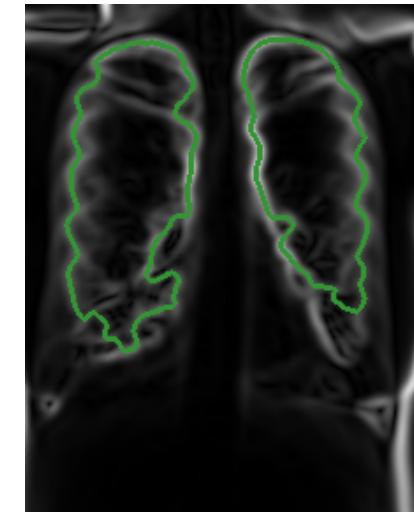
Lung x-ray



Viscosity 5



Viscosity 2



Viscosity 0.5



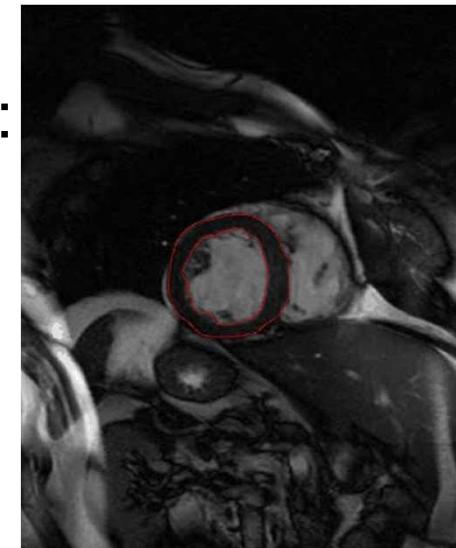
## Level Sets – further examples

Real-time segmentation:

[Sethian Segmentation.htm](#)

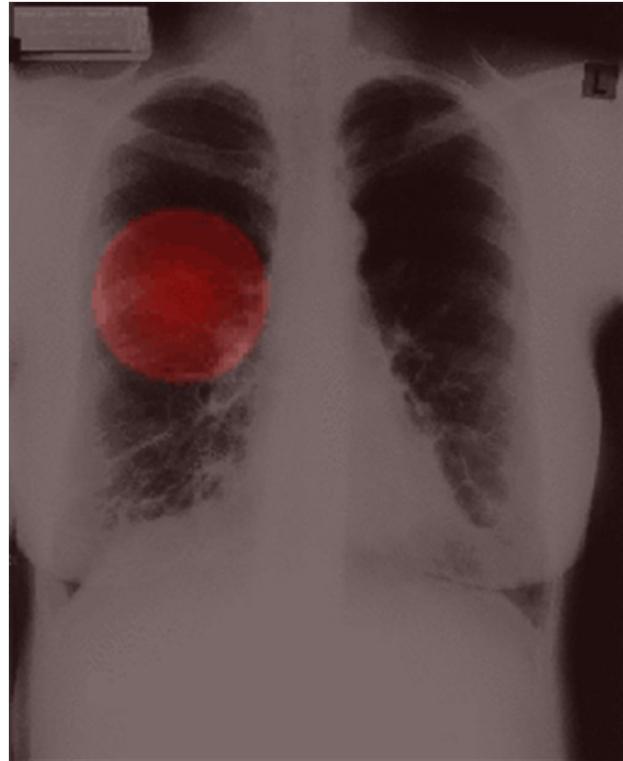
Combined segmentation and tracking:

The problem of structure:





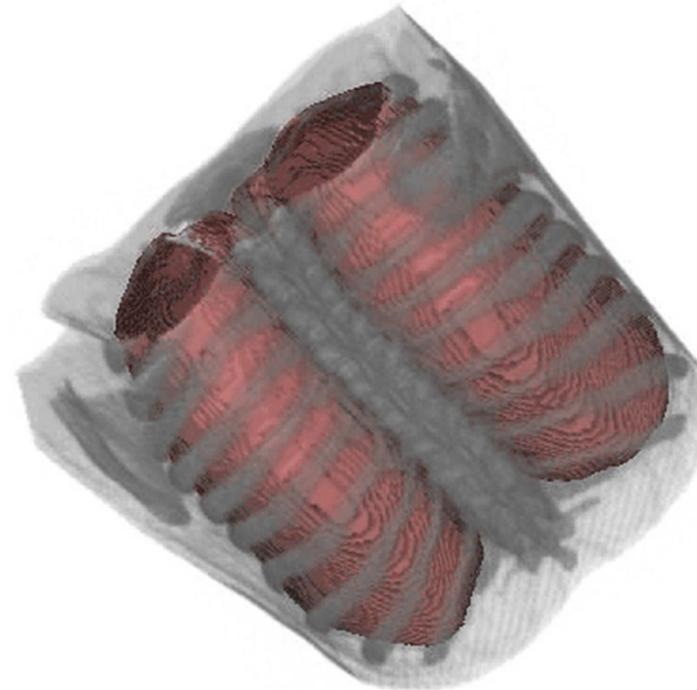
## Our Work: Globally Minimal Surfaces



**Appleton et al, PAMI**

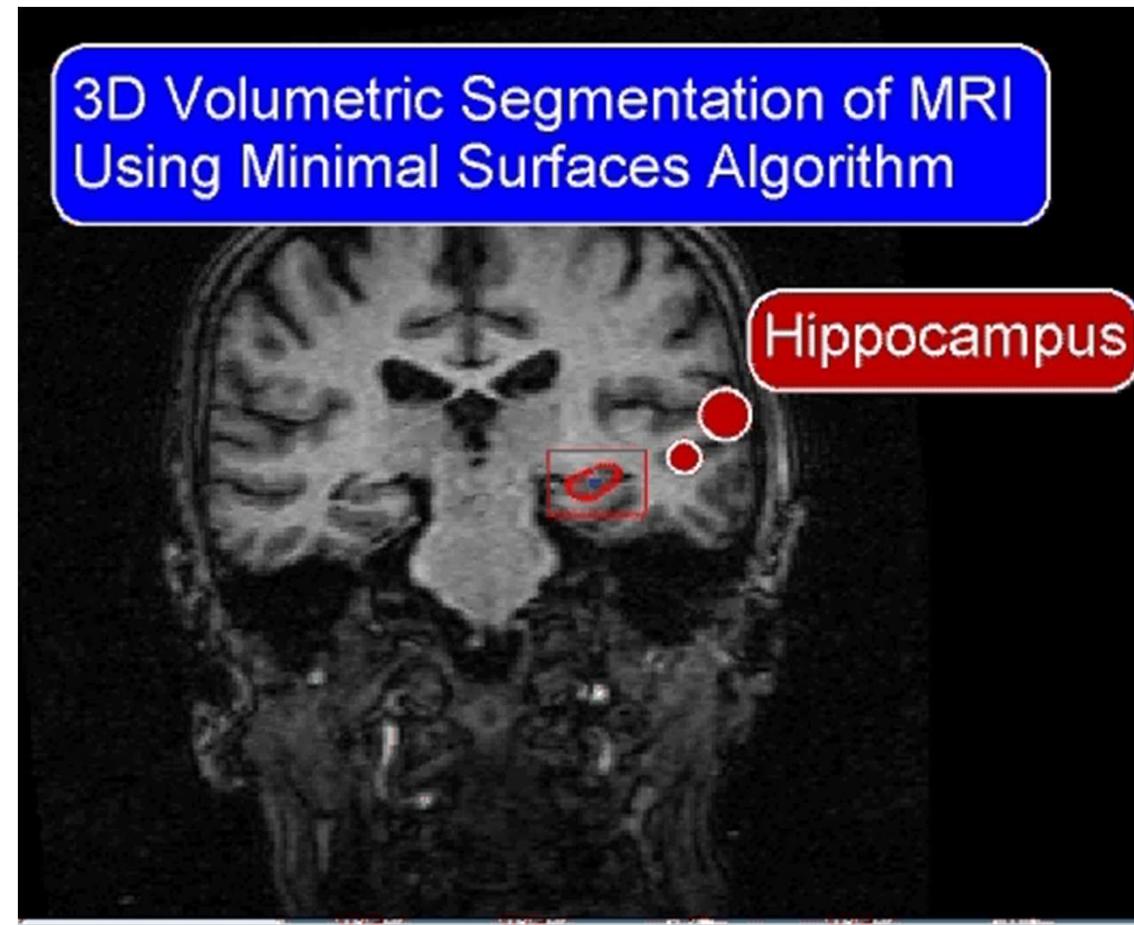


# Lung Segmentation



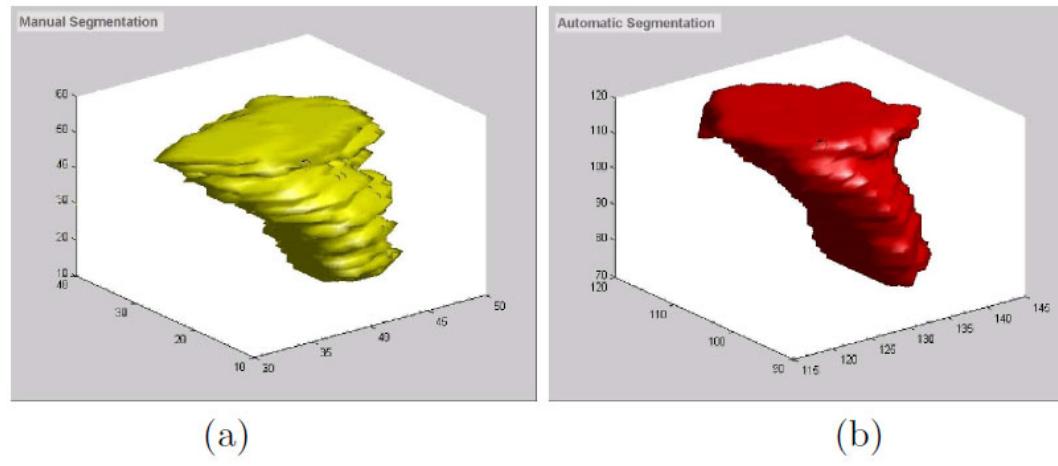


## Hippocampus Segmentation





## Hippocampus Segmentation



**Fig. 2.23.** Comparison of manual and automatic segmentation of the hippocampus in the human brain. Image (a) is a manual segmentation by a clinician which required about 2 hours of labelling and (b) is a fully-automated segmentation via GMS using multiple sources and sinks positioned by cross-validated training on labelled images which required just 2 minutes of computation (from [41]).



## Conclusion

What we've covered:

- Dynamic Programming
- Shortest Path Algorithms: Dijkstra and Fast Marching
- Evolving Contours and Surfaces: Level Sets
- Applications



## Questions?

?



# Reconstruction and Video Enhancement

**Courtesy Richard Hartley, ACCV Keynote**



**Courtesy Marc Pollefey**



**Courtesy Marc Pollefeys**



**Courtesy Marc Pollefey**



**Courtesy Marc Pollefeys**



**Courtesy Marc Pollefeys**



## Steps of reconstruction:

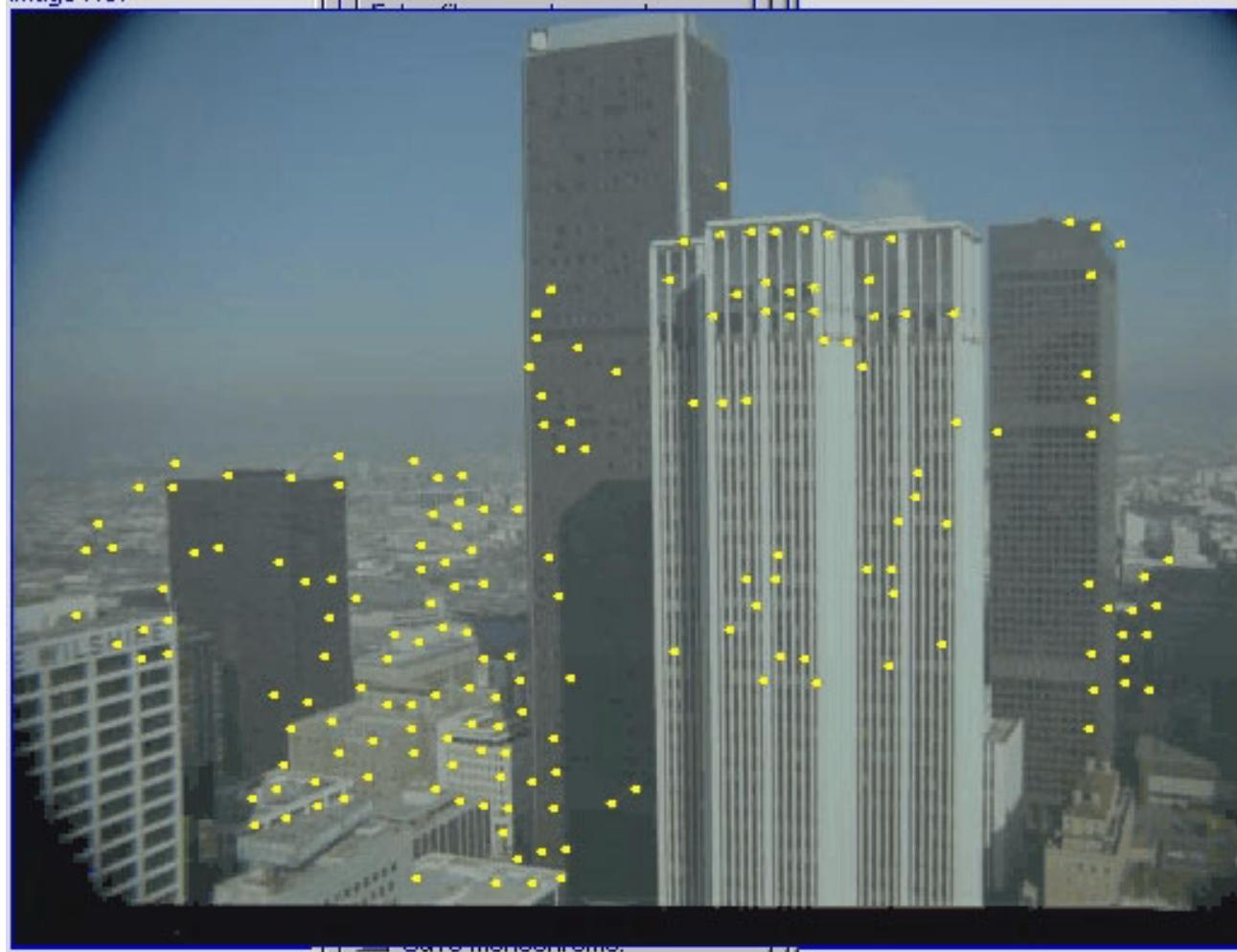
1. Tracking of points in the video
2. Weeding out bad tracks
3. Projective reconstruction
4. Self calibration / Euclidean reconstruction
5. Model building



## Tracking

**First step is to weed the points**

1. Produces lots of bad matches
2. Must be weeded out using a weeding program





## Homogeneous coordinates

- A point  $(x, y)$  on a plane is represented by a 3-vector  $(x, y, 1)^\top$ .
- All multiples of  $(x, y, 1)^\top$  represent the same point. Thus

$$(x, y, 1)^\top = (2x, 2y, 2)^\top = \dots = (kx, ky, k)^\top$$

for any non-zero  $k$ .

- An arbitrary 3-vector  $(x, y, w)^\top$  represents the point  $(x/w, y/w)$
- Points with  $w = 0$  represent points on the “plane at infinity”



## Projective space, $\mathcal{P}^2$ and $\mathcal{P}^3$

- The set of all (equivalence classes of ) homogeneous  $(n+1)$ -vectors is known as “projective  $n$ -space”,  $\mathcal{P}^n$ .
- Points with last coordinate equal to zero are called “points at infinity”.
- Thus,  $\mathcal{P}^2 = \mathbb{R}^2 \cup \{\text{line at infinity}\}$
- $\mathcal{P}^3 = \mathbb{R}^3 \cup \{\text{plane at infinity}\}$



## Homography between planes

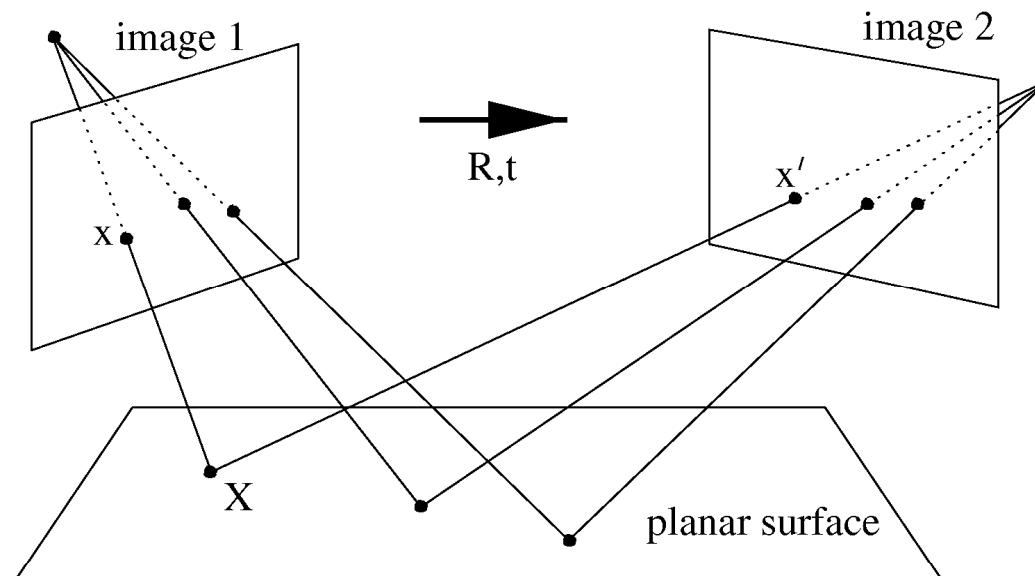
**Definition** Any point-to-point mapping from  $\mathcal{P}^2$  to  $\mathcal{P}^2$  that takes lines to lines is called a homography.

Other names : collineation, projectivity, projective transform.



## How do 2D homographies arise

- Between a plane in the world and its image with a perspective camera.
- Between two images of a plane.
- Between two images of the world taken with a rotating (but not moving) camera.

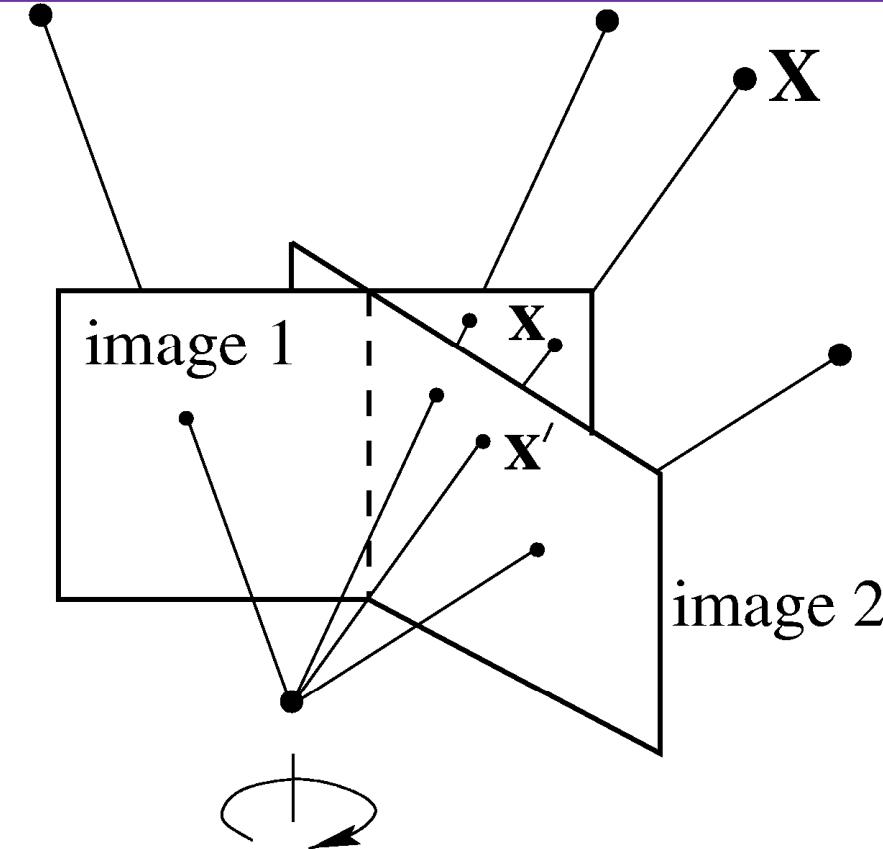


Two images of a plane are related by a homography

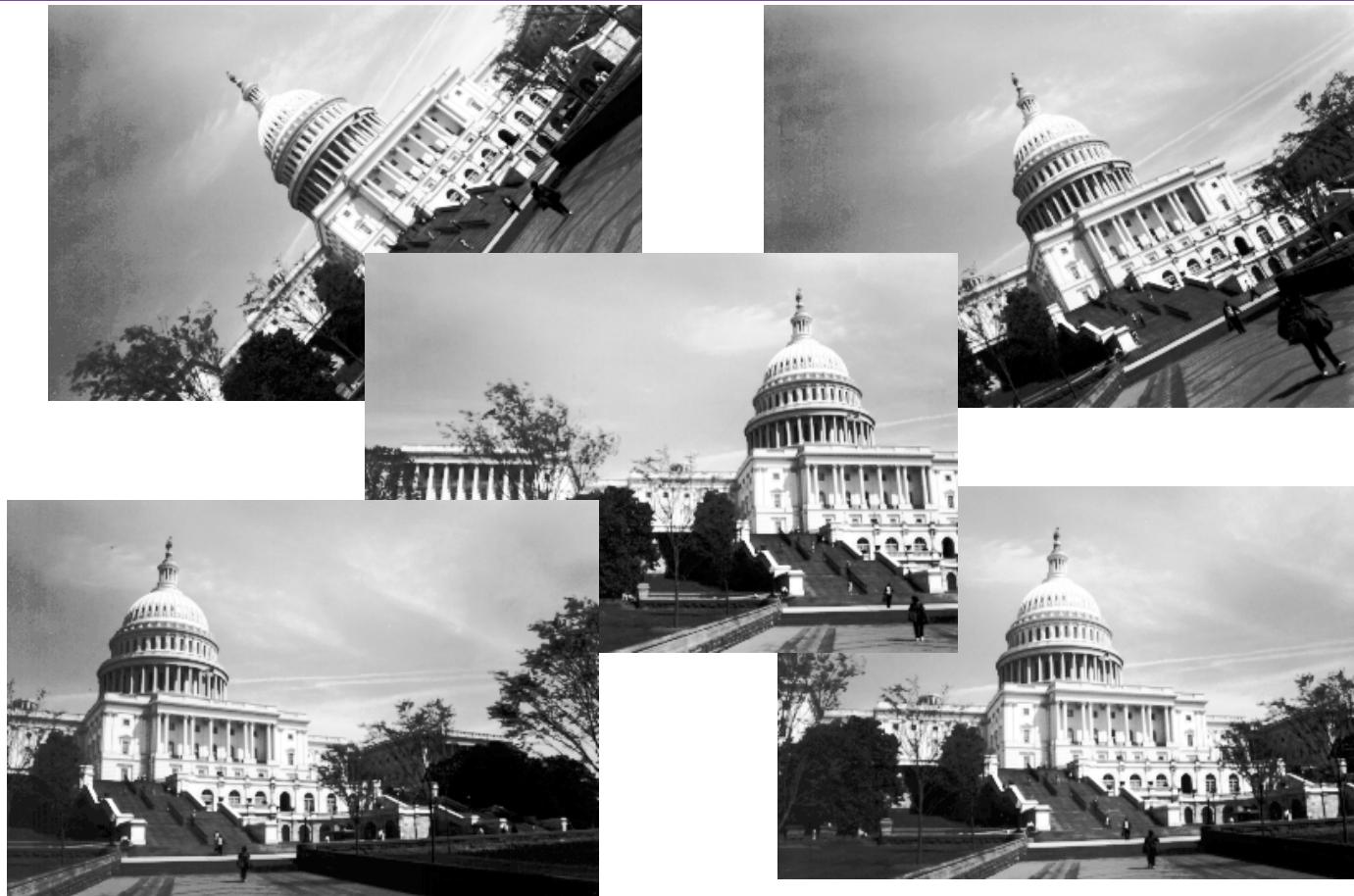


## Images of floor, related by homographies





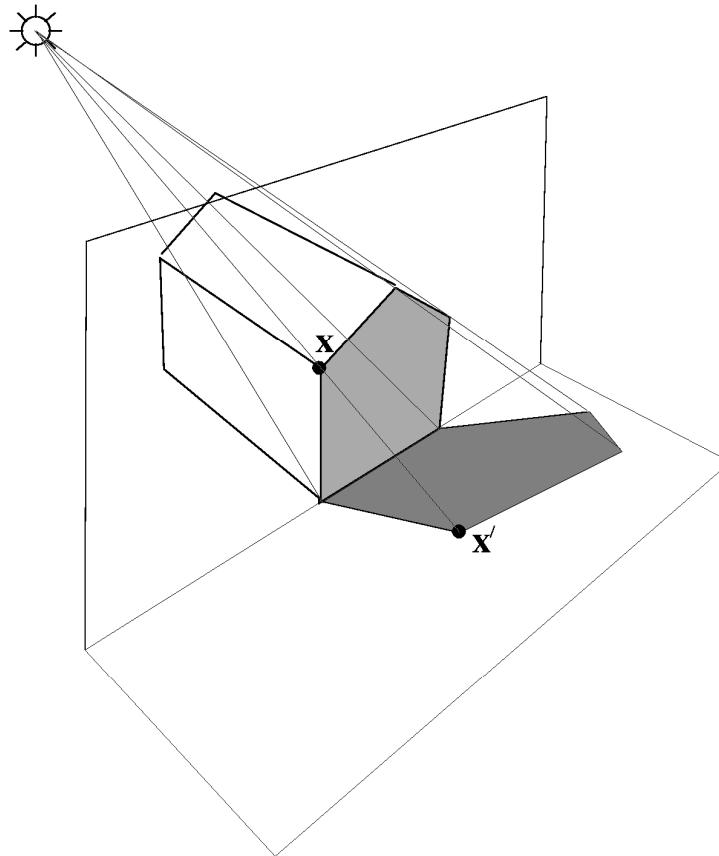
Images taken with a rotating camera.



**Image taken from the same location**



## Mosaicking by homographic warping



Images of a planar object and its shadow.



## Algebraic formulation of a homography

- Points in the plane are represented by homogeneous coordinates  $\mathbf{x} = (x, y, w)^\top$
- Homography is represented by a  $3 \times 3$  matrix  $\mathbf{H}$

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{23} & h_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

- Thus, homography is just a linear transformation on homogeneous coordinates.



## Homography in non-homogeneous coordinates

- In non-homogeneous coordinates, homography is written as follows:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

- In homogeneous coordinates we write:

$$\mathbf{x}' = \mathbf{H}\mathbf{x}$$



## Camera matrix

- Projection from  $\mathcal{P}^3$  (3D world) to  $\mathcal{P}^2$  (image).
- Expressed as a linear map in homogeneous coordinates:

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

- $\mathbf{P}$  is a  $3 \times 4$  matrix – the “camera matrix” .



## Computation of 2D homography

- Homography  $H$  has 8 degrees of freedom (9 entries, but scale irrelevant).
- Each point provides two constraints on  $H$ .
- Thus 4 point matches are required to compute  $H$ .
- With more than 8 point matches, least-squares techniques are used.
- Computation of homographies is reliable and easy.



## Tracking planes – original video





## Stabilization via homographies





## Image sequence of Wilshire Boulevard, LA

Courtesy Oxford Visual Geometry group



**Tracking a plane section of the image**



## Stabilization on the plane



## Image enhancement using homographies

Courtesy Oxford Visual Geometry group



## 3D reconstruction

- Reconstruct a scene given point tracks (correspondences) in the images.
- Two-step reconstruction, ‘stratified reconstruction’ :
  1. Projective reconstruction (uncalibrated cameras)
  2. Euclidean reconstruction (autocalibration)



## Mathematical formulation

- Given correspondences  $\mathbf{x}_{ij}$  ( $i$ -th point in  $j$ -th image)
- Compute camera matrices  $\mathbf{P}_j$  and 3D-points  $\mathbf{x}_i$  such that

$$\mathbf{x}_{ij} = \mathbf{P}_j \mathbf{x}_i$$



## Projective Reconstruction using Planes

**Two methods:**

1. Carlsson/Rother – ICCV 2001. Method solves for 3D points and cameras all together.
2. Hartley/Kaucic – Hartley-Zisserman (CUP 2000), ICCV 2001. Solves only for the cameras. 3D points are computed separately.



**Identify a planar section of the scene  
and compute homographies**



## What do the plane-plane homographies tell us

- Knowledge of homographies between the images means we know the first 3x3 part of the camera matrices.
- Remains only to know the last column of the camera matrices.

$$P = \left[ \begin{array}{c|c} M & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \end{array} \right]$$



## How plane homographies help reconstruction

We may suppose:

- Plane inducing the homographies is the plane at infinity, with points  $\mathbf{x}_j = (x_j, y_j, z_j, 0)^\top$ .
- Camera matrices are  $P_i = [\mathbf{M}_i | \mathbf{t}_i]$
- First camera is  $P_0 = [I | 0]$



Then

$$\begin{aligned}\mathbf{x}_{0j} &= \mathbb{M}_0(x_j, y_j, z_j)^\top = (x_j, y_j, z_j)^\top \\ \mathbf{x}_{ij} &= \mathbb{M}_i(x_j, y_j, z_j)^\top \\ &= \mathbb{M}_i \mathbf{x}_{0j}\end{aligned}$$

$\mathbb{M}_i = H_{0i}$  is the homography from image 0 to image  $i$ .



## Carlsson-Rother method of projective reconstruction

Suppose point  $x = (x, y, z, 1)^\top$  is not on the plane at infinity (the one inducing the homographies).

Point projection:

$$\lambda u = Px = [M|t]x = [M|t] \begin{pmatrix} \tilde{x} \\ 1 \end{pmatrix}$$

More precisely

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} m_1^\top & t_1 \\ m_2^\top & t_2 \\ m_3^\top & t_3 \end{bmatrix} \begin{pmatrix} \tilde{x} \\ 1 \end{pmatrix} = \begin{pmatrix} m_1^\top \tilde{x} + t_1 \\ m_2^\top \tilde{x} + t_2 \\ m_3^\top \tilde{x} + t_3 \end{pmatrix}$$



Equations can be written as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \times \begin{pmatrix} \mathbf{m}_1^\top \tilde{\mathbf{x}} + t_1 \\ \mathbf{m}_2^\top \tilde{\mathbf{x}} + t_2 \\ \mathbf{m}_3^\top \tilde{\mathbf{x}} + t_3 \end{pmatrix} = 0$$

This provides two equations

$$\begin{aligned} u(\mathbf{m}_3^\top \tilde{\mathbf{x}} + t_3) - (\mathbf{m}_1^\top \tilde{\mathbf{x}} + t_1) &= 0 \\ v(\mathbf{m}_3^\top \tilde{\mathbf{x}} + t_3) - (\mathbf{m}_2^\top \tilde{\mathbf{x}} + t_2) &= 0 \end{aligned}$$



- Equations are linear in unknowns  $\tilde{\mathbf{x}}_j = (x_j, y_j, z_j)^\top$  and  $\mathbf{t}_i = (t_{i1}, t_{i2}, t_{i3})^\top$ .

$$\begin{bmatrix} u\mathbf{m}_3^\top - \mathbf{m}_1^\top & -1 & 0 & u \\ v\mathbf{m}_3^\top - \mathbf{m}_2^\top & 0 & -1 & v \end{bmatrix} \begin{pmatrix} \tilde{\mathbf{x}} \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0$$

- With  $m$  views involving  $n$  points there are equations in  $3n + 3m$  unknowns.
- Solve linearly for structure and motion
- Unlike Tomasi-Kanade factorization, we do not need all points visible in all views.



## Method of Hartley-Zisserman/Kaucic

- Uses multilinear relationships to solve for cameras.
- Eliminates the structure  $\mathbf{x}_j = (x_i, y_i, z_i, t_i)^\top$  leaving only the motion parameters  $t_i$  to solve for.



Projection equation:  $\lambda \mathbf{u} = \mathbf{P}\mathbf{X}$  may be written:

$$(\mathbf{P} - \lambda \mathbf{u}) \begin{pmatrix} \mathbf{X} \\ -\lambda \end{pmatrix} = 0$$

Suppose a correspondence  $\mathbf{u}_1 \leftrightarrow \mathbf{u}_2 \leftrightarrow \dots \leftrightarrow \mathbf{u}_k$  across  $k$  views.

Equations may be written as

$$\begin{bmatrix} [\mathbf{M}_1 \mathbf{t}_1] & \mathbf{u}_1 & & & \\ [\mathbf{M}_2 \mathbf{t}_2] & & \mathbf{u}_2 & & \\ \vdots & & & \ddots & \\ [\mathbf{M}_k \mathbf{t}_k] & & & & \mathbf{u}_k \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ -\lambda_1 \\ -\lambda_2 \\ \vdots \\ -\lambda_k \end{pmatrix} = 0$$



- Since this has a solution, matrix columns are linearly dependent.
- Whole matrix is known, except  $t_i$ .
- Column of  $t_i$  must be in span of the other columns.
- Provides  $2k - 3$  linear equations in the unknown entries  $t_i$ .
- Solve set of linear equations to find the  $t_i$ .

Similar to some of the ideas of Yi Ma – Multi-view Matrix (this conference).



## Finding other planes in the Scene

**Problem** Given camera matrices, find a plane that induces image correspondences between images.

- Correspondences  $x \leftrightarrow x'$  between two images, due to plane homography.
- Known camera matrices  $P_0 = [I|0]$ , and  $P_1 = [M|t]$
- Find the plane.



## One approach :

- Reconstruct the 3D points and fit to a plane.

Not a good idea:

- Plane-fitting in a projective space is difficult (point-plane distance has no meaning).
- Reconstructed points may be well off the plane.



**Alternative solution:** Work directly with image coordinates.

**Theorem :** *Homography from image  $P_0 = [I|0]$  to image  $P_1 = [M|t]$  due to a plane  $ax + by + cz + 1 = 0$  is given by*

$$M - tv^\top$$

where  $v = (a, b, c)^\top$ .

**Calculate :**

$$\begin{aligned}\lambda x' &= Hx \\ &= (M - tv^\top)x \\ &= Mx - tv^\top x \\ &= Mx - (tx^\top)v\end{aligned}$$



## Equations:

$$\lambda \mathbf{x}' = \mathbb{M}\mathbf{x} - (\mathbf{t}\mathbf{x}^\top)\mathbf{v}$$

- Everything in this equation known except for  $\mathbf{v}$ .
- Eliminate the scale factor to get two linear equations in  $\mathbf{v}$ .



## Bundle-adjusting homographies

- Bundle-adjustment is a way of throwing all known data into a single minimization problem.
- Parametrize unknown data (e.g. a homography) in terms of a set of parameters.
- Define a cost function that depends on the measured data and the unknown parameters.
- Minimize the cost function with respect to the parameters.



## Advantages of bundle-adjustment

- Extremely powerful technique with a lot of known theory.
- Geometrically / statistically correct cost functions may be minimized.
- Maximum-likelihood estimate assuming Gaussian noise in measurements.
- Handles very general constraints.
- Rumours of its slowness are greatly exaggerated



## One-sided geometric distance.

- Find homography  $H$  mapping  $\mathbf{x}_i$  to  $\mathbf{x}'_i$ .
- Parameters are the entries of a homography matrix,  $H$ .
- Measured data are the point correspondences  $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$  between the two images
- Cost function is

$$\sum_i d(\mathbf{x}'_i, H\mathbf{x}_i)^2$$

where  $d(., .)$  is the image distance between pair of points.



## Homographies defined by a plane with known cameras.

- Parameters are the coordinates  $\mathbf{v}$  of the plane.
- Homographies given by formula

$$\mathbf{H} = \mathbf{M} - \mathbf{t}\mathbf{v}^\top$$

- Measurements are the correspondences  $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$  between images  $\rho(i)$  and  $\sigma(i)$ .
- Cost function is

$$\sum_i d(\mathbf{x}'_i, \mathbf{H}_{\rho(i)\sigma(i)} \mathbf{x}_i)^2$$



## Projective reconstruction given homographies

- Assume camera matrices  $P_i = [M_i | t_i]$ , with the  $M_i$  known.
- Parameters are the values  $t_i$  for each camera, plus 3D points  $X_j$ .
- Measurements are the image coordinates  $u_{ij}$  of the 3D points.
- Cost is

$$\sum_{i,j} d(P_i X_j, u_{ij})^2$$



## Bundle-adjustment – Cost considerations

- Major cost consideration is the number of parameters.
- Essentially cubic cost in the number of parameters.
- **But** Sparse techniques cut the cost down by orders of magnitude.



## Homographies between many images.

- Given point correspondences  $\mathbf{x}_{ij}$ , the image of an  $i$ -th point in the  $j$ -th image.
- Find the homographies  $\mathbb{H}_{jk}$  between images,  $(j, k)$  such that

$$\mathbb{H}_{jk} \mathbf{x}_{ij} = \mathbf{x}_{ik}$$

- Homographies are members of a group – must obey group composition rule.

$$\mathbb{H}_{rt} = \mathbb{H}_{st}\mathbb{H}_{rs}$$

- Assign a matrix  $\mathbb{M}_j = H_{0j}$  to each image.  
Require

$$\mathbb{H}_{jk} = \mathbb{M}_k \mathbb{M}_j^{-1}$$



## Full bundle-adjustment

- Parameters are the entries of all the  $\mathbf{M}_j$ , plus “virtual” points  $\bar{\mathbf{x}}_i$
- Cost function is

$$\sum_{ij} d(\mathbf{M}_j \bar{\mathbf{x}}_i, \mathbf{x}_{ij})^2$$

- Total number of parameters is  $8(m-1) + 3n$  for  $m$  images of  $n$  points.



## Philosophy

This bundle-adjustment method is motivated by assumption :

- there is a “true” point in the world (represented by  $\bar{x}_i$ ) and each  $x_{ij}$  is an (inexact) image of this point.



## Remarks on trackers

- Most trackers simply track points from one image to the next (e.g Kanade-Lucas tracker).
- Usually do not work by finding a world point in images.
- A track  $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \dots \leftrightarrow \mathbf{x}^{(k)}$  is usually built up from pairwise matches  $\mathbf{x}^{(j)} \leftrightarrow \mathbf{x}^{(j+1)}$ .
- **For homographies** there is no more information in a track  $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \dots \leftrightarrow \mathbf{x}^{(k)}$  than in the pairwise matches  $\mathbf{x}^{(j)} \rightarrow \mathbf{x}^{(j+1)}$ .
- This is **not** true for non-planar points, where matches between consecutive views is not sufficient to obtain structure.



## Bundle adjustment for homographies

- Given point correspondences  $\mathbf{x}_i \rightarrow \mathbf{x}'_i$ , where  $\mathbf{x}_i$  is in image  $\rho(i)$  and  $\mathbf{x}'_i$  is in image  $\sigma(i)$
- Find the homographies  $\mathbf{H}_{jk}$  such that
  1.  $\mathbf{H}_{\rho(i)\sigma(i)} \mathbf{x}_i \approx \mathbf{x}'_i$
  2. and  $\mathbf{H}_{jk} = \mathbf{M}_k \mathbf{M}_j^{-1}$

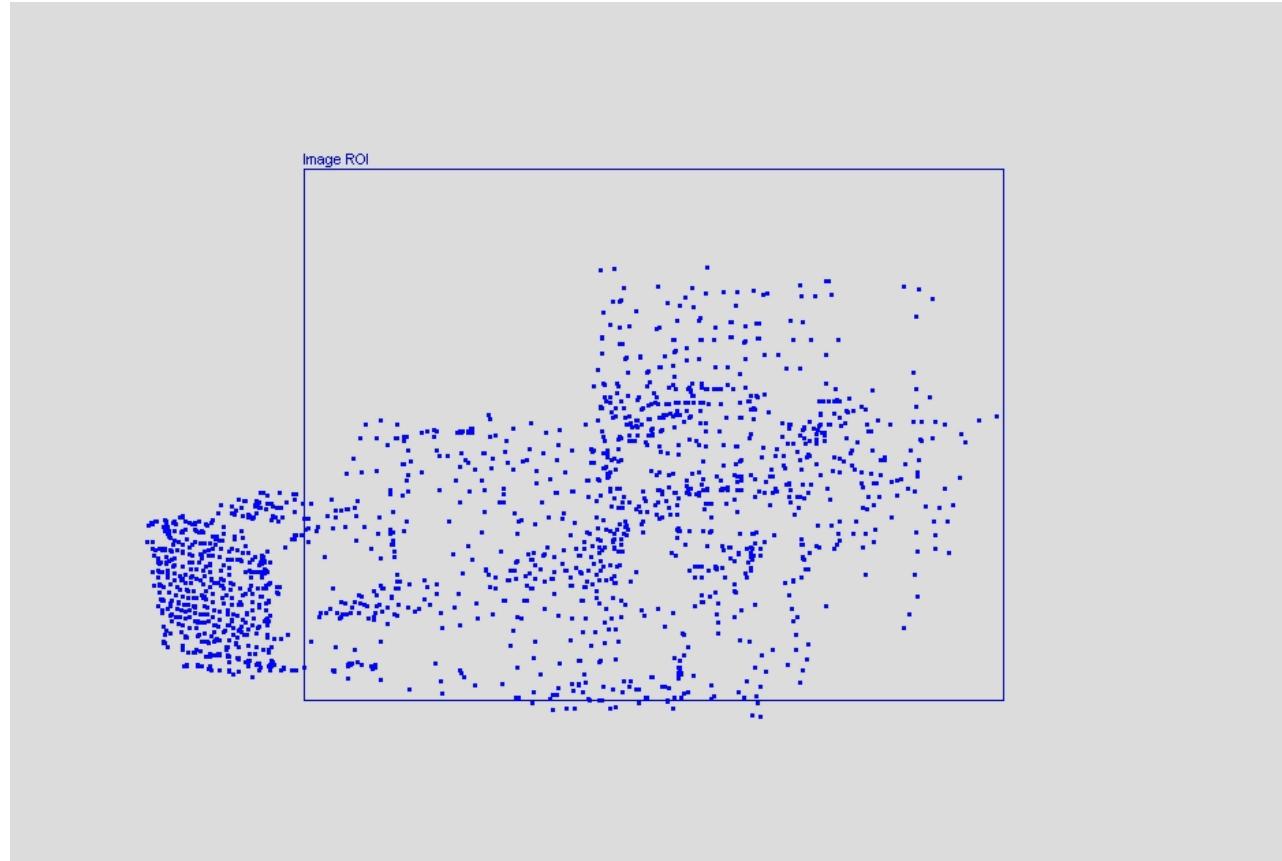


## One-sided bundle adjustment

- Parameters are the entries of all the  $\mathbb{M}_j$
- Cost function is

$$\sum_i d(\mathbb{H}_{\sigma(i)} \rho(i) \mathbf{x}_i, \mathbf{x}'_i)^2 = \sum_i d(\mathbb{M}_{\sigma(i)}^{-1} \mathbb{M}_{\rho(i)} \mathbf{x}_i, \mathbf{x}'_i)^2$$

- Total number of parameters is  $8(m - 1)$ , where  $m$  is the number of views.

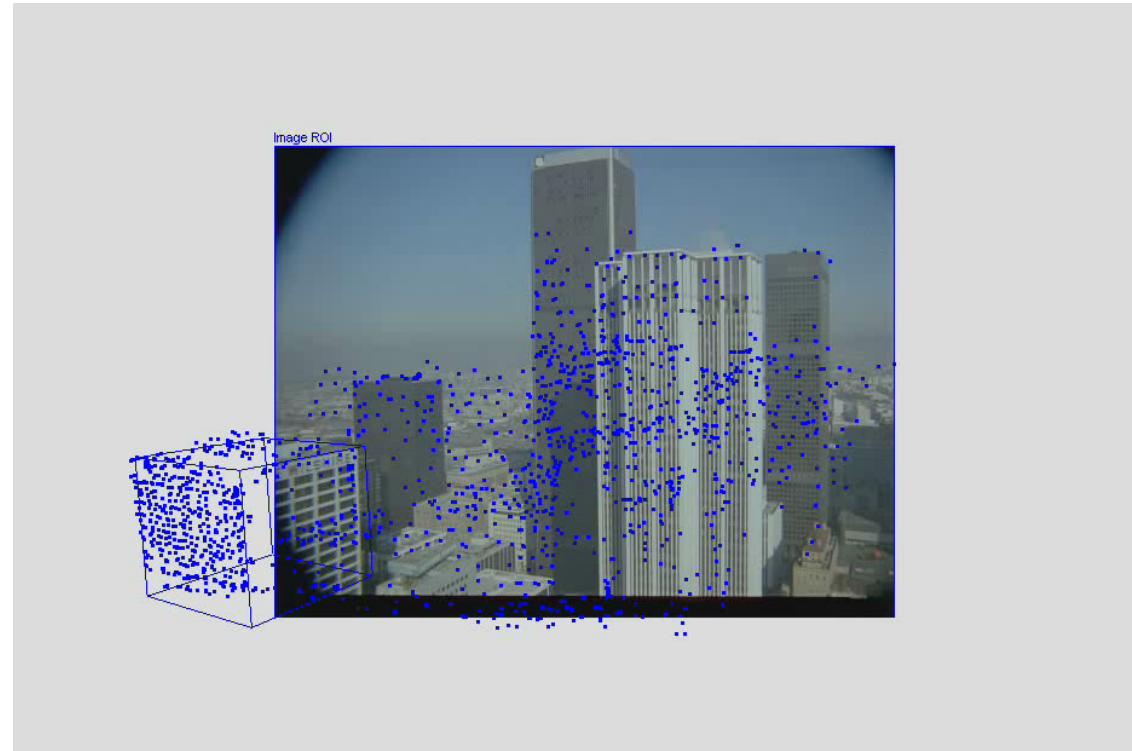


## Results of reconstruction of points

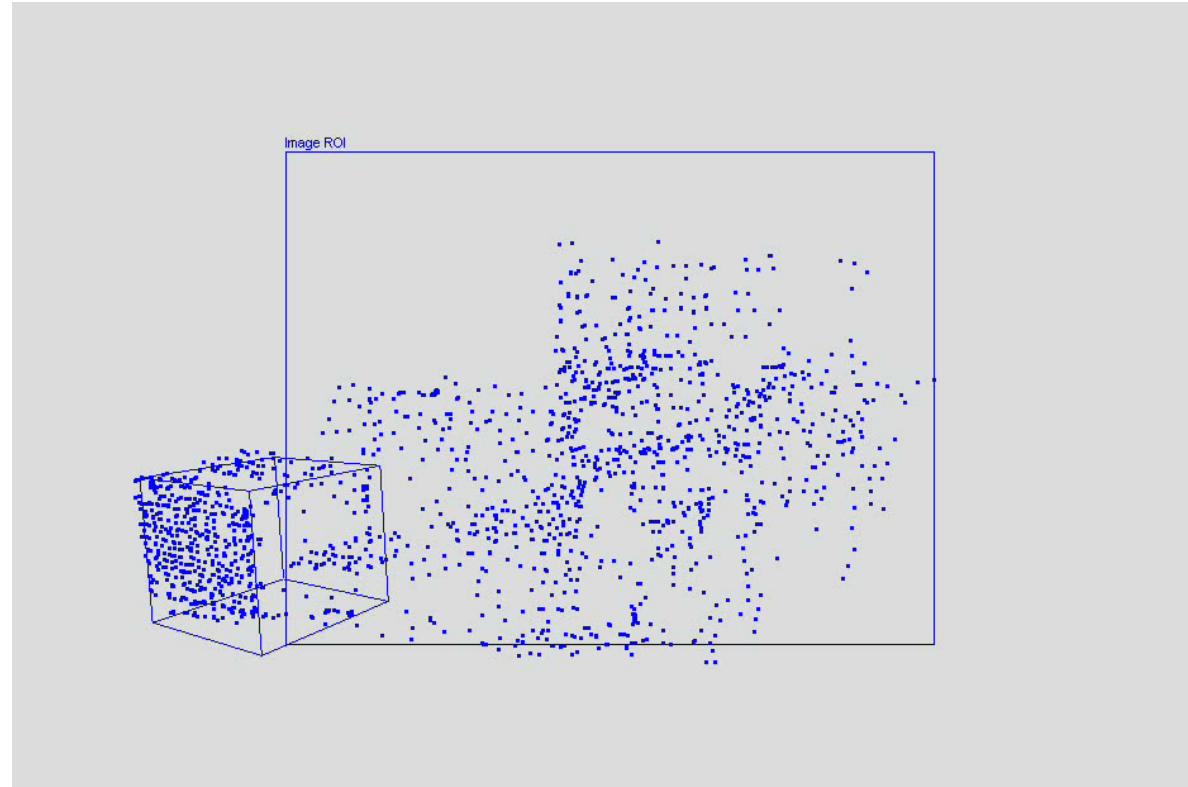


## **Self-calibration and Euclidean reconstruction**

- Calibration of camera based only on internal evidence in the video.
- Plane-based reconstruction possible (Triggs - ECCV98)
- Correct geometry of scene results.



## Rectangular reference frame



## Rectangular reference frame



## Enhanced video

Courtesy Oxford Visual Geometry group



## Interlude – Conformal point



## Measurement of angles in plane from their images

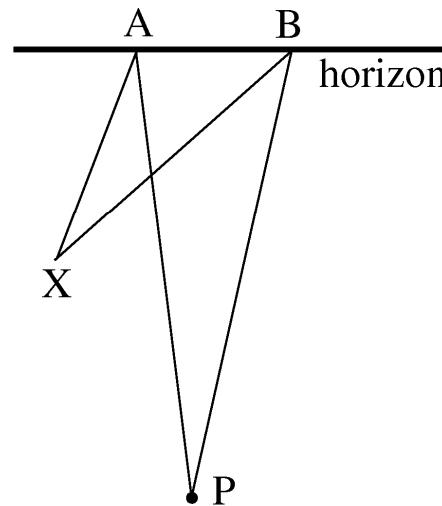
Camera model used:

- partially calibrated – pixels assumed square
- Focal length and principal point not known.
- Method relies on identifying the horizon line and the “conformal point”.



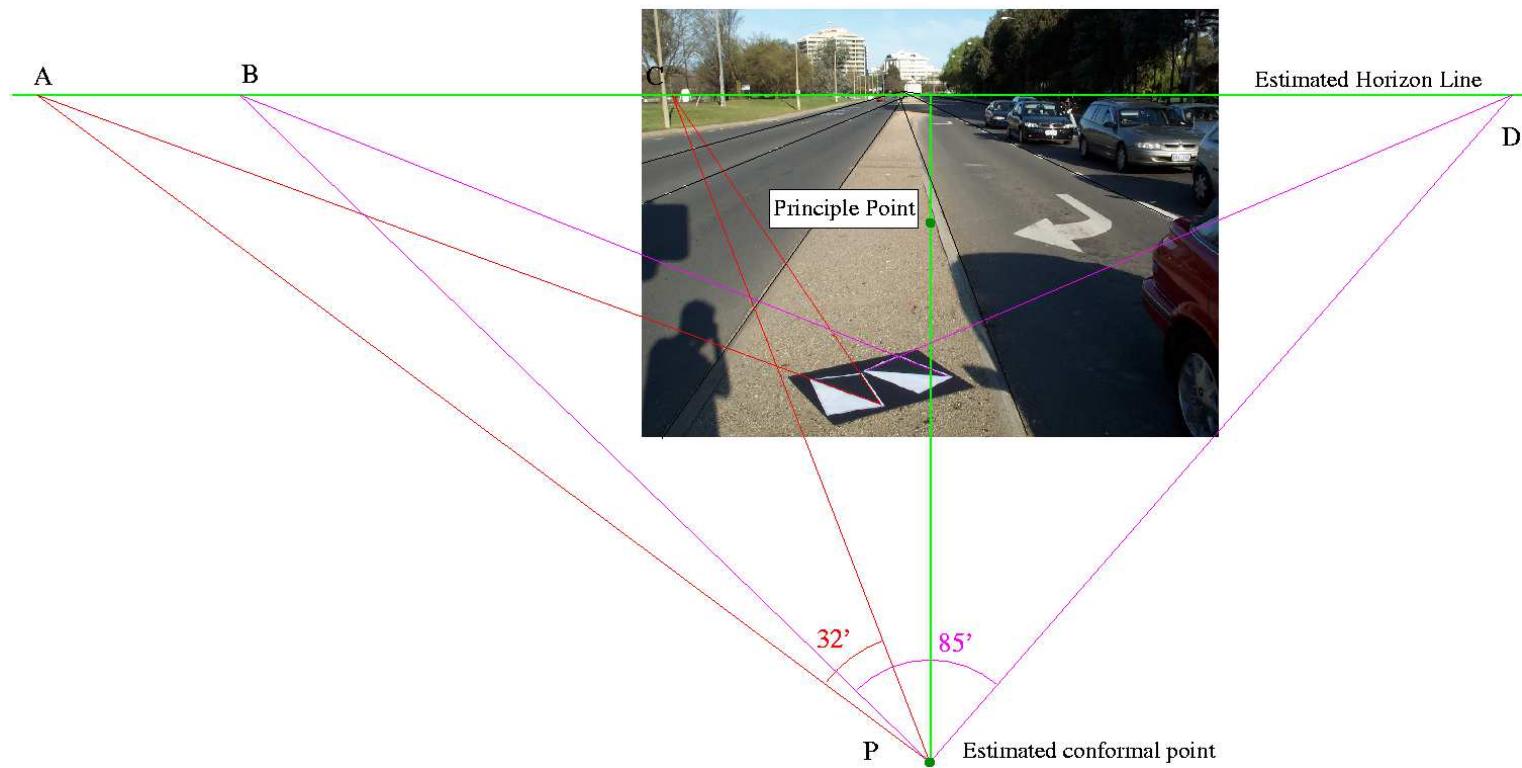
## Angle measurement

- Extend lines to the horizon.
- Connect back to the conformal point



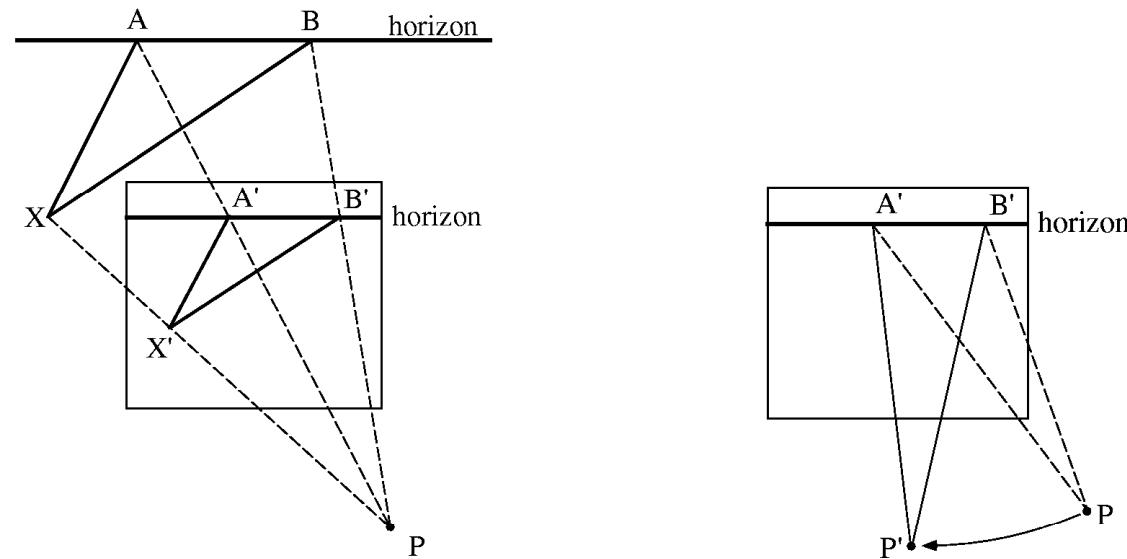


# Example

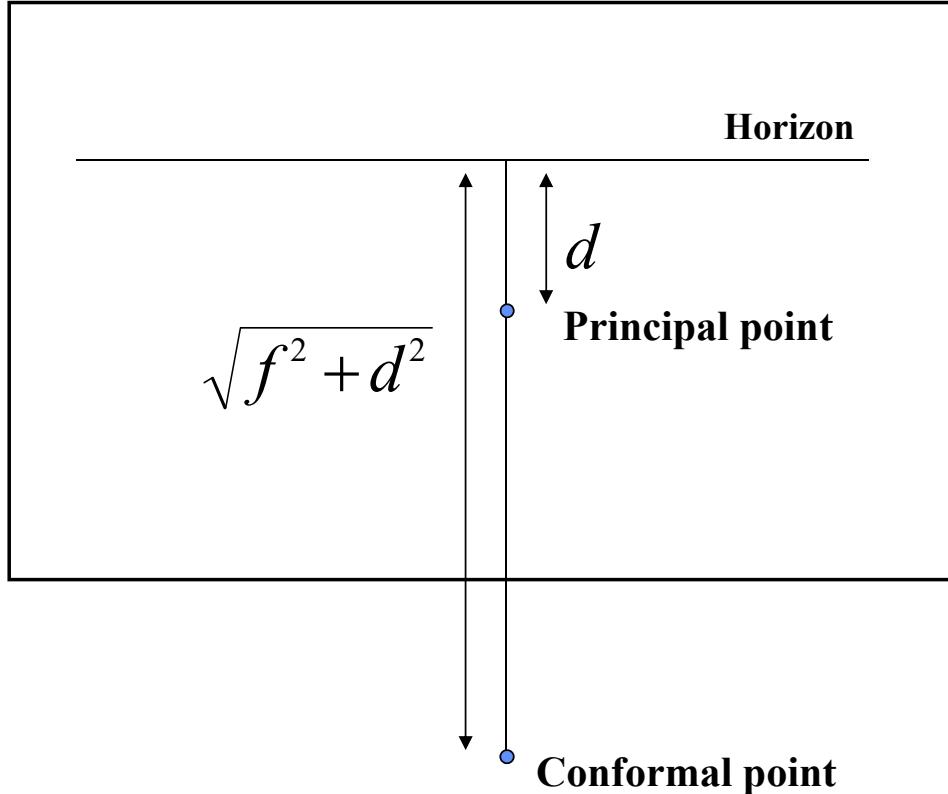




## Proof



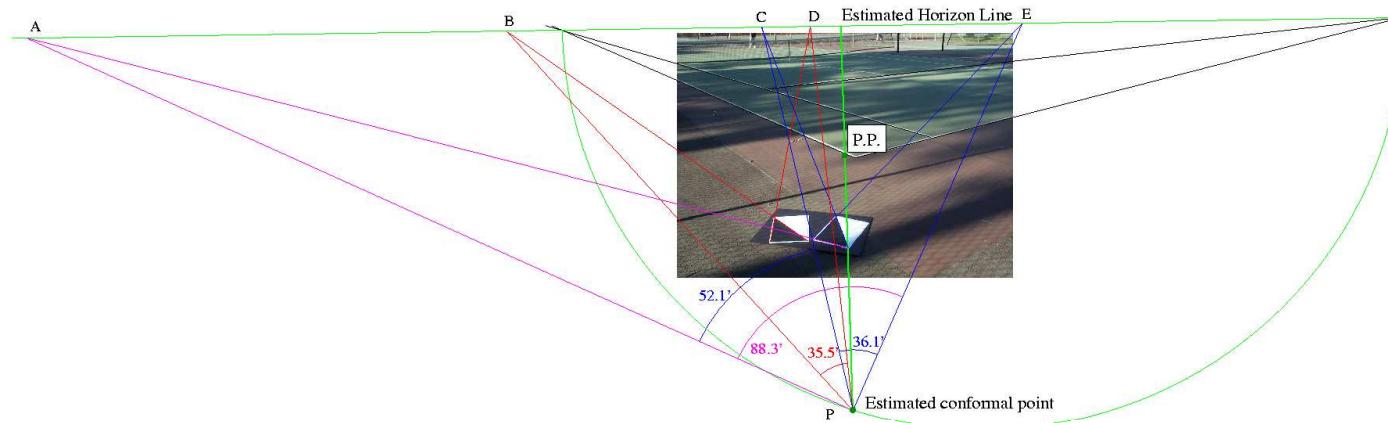
$$\widehat{AXB} = \widehat{APB} = \widehat{A'PB'} = \theta$$
$$\widehat{A'P'B'} = \widehat{A'PB'} = \theta$$



Conformal point lies a distance  $(f^2 + d^2)^{1/2}$  from the horizon.



## Example



Conformal point can be computed from observing a right-angle in the image.



## Methods of computing the horizon

- Interactively drawing it (once only)
- From constructing two vanishing points
- From planar motion : line passing through epipoles.
- Length ratios on lines in an image



## Computing the conformal point

- Directly from known focal length and principal point.
- From known principal point and an observed known angle
- From two known observed angles
- From three equal unknown angles.
- Planar motion : from the motion of three points from one frame to the next.



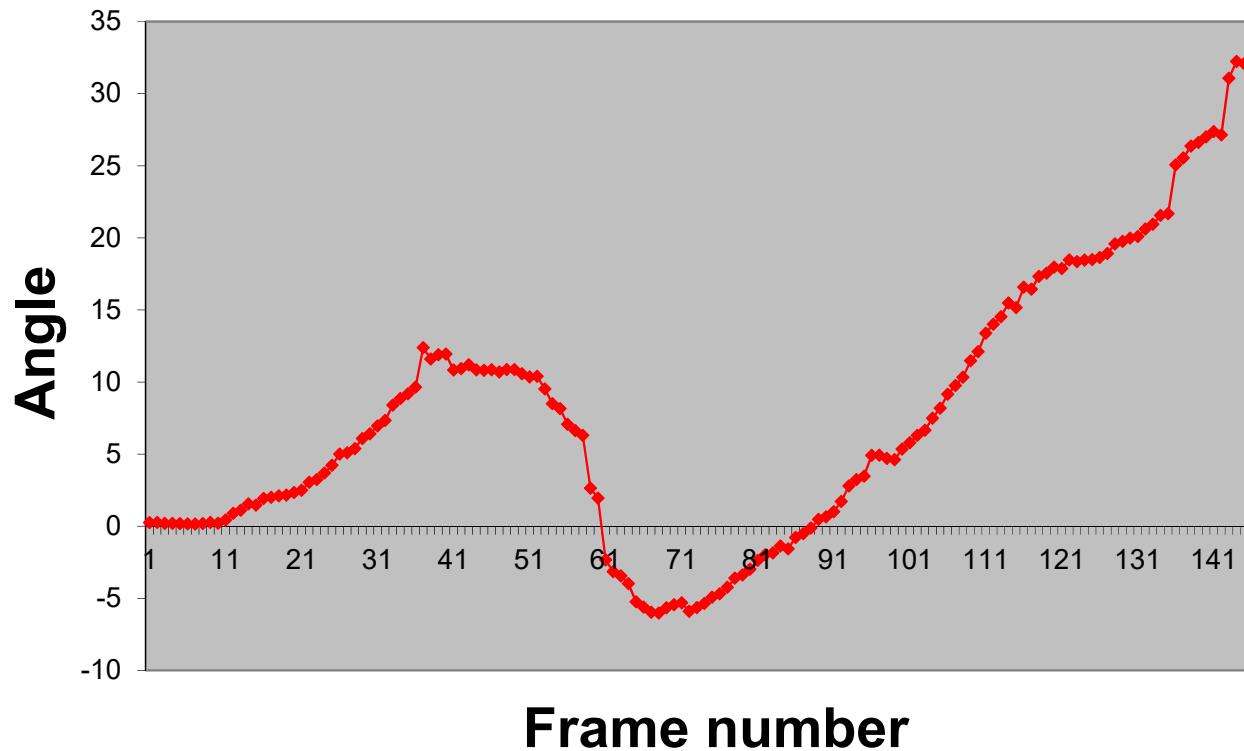
## Application: Odometry



**Tracked points**

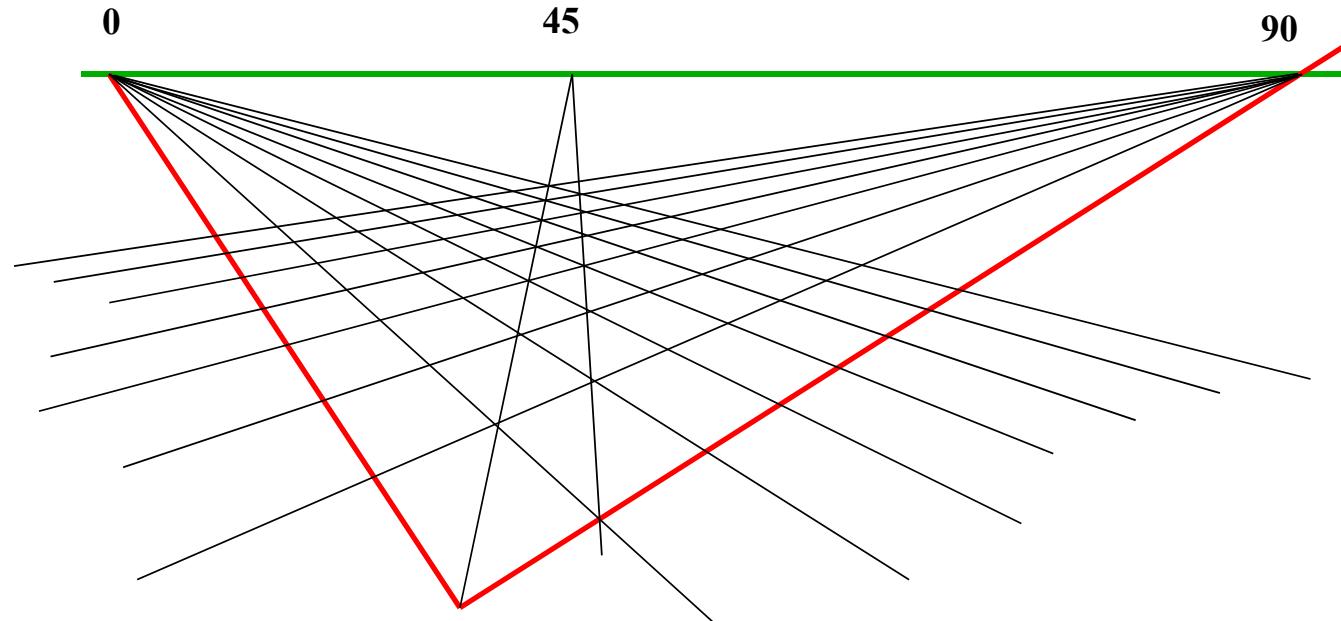


## Cumulative angle





## Perspective and Art-history



Drawing perspectively correct pavement grids.

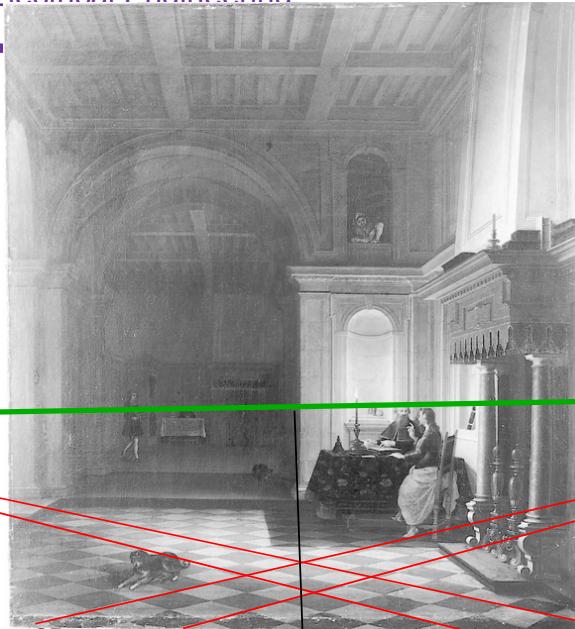


### Rules for perspective enunciated by Brunelleschi

**Method for drawing perspectively correct planes given by Leon Battista Alberti,  
“De Pictura” (1435)**

<http://www.mega.it/eng/egui/pers/lbalber.htm>





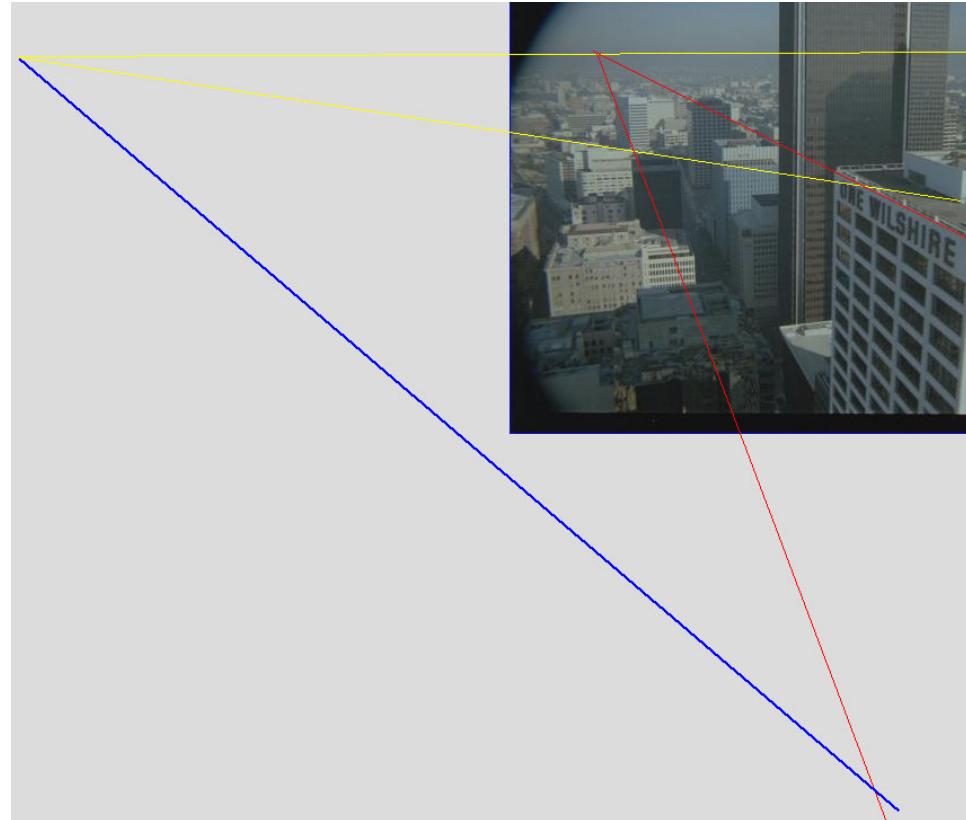
**Horizon**

**Conformal point**



## Application : Sun direction

- Measure angle between shadows and building lines



**Sun direction : 29.1 degrees**



# 2D3 Video



Courtesy 2D3



# **Introduction to Biometrics**

Brian Lovell



# Outline

- Definition of Biometrics
- Biometrics vs alternative methods of identification
- Types of Biometric technologies
  - Fingerprint, face, iris, voice
- Terminology
  - Enrolment, templates, matching, accuracy, false reject, false accept
- Description of technologies:
  - Fingerprint, Face, Iris, Voice
- Implementation issues
- Cultural and social issues
- Overview of private sector trends
- Government uses of biometrics



## Definition of Biometrics

1. Biometrics is the science and technology of measuring and analyzing biological data. (e.g. Biometrika Journal)
2. In information technology, **biometrics refers to technologies that measure and analyze human body characteristics**, such as DNA, fingerprints, eye retinas and irises, voice patterns, facial patterns and hand measurements, **for authentication purposes**.
  - <http://searchsecurity.techtarget.com/definition/biometrics>



# Early History

- In the early 1900's the law was changed so that punishment was increased for repeat offenders
- The need emerged to identify re-offenders
- Bertillon's *Identification anthropométrique* (1893)

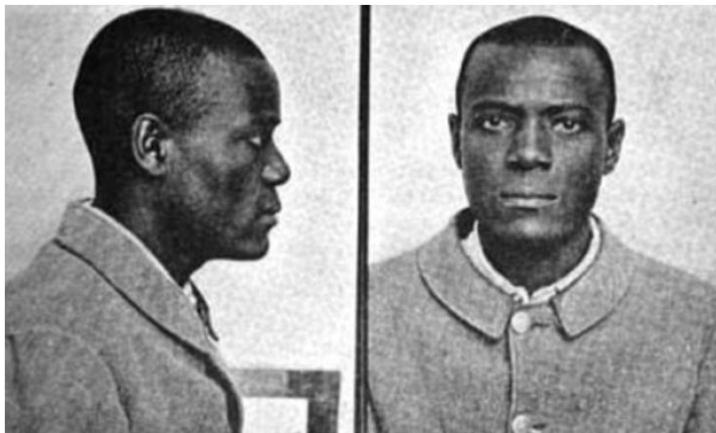




## The Strange Case of Will West and William West



Where there's a Will: In 1903 Will West arrived at Leavenworth Penitentiary in Kansas, where the records clerk was certain that he'd seen him before



Where there's another Will: The record clerk pulled out this file photo of William West, who looked almost identical to Will West

McClaughry, still convinced the man before him had already been to the prison, looked up his name in his filing system and found one William West – who looked identical to Will West in the photographs in every respect.

They even shared the same Bertillon measurements.

But Will West insisted to McClaughry that it was not him: ‘That’s my picture, but I don’t know where you got it, for I know I have never been here before.’

To McClaughry’s shock, he was absolutely right, too. William West was a different person altogether and in fact had been admitted to the prison two years previously for murder.

The case highlighted the flaws in the Bertillon method and it wasn’t long before the U.S authorities turned to fingerprinting.



# Biometrics vs Other Authentication Methods

- Artefacts (Keys, Cards)
  - Easily lost
  - Can be retrieved to pass on and revoke authority (car keys)
  - Fairly easily cancellable (change locks)
  - Easy to manage
- Secret Knowledge (PINs, Passwords)
  - Easily lost or forgotten
  - Once disclosed there is no way to retrieve them
  - Easily cancellable (password reset)
  - Simple to manage
- Biometrics
  - Rarely lost or forgotten
  - **Unique to each person**, cannot be transferred (no good for car)
  - Not cancellable in most cases
  - Complicated technology with significant false accepts and rejects

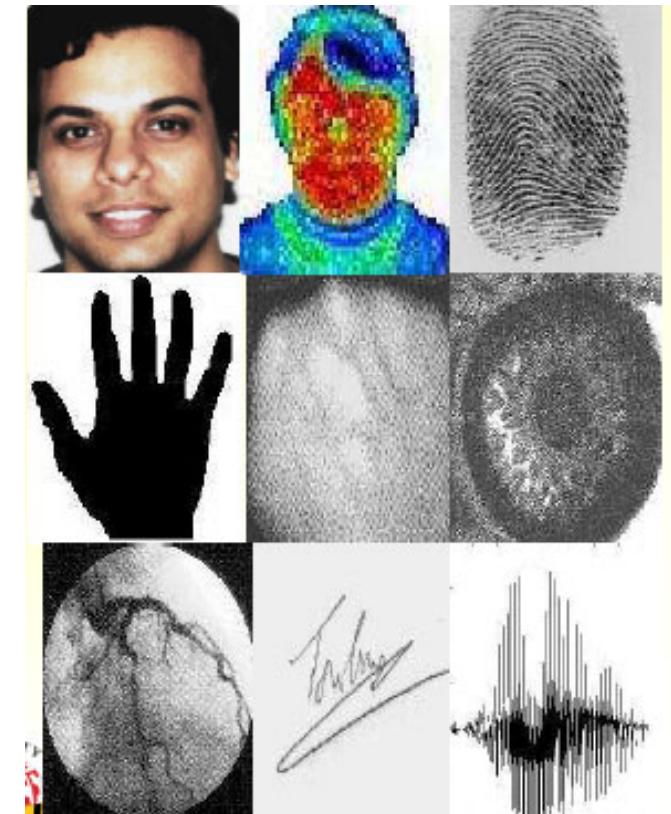
Three Factors

- Something you have
- Something you know
- Something you are



# Types of Biometric Technology

- **Fingerprint**
- **Face**
- **Iris**
- **Voice**
- Hand geometry
- Signature
- Retina
- Thermal
- Vein Patterns





# Terminology

- Enrolment
  - Adding a new person to the biometric collection or **gallery**
- Template
  - The mathematical representation of the biometric used for search and matching purposes. The template is generally much more compact than the raw representation (e.g., image, audio).
- Matching
  - Templates that are very similar to each other according to some distance measure (metric) are said to match
- Accuracy
  - **False reject** rate: Rate at which the true person is not accepted (big problem)
  - **False accept** rate: Rate at which imposters are accepted
- Gallery:
  - The collection of enrolled persons. Gallery size and makeup is important.
- Probe
  - The query image or template for matching



# Terminology

- Cooperative
  - Person knows about the biometric examination
- Non-cooperative
  - Person does not know the biometric is being captured
- Detectable
  - Person can detect biometric capture equipment
- Undetectable
  - Person cannot detect biometric equipment
    - e.g. due to long range capture
- Cancellable Biometric
  - Reduces risk if template is compromised. Template + encryption or distortion

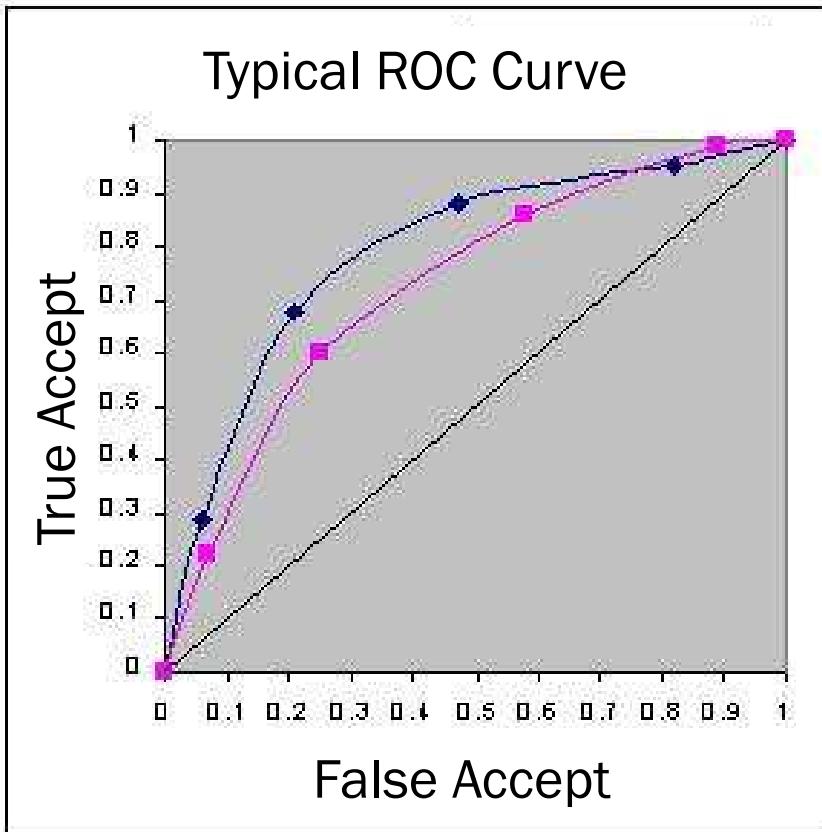


# Biometric Modes of Operation

- Verification or 1-to-1 matching
  - Is the person who they claim to be?
- Recognition, or 1-to-many matching
  - Who is this person? Are they in the gallery?
- Watch List or 1-to-some matching
  - Is this person on our watch list?
- Closed Set Matching
  - Everyone we encounter is in the gallery
  - Find best match
- Open Set Matching
  - Many people we see are not in the gallery
  - Best match is unlikely to be true identity
  - Need to estimate likelihood that person is in gallery



## Receiver Operating Characteristic (ROC) Curve



$$\text{True\_Accept} = 1 - \text{False\_Reject}$$

False Accept up  $\rightarrow$  False Reject down

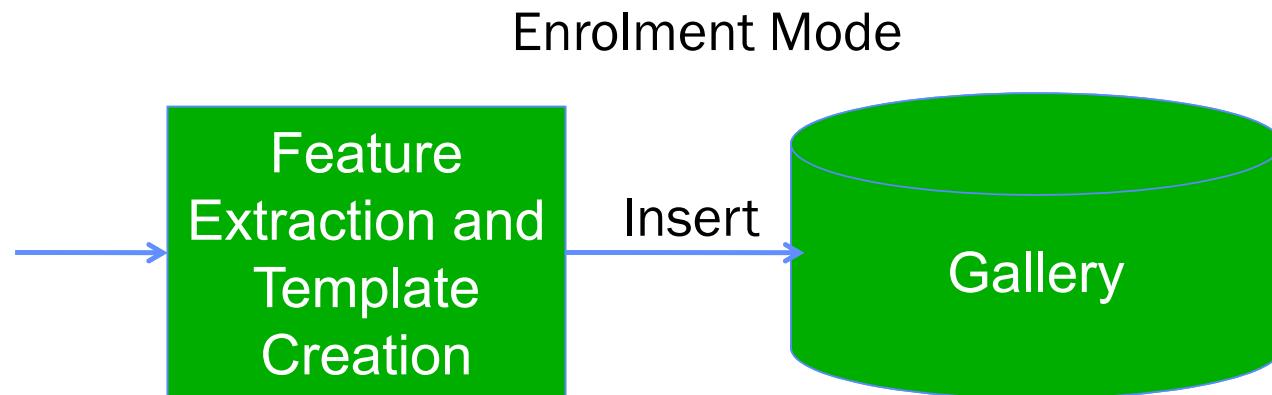
There is **always** a tradeoff between acceptance and rejection errors

The moral of the story is that one error can always be improved at the expense of the other

Typically fix FRR at 0.1% and report FAR



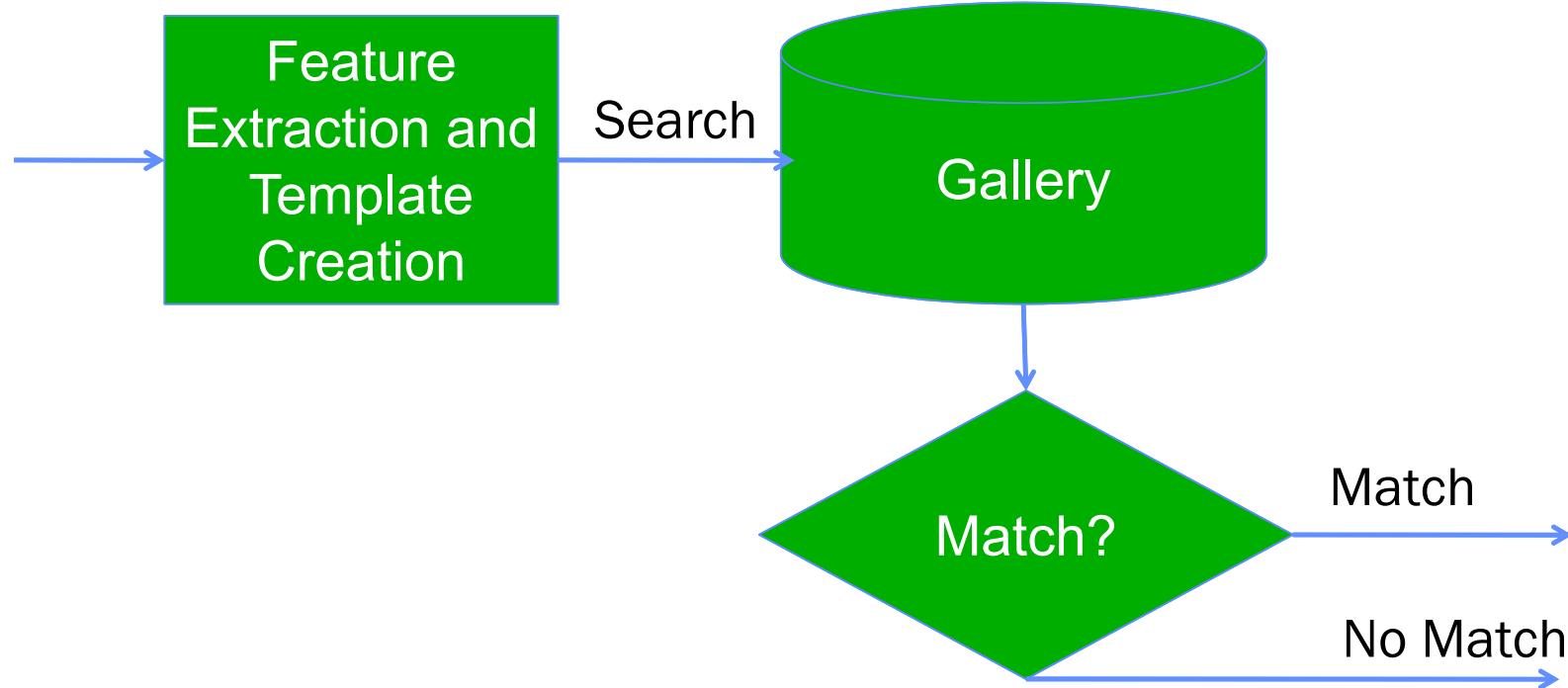
# How Biometrics Work





# How Biometrics Work

Verification/Recognition Mode





# Fingerprint Recognition

- Well-developed technology with good forensic accuracy
- Contact-based technology
- Hygiene issues for mass transport usage
- Slow enrolment but fairly fast recognition
- Unsuitable for about 3% of the population (weak fingerprint ridges)
- Can tell twins apart –fingerprint and iris are not DNA coded
- Mostly cooperative and very detectable

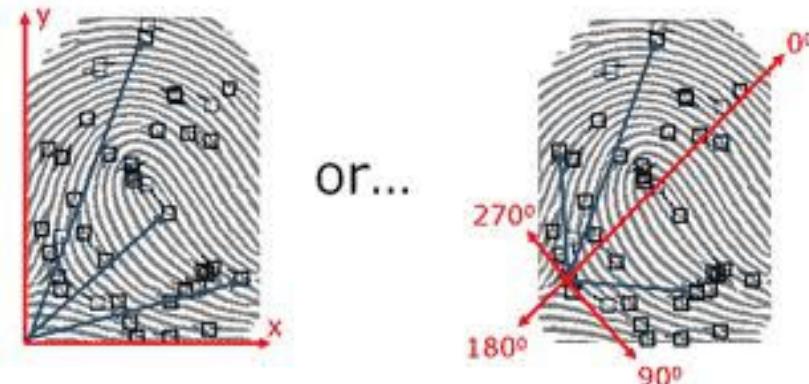
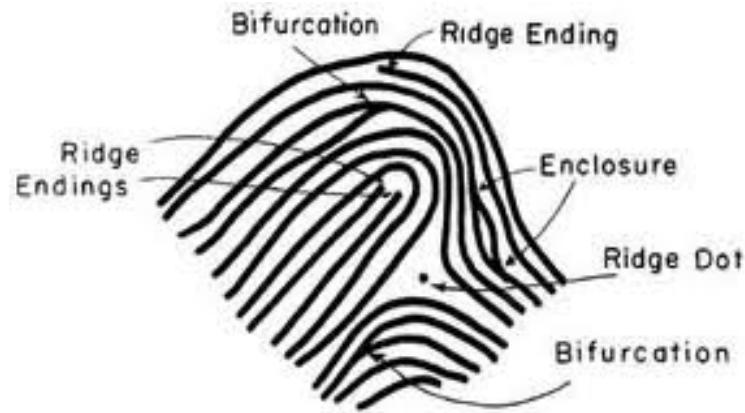


# Fingerprint Recognition





# Principle of Operation



Match 7 to 10 minutiae points by aligning prints. The more matches, the more accurate the fingerprint match.

Can use all fingers and thumbs to improve match.

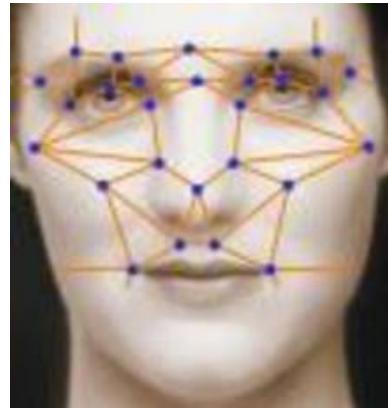


# Face Recognition

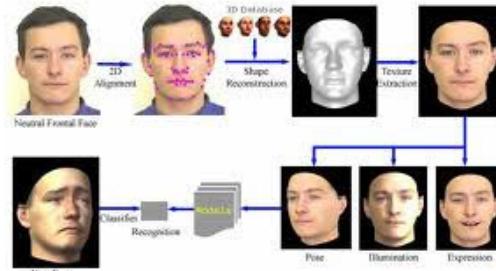
- Often needs a **face detection** stage before recognition
- Good performance generally requires controlled image capture conditions
- Can work over the internet (social networks)
- Passport quality verification works quite well (SmartGate)
- CCTV, uncooperative, and low-resolution recognition is much harder
- This is the biometric that humans use everyday, so untrained people can verify automated result. Thus there can be a low cost for false alarms, if implemented well
- Can be problems with large galleries and twins
- Easy to collect from a large distance with telephoto lenses
- Can be uncooperative and undetectable



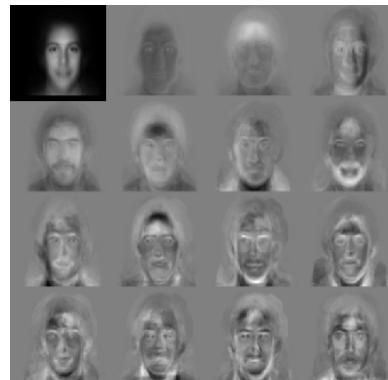
# Principles of Operation



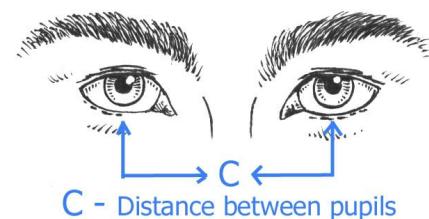
Elastic Bunch Graphs



Pose Compensation



PCA or Eigenface



Huge variety of methods

Very complicated due to pose, illumination, and expression compensation

Aim is to extract mathematical features which match identity. No physical measurements of face are generally used!

Holy grail of biometrics, but generally considered to be unreliable compared to iris and fingerprint



## Problem: Matching unfamiliar faces is not as easy as it sounds





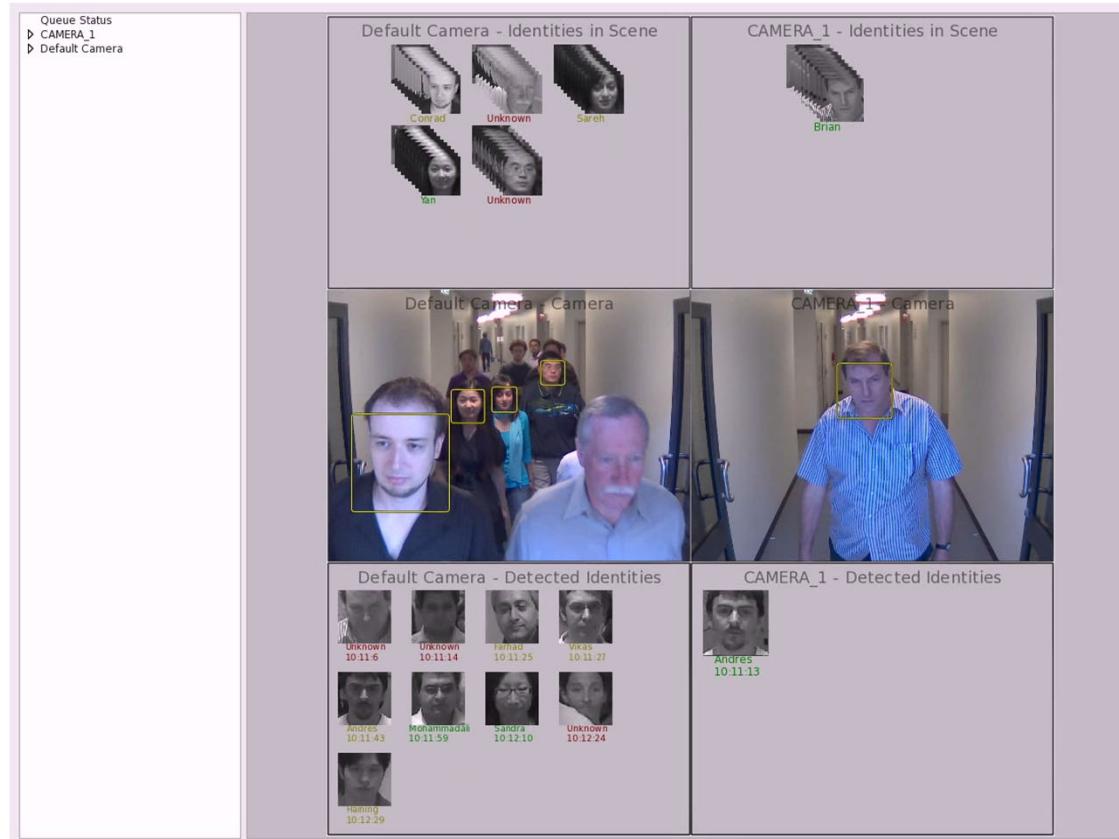
# Facial Feature Detection for Alignment



IEEE Trans. Image Processing, Nov. 2002. Moon, Chellappa and Rosenfeld



# Video: CCTV Face Search in Operation



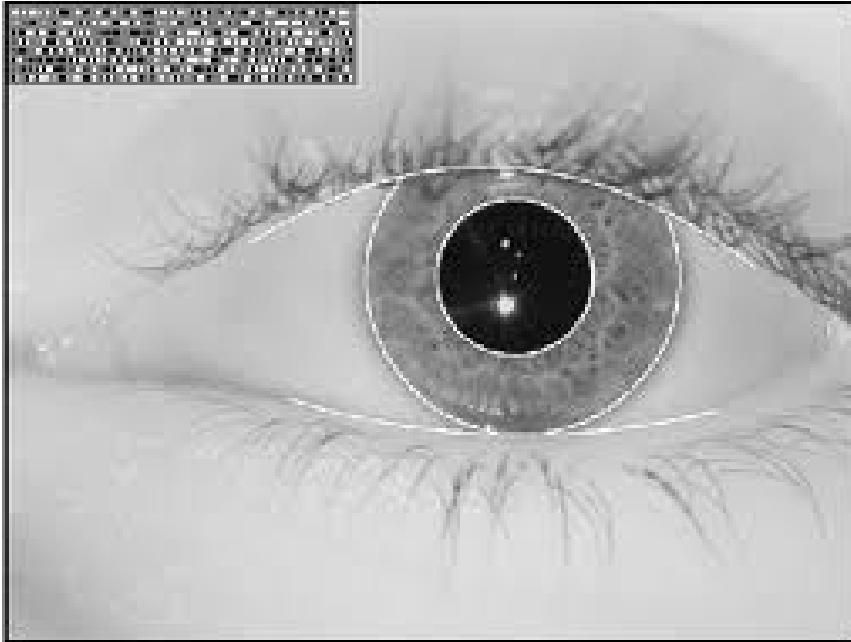


# Iris Recognition

- Often considered to be the most accurate biometric technology
- Very low false accept (alarm) rates
- New technologies such as **iris in motion** allow collection from up to 1m (40") standoff. Can process up to 50 persons per minute.
- Can tell twins apart
- Can be uncooperative though usually detectable



## Principle of Operation



Produces mathematical code from texture of the iris under near infrared illumination

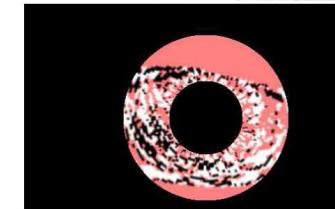
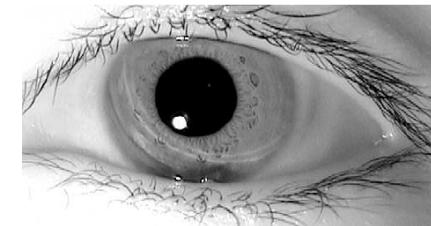
Can match both left and right eyes

Very good recognition performance



## Summary of Iris in Motion

- For
  - Very reliable and fast recognition performance
  - Touch free and continuous operation
  - No need to remove glasses
  - Works in a variety of lighting conditions
  - Enrolment on the move
- Against
  - Requires new infrastructure in addition to CCTV
  - Identity based on iris unable to be verified by human
  - Can have problems with contact lenses including possibility of identity fraud via printed lenses
  - Collection of yet another biometric database (iris)
- May be overcome by combining with cctv face technology



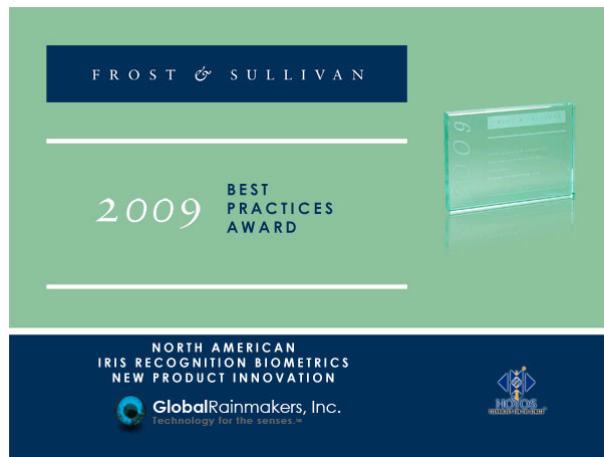


## Fusion of Face and Iris in Motion

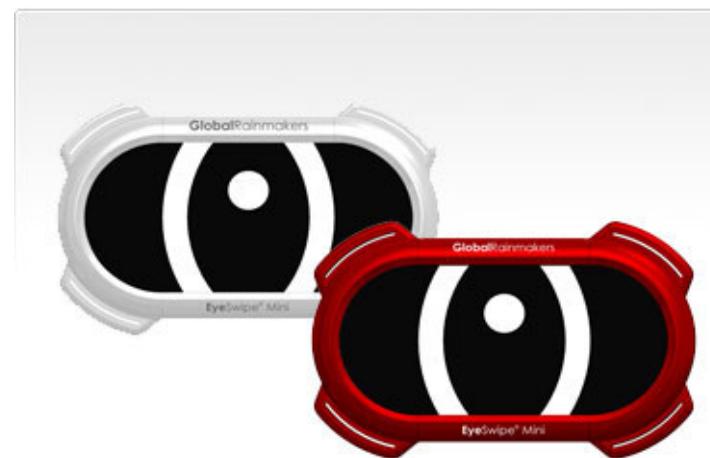
Iris in Motion technology  
as developed for the CIA



Winner of 2009 Frost &  
Sullivan Best Practice Award



Very reliable and fast (50 per minute) recognition of people.  
Fuse with uncooperative face recognition for more powerful system.





## Video: Iris in Motion in Operation



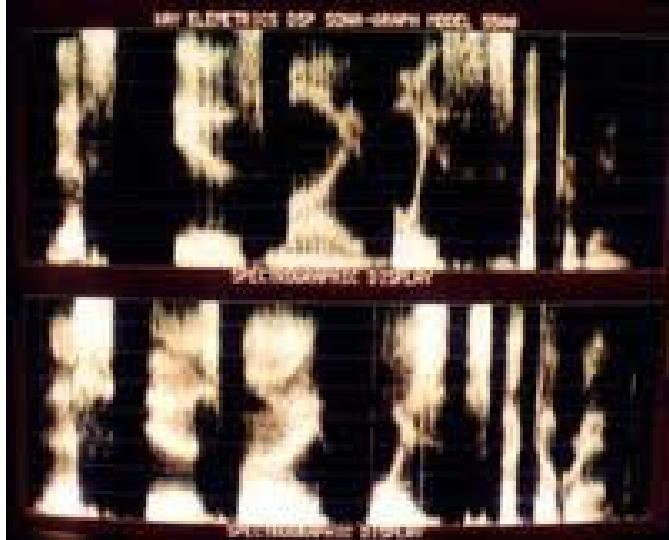


## Voice Identification

- Low accuracy
- Mostly short distance capture
- Difficult to capture in **noisy public environments**
- Affected by common colds and flu
- Probably least suitable for border control
- Better suited to investigation
- Can be uncooperative and undetectable



## Principle of Operation



Perform frequency analysis of voice to determine resonances of head cavities which indicate identity

Fairly unreliable

Mostly used for verification rather than recognition



# Typical Applications

- Face
  - border control, investigative, drivers licences, Smartgate
- Iris
  - border control, forensic (horse ID, labour control)
- Fingerprint
  - USA border control, forensic
- Voice
  - Investigative, telephone intercepts



# Implementation Issues

- Accuracy
- Recognition performance on large galleries
  - Performance always drops with gallery size
- Performance against demographics
  - (HP face recognition accused of racism)
- Control of capture environment
  - Face: Pose, illumination, expression, glasses etc
  - Iris: Pose, eye blink, gaze angle
  - Fingerprint: correct print technique
  - Voice: background noise, quality of channel
- Can result be verified by human operator?
  - Ok for Face, not for others



# Implementation Issues

- Attended vs unattended mode
  - All can operate in unattended mode to some extent
- Language and Cultural Considerations
  - Face is difficult to capture for women in Islamic Countries
  - Fingerprint is associated with criminals and has low public acceptance
  - Hygiene is an issue for contact based technologies
- Most systems are used in verification mode
  - In this case false reject rate is the major practical issue
  - Often false accept rate must be unacceptably high to keep false reject levels low defeating a prime aim of biometric technology
  - E.g. Manchester Airport where husband swapped passport with wife



## Cultural and Social Issues

- Fingerprints are associated with criminals
- Face recognition can be difficult with dark skin
- Faces are covered in Islamic societies
- Mass use of biometrics raises security concerns for mass biometric databases –need for cancellable biometrics
- Biometric systems can be fooled
  - Photos of faces, videos of faces
  - Iris contact lenses
  - Fingerprint tape
  - Voice manipulation
- To prevent biometric defeats, may need to run system in attended mode



## Private Sector Trends

- Biometrics such as fingerprint and face used as convenient alternatives to passwords
- Face and fingerprint biometric appearing on mobile devices
- Typically encourage voluntary adoption of biometrics for convenient and fast security checks
- Disneyland uses fingerprints to manage large scale ticketing systems.
- Bank of America deploying iris on the move internally
- Iris used to detect racehorse substitution (Fine Cotton Affair)



# Government Uses of Biometrics

- Border Control
  - Face, Iris, Fingerprint
- Driver's Licence
  - Face Recognition to prevent false licence issuance
  - Australian Digital Licence Project
- Enforcement of Disbarment
  - The United Arab Emirates checks irises of Third Country National labour (mostly Indian and Pakistani) to enforce disbarment at the borders



# Questions



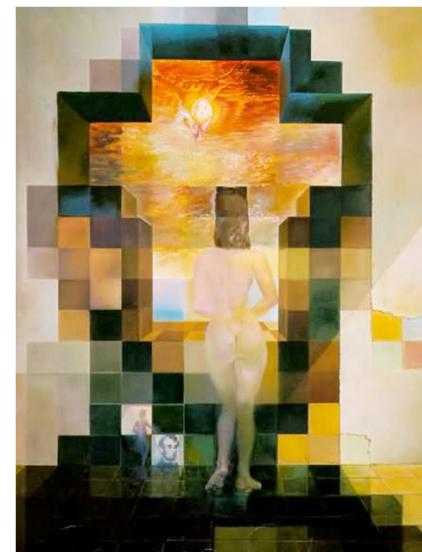


# Face detection

**Some slides courtesy Kaihang Wu and Antonio Torralba**



# Faces





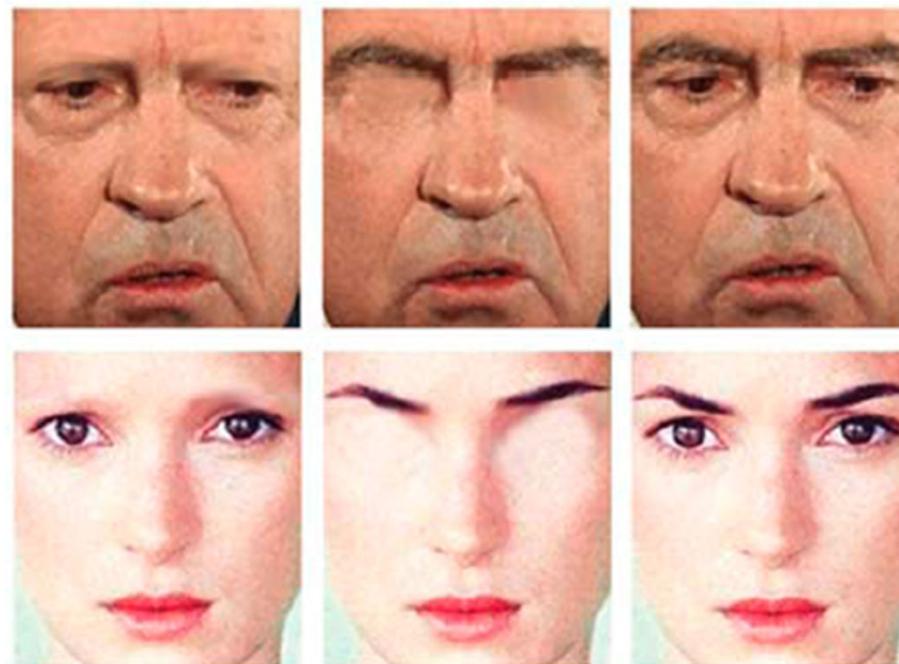
Sinha et al.: Face Recognition by Humans: Nineteen Results Researchers Should Know About



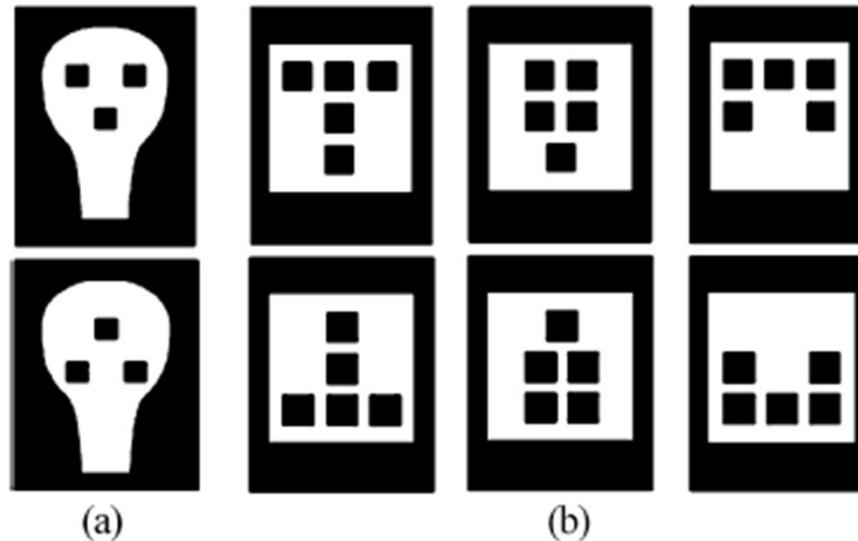
**Fig. 1.** Unlike current machine-based systems, human observers are able to handle significant degradations in face images. For instance, subjects are able to recognize more than half of all familiar faces shown to them at the resolution depicted here. Individuals shown in order are: Michael Jordan, Woody Allen, Goldie Hawn, Bill Clinton, Tom Hanks, Saddam Hussein, Elvis Presley, Jay Leno, Dustin Hoffman, Prince Charles, Cher, and Richard Nixon.



## The Importance of Eyebrows



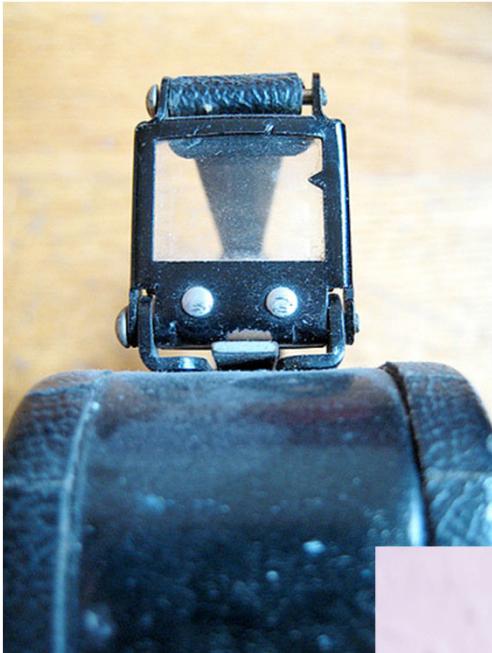
**Fig. 5.** Sample stimuli from Sadr et al.'s [70] experiment assessing the contribution of eyebrows to face recognition: original images of President Richard M. Nixon and actor Winona Ryder, along with modified versions lacking either eyebrows or eyes.

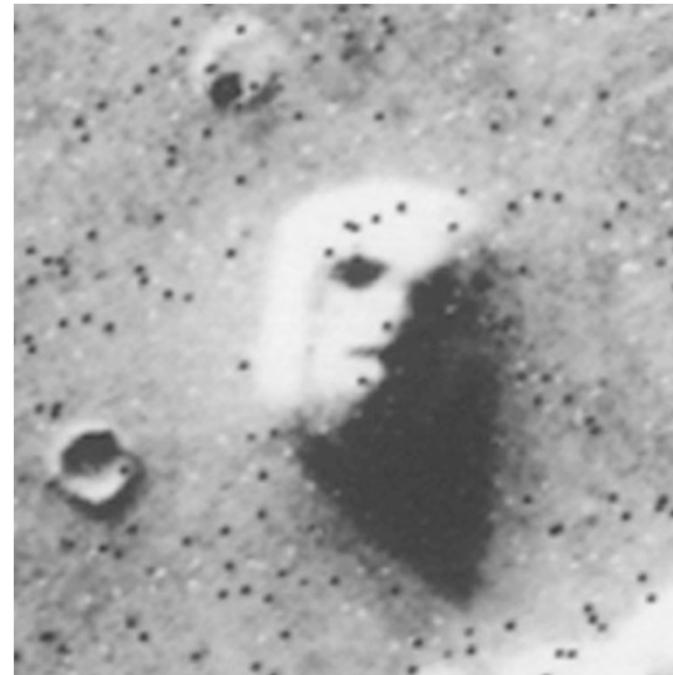


**Fig. 15.** (a) *Newborns preferentially orient their gaze to face-like pattern on top, rather than one shown on bottom, suggesting some innately specified representation for faces (from [36]).* (b) *As a counterpoint to idea of innate preferences for faces, Simion et al. [73] have shown that newborns consistently prefer top-heavy patterns (left column) over bottom-heavy ones (right column). It is unclear whether this is the same preference exhibited in earlier work, and if it is, whether it is face-specific or some other general-purpose or artifactual preference.*



## Faces everywhere

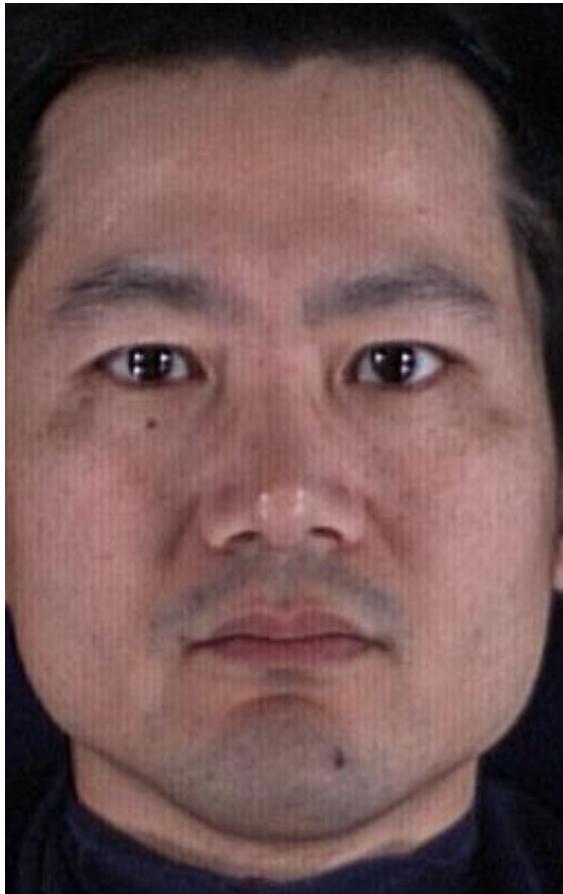




**Why is that nobody has reported finding an arm, or an ear on one of the Mars pictures?**



## Challenges: illumination





# FACE DETECTION





# Face Detection

- Goal: Identify and locate human faces in an image (usually gray scale) regardless of their position, scale, in plane rotation, orientation, pose and illumination
- The first step for any automatic face recognition system
- A very difficult problem!
- First aim to detect upright frontal faces with certain ability to detect faces with different pose, scale, and illumination
- One step towards Automatic Target Recognition or generic object recognition



**Where are the faces, if any?**



# Why Is Face Detection Difficult?

- **Pose:** Variation due to the relative camera-face pose (frontal, 45 degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded.
- **Presence or absence of structural components:** Facial features such as beards, mustaches, and glasses may or may not be present, and there is a great deal of variability amongst these components including shape, color, and size.
- **Facial expression:** The appearance of faces are directly affected by a person's facial expression.
- **Occlusion:** Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.
- **Image orientation:** Face images directly vary for different rotations about the camera's optical axis.
- **Imaging conditions:** When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face.



# Challenges

- Challenging Problem
  - Image conditions:  
size, lighting, resolution, distortion
  - Out-of-plane / In-plane rotation
  - Partial occlusions:  
sunglasses, long hair, hats, masks
- PIE - Pose, Illumination, and Expression





## ***Rapid Object Detection Using a Boosted Cascade of Simple Features***

**Paul Viola      Michael J. Jones**  
**Mitsubishi Electric Research Laboratories (MERL)**  
**Cambridge, MA**

**Most of this work was done at Compaq CRL before the authors moved to MERL**

**Manuscript available on Blackboard:**

<http://citeseer.ist.psu.edu/cache/papers/cs/23183/http:zSzzSzwww.ai.mit.eduSzpeopleSzviolaSzresearchSzpublicationsSzICCV01-Viola-Jones.pdf/viola01robust.pdf>



## What is novel about this approach?

- Feature set (... is huge about 16,000,000 features)
- Efficient feature selection using AdaBoost
- New image representation: Integral Image
- Cascaded Classifier for rapid detection
  - Hierarchy of Attentional Filters

**The combination of these ideas yields the fastest known face detector for gray scale images.**

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



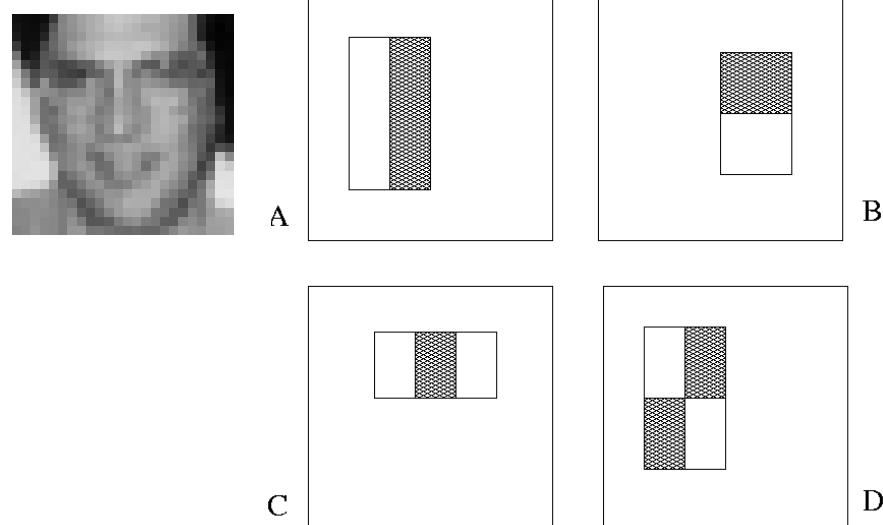
# Image Features

**“Rectangle filters”**

**Similar to Haar  
wavelets**

**Differences between  
sums of pixels in  
adjacent rectangles**

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$



$$160,000 \times 100 = 16,000,000$$

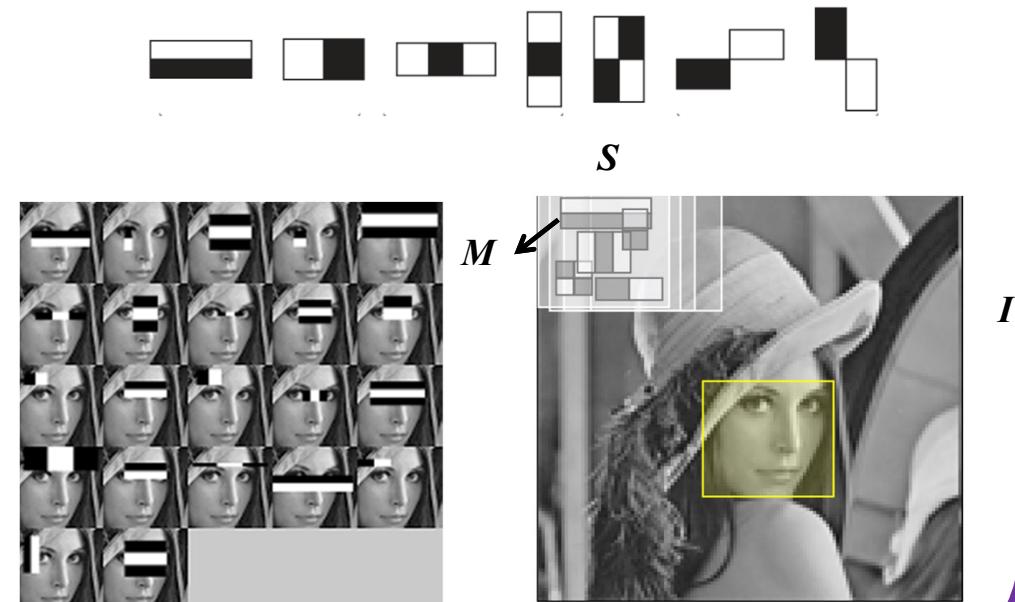
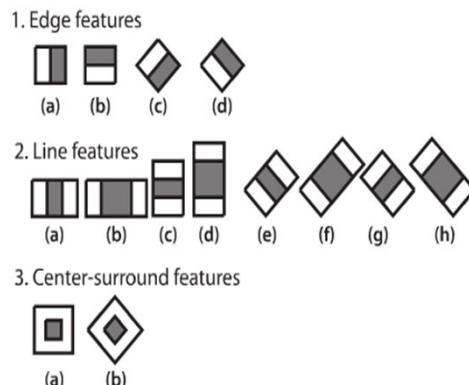
**Unique Features**

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



# Haar Features

- Rectangle Features
  - Subtract sum of pixels in white area from the sum of pixels in black area
- Three original types of features and several new features are generally used
- Extended feature set





# Integral Image

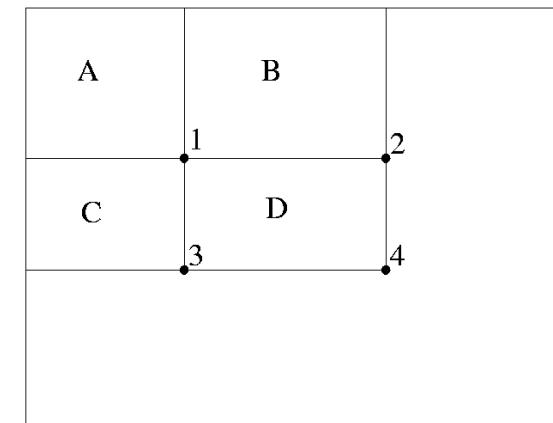
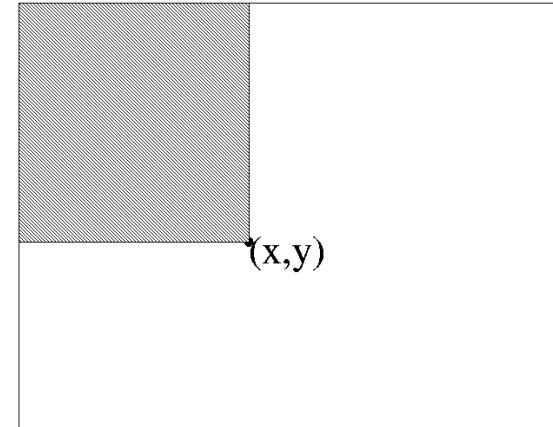
- Define the Integral Image

$$I'(x, y) = \sum_{\substack{x' \leq x \\ y' \leq y}} I(x', y')$$

- Any rectangular sum can be computed in constant time:

$$\begin{aligned} D &= 1 + 4 - (2 + 3) \\ &= A + (A + B + C + D) - (A + C + A + B) \\ &= D \end{aligned}$$

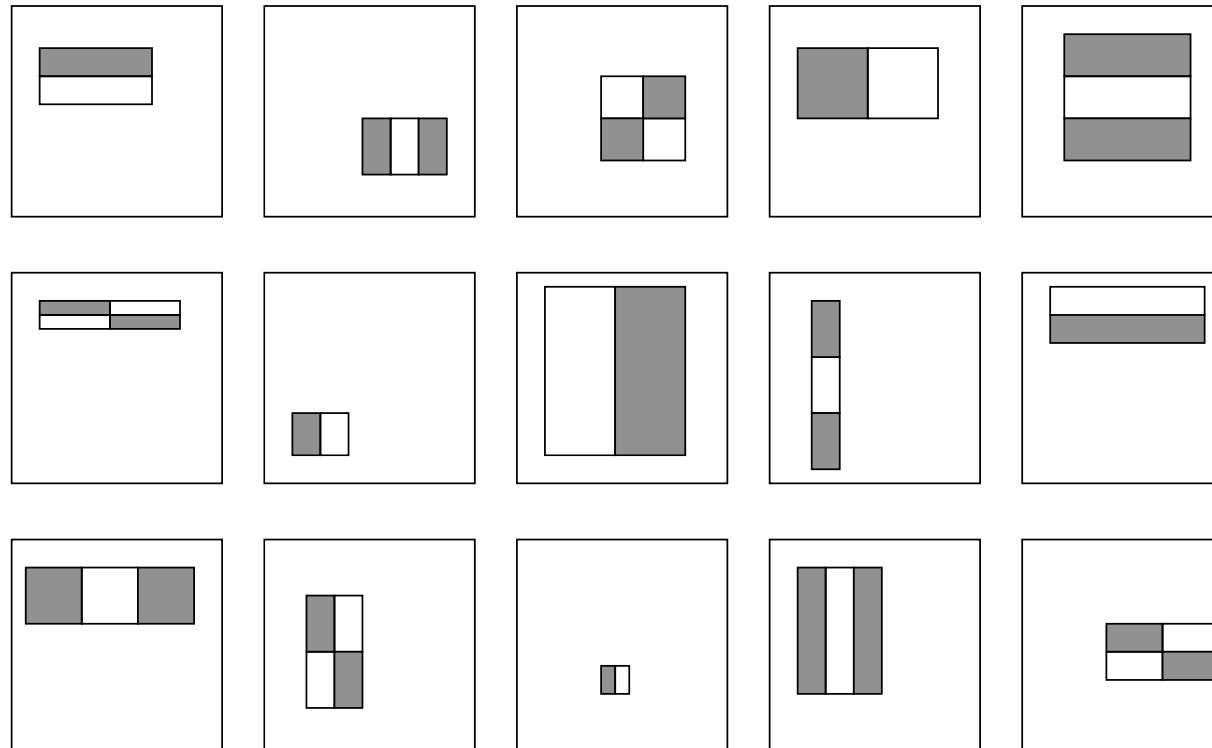
- Rectangle features can be computed as differences between rectangles



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



## Huge “Library” of Filters



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



# AdaBoost for Efficient Feature Selection

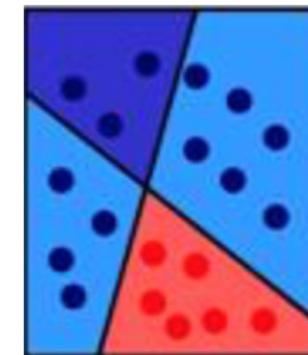
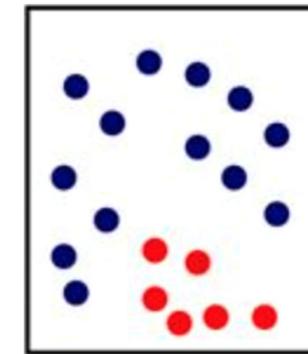
- Our Features = Weak Classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples
  - (There are many tricks to make this more efficient.)

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



## Adaboost – Machine Learning classifier

- Given: example N images labeled face/non-face
  - Initially, all weights set equally  $w = 1/N$ .
- Repeat N times
  - Step 1: choose the most efficient weak classifier with the lowest detection error
  - Step 2: Update the weights to emphasize the examples which were incorrectly classified by the first weak classifier and consider the next best classifier. This makes the next weak classifier focus on “harder” examples
- Final (strong) classifier is a weighted combination of the N “weak” classifiers
  - Weighted according to their importance





## The final strong classifier

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Strong classifier    Image    Weight    Weak classifier

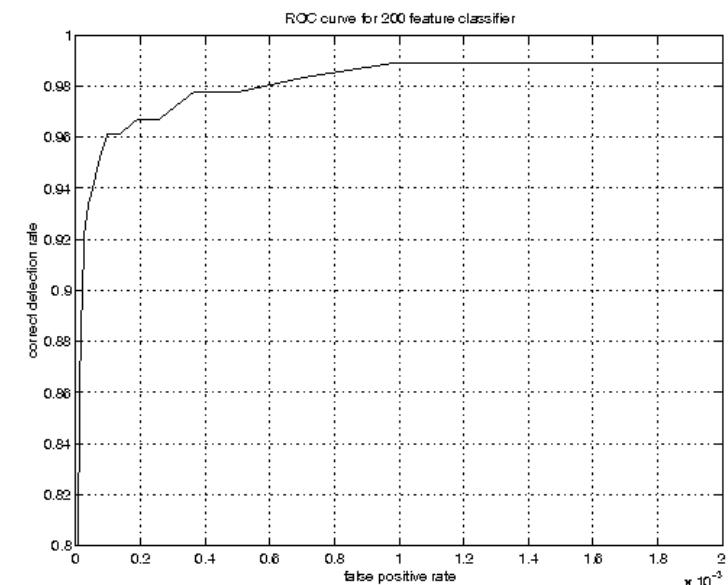
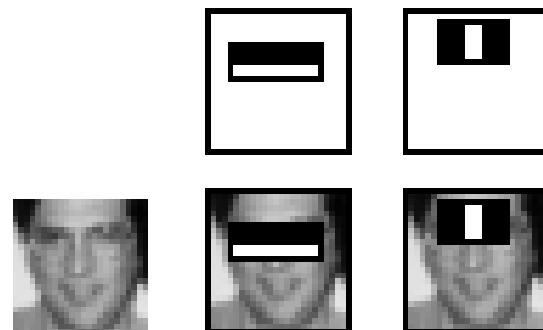


# Example Classifier for Face Detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084 false positives.

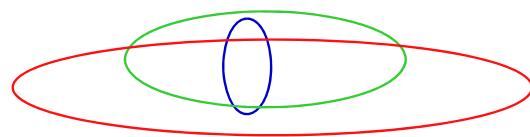
Not quite competitive.  
Need to add more features,  
but then that slows it down.



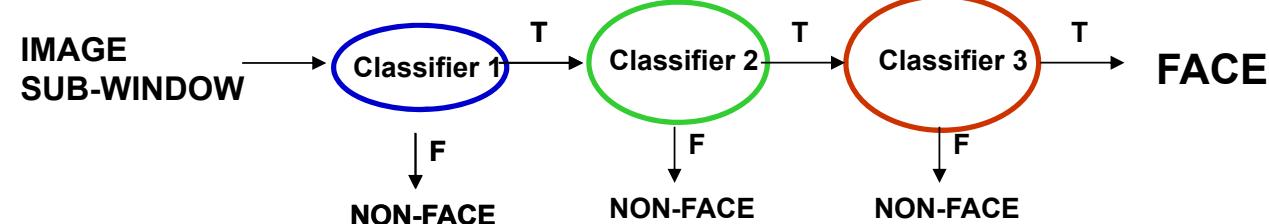
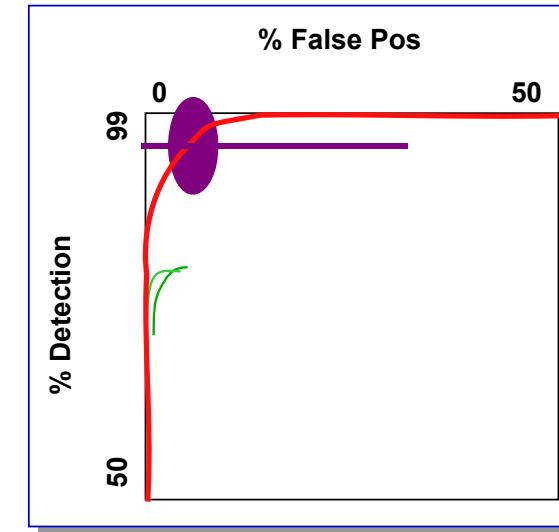
Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

## Develop fast, accurate classifier using a cascade

- Given a nested set of classifier hypothesis classes



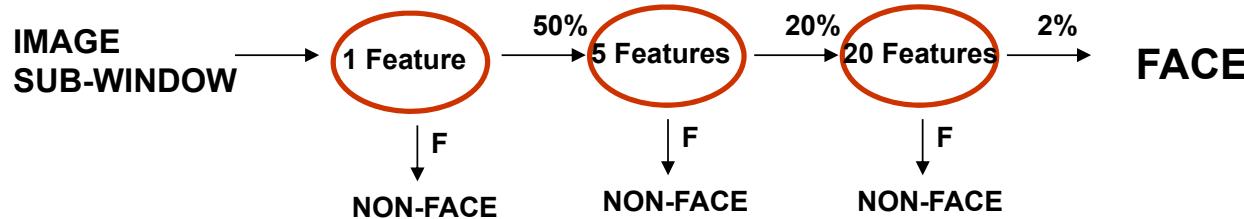
- Computational Risk Minimization



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



# Cascaded Classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



## A Real-time Face Detection System

**Training faces:** 4916 face images (24 x 24 pixels) plus vertical flips for a total of 9832 faces

**Training non-faces:** 350 million sub-windows from 9500 non-face images

**Final detector:** 38 layer cascaded classifier  
The number of features per layer was 1, 10, 25, 25, 50, 50, 50, 75, 100, ..., 200, ...

**Final classifier contains 6061 features.**

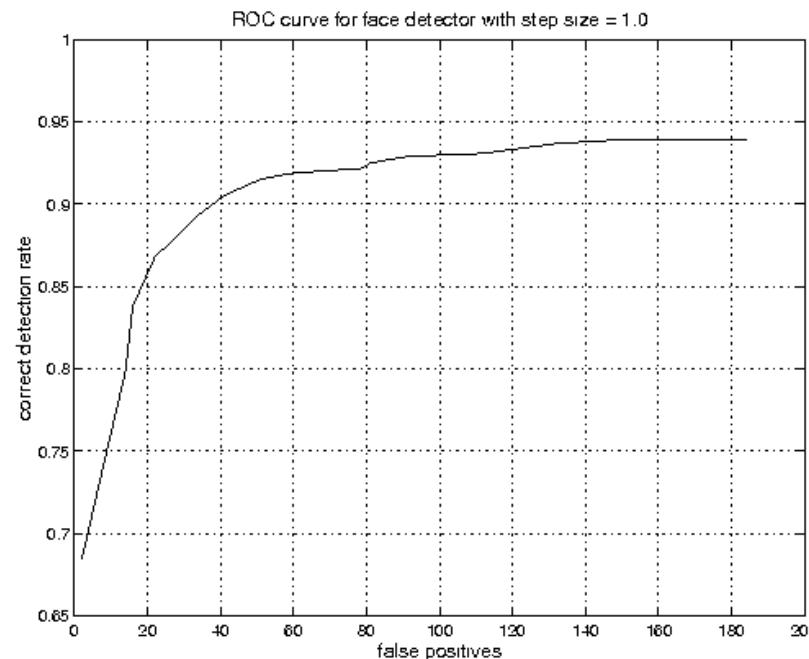


Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



# Accuracy of Face Detector

**Performance on MIT+CMU test set containing 130 images with 507 faces and about 75 million sub-windows.**



**Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001**



# Comparison to Other Systems

Detector \ False Detections	10	31	50	65	78	95	110	167
Detector								
Viola-Jones	76.1	88.4	91.4	92.0	92.1	92.9	93.1	93.9
Viola-Jones (voting)	81.1	89.7	92.1	93.1	93.1	93.2	93.7	93.7
Rowley-Baluja-Kanade	83.2	86.0				89.2		90.1
Schneiderman-Kanade				94.4				

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



## Speed of Face Detector

**Speed is proportional to the average number of features computed per sub-window.**

**On the MIT+CMU test set, an average of 9 features out of a total of 6061 are computed per sub-window.**

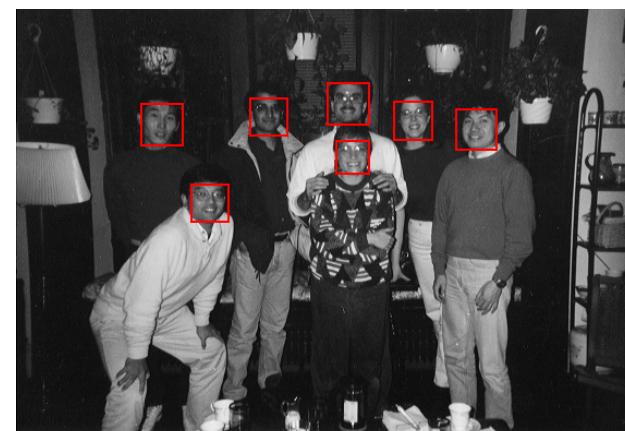
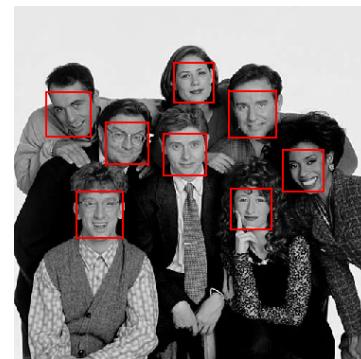
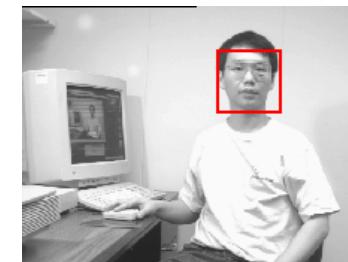
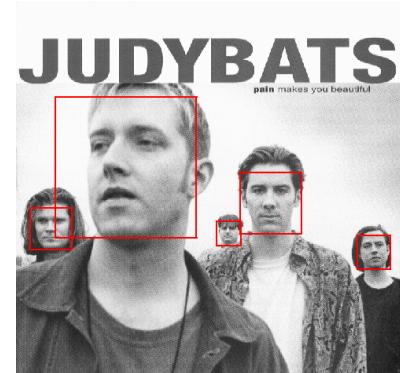
**On a 700 Mhz Pentium III, a 384x288 pixel image takes about 0.067 seconds to process (15 fps).**

**Roughly 15 times faster than Rowley-Baluja-Kanade and 600 times faster than Schneiderman-Kanade.**

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



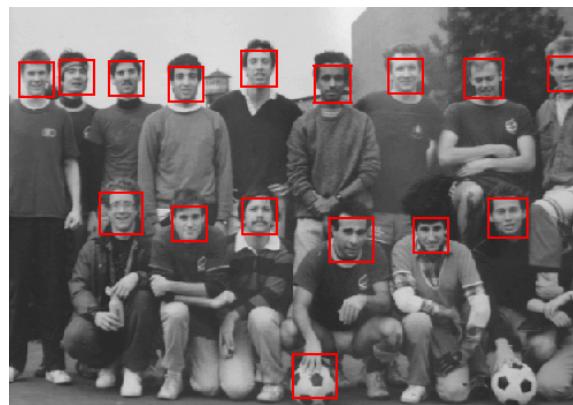
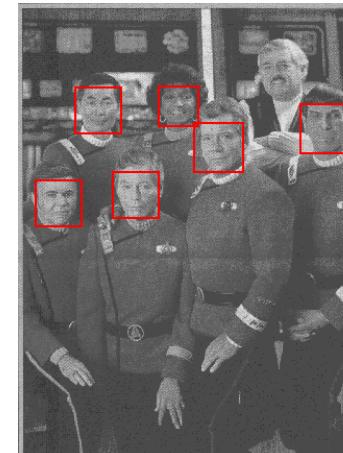
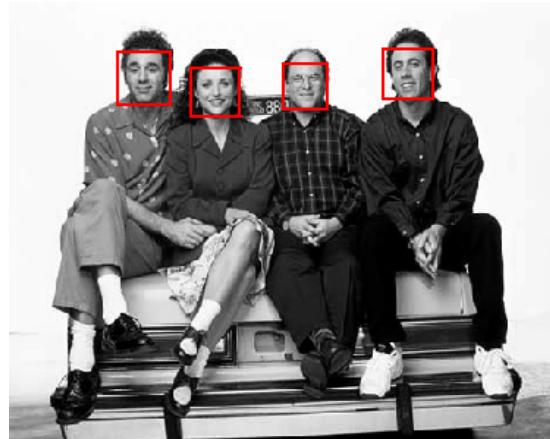
# Output of Face Detector on Test Images



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



## More Examples



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001



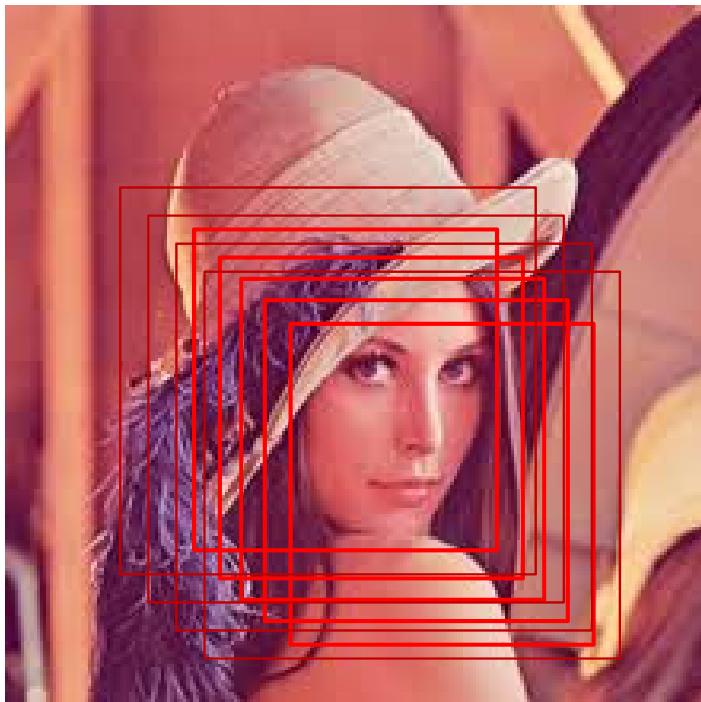
## Slide and rescale window

- Slide smallest window across image with stepsize (stride) of several pixels for speed
- Increase window size by, say, 20% and repeat
- Continue until you reach largest window size





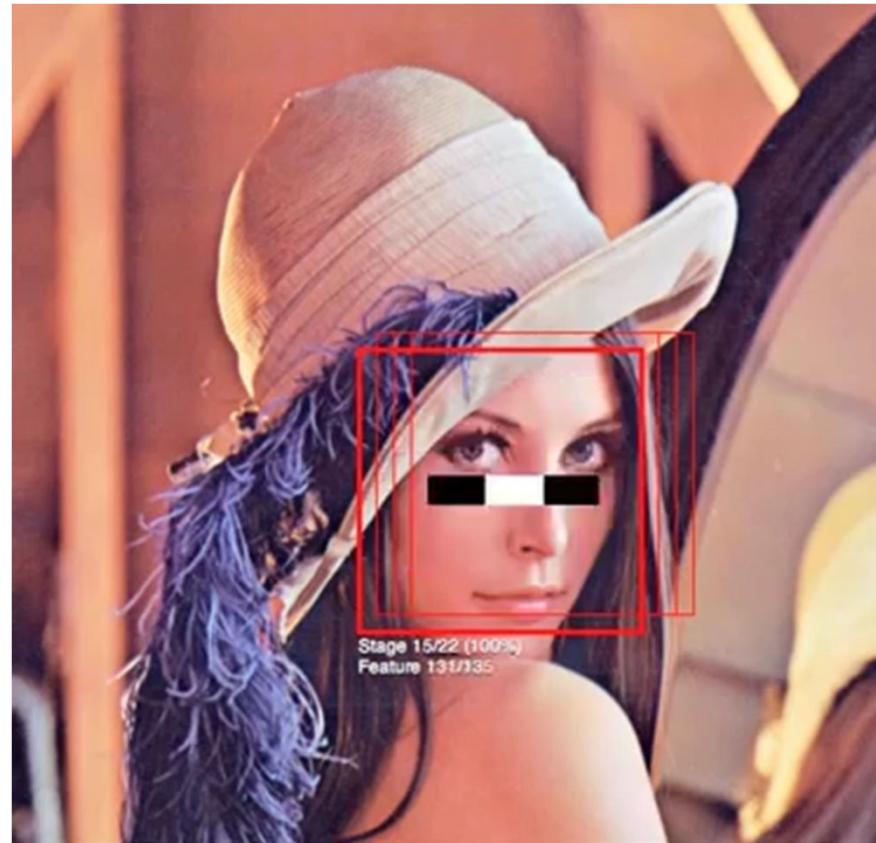
## Non-Maximal Suppression



**If you set a threshold for ‘faceness’  
then you will get many detections  
over your threshold as you pass over a  
face.  
You need to suppress the weaker  
detections and keep the strongest  
response.**



## Viola-Jones Cascade Demo





## Comments

- Good for near frontal faces with up to 15 degrees pose angle
- Faces are often not well-centred due to stride of cascade algorithm
- Often need feature based aligner for face recognition
- For frontal faces, we can align the eyes
- The eyes can also be used to correct for in-plane rotation (tilt)
- For non-frontal faces, what do we align?
- Difficult to adapt to high pose angle detection because that would require more a more complex first stage which slows the algorithm and increases false acceptance

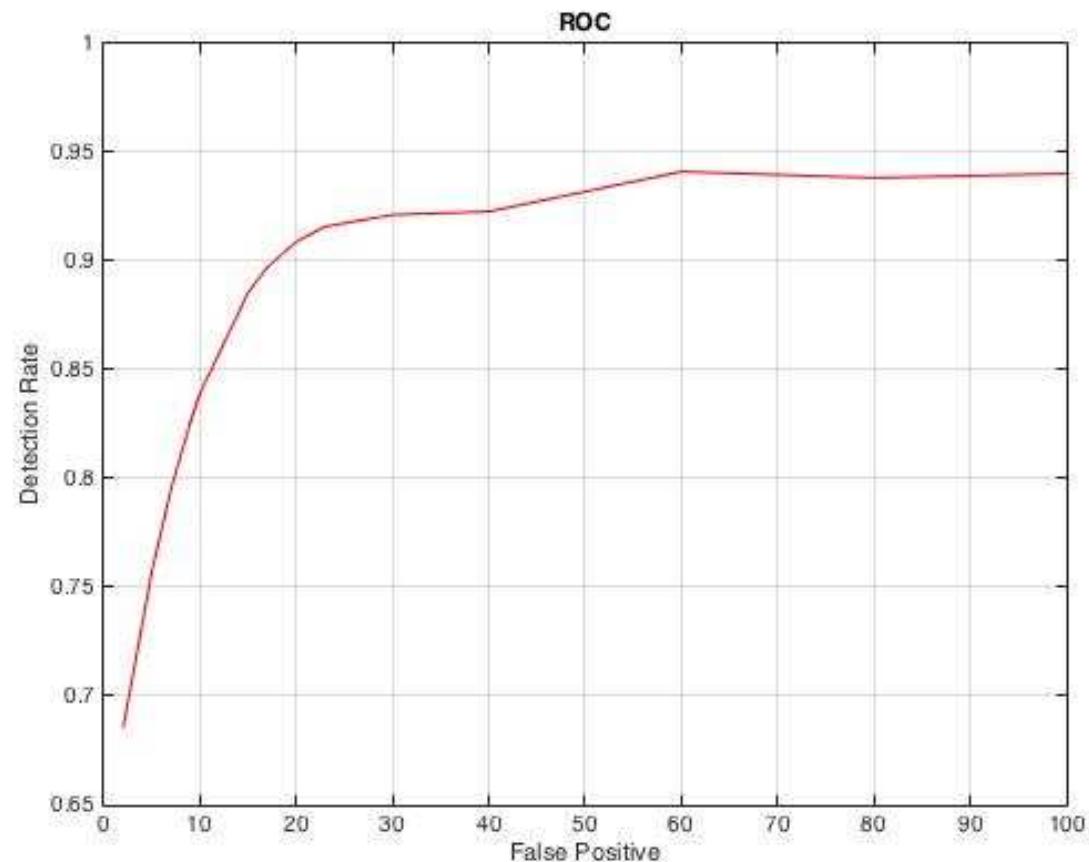


# Our Performance of a single-node classifier

Examples for face samples:



Examples for non-face samples:





## Our Test Performance on 4 datasets

*Table 1. Datasets*

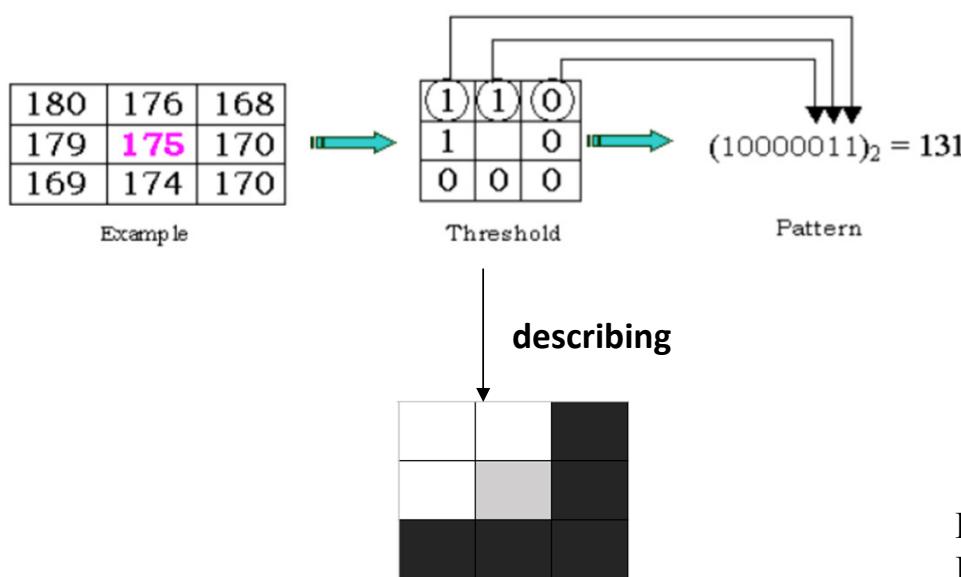
Dataset	Number of images	Average image dimension	Number of annotated faces
Fddb	2845	377 x 399	5171
CMU	180	421 x 422	721
Yale	165	320 x 243	165
ORL	400	112 x 92	400

*Table 2. The comparison of performance*

Test set performance									
Traing set	Fddb		CMU		Yale		ORL		
stage	Time	Precision	Detection Rate	Time	Precision	Detection Rate	Time	Precision	Detection Rate
1	34998s	0.512	0.512	13344s	0.751	0.751	10464s	0.584	0.584
3	56032s	0.527	0.527	28976s	0.774	0.774	25121s	0.561	0.561
10	~23h	0.654	0.654	~54000s	0.876	0.876	~53560s	0.640	0.640
20	~50h	0.740	0.740	~27h	0.958	0.958	~27h	0.769	0.769

## Let's try LBP features

- LBP features:
- Comparative result (Time)



Feature	V.J. algorithm	
	Time(s)	Detection Rate
Haar	2204.4	0.937
LBP	1113.33	0.870

**LBP is Local Binary Patterns**  
**Faster than Haar but slightly less accurate**

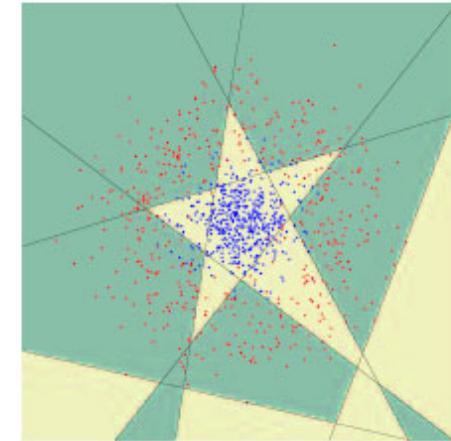


## References

1. Min R, Hadid A, Dugelay J-L. Efficient Detection of Occlusion prior to Robust Face Recognition. *The Scientific World Journal*. 2014;2014:519158. doi:10.1155/2014/519158.
2. Zhang, Zhanpeng, et al. "Facial landmark detection by deep multi-task learning." *Computer Vision–ECCV 2014*. Springer International Publishing, 2014. 94-108.
3. J. Chang-yeon, "Face Detection using LBP features," *Final Project Report*, vol. 77, 2008.
4. J. Šochman and J. Matas, "AdaBoost and face detection," *CZECH TECHNICAL UNIVERSITY. Repùblica Checa*, 2003.
5. P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, pp. 137-154, 2004.



# Adaboost



Strong Classifiers formed from Weak Classifiers  
Can a crowd be smarter than the participants in the crowd?

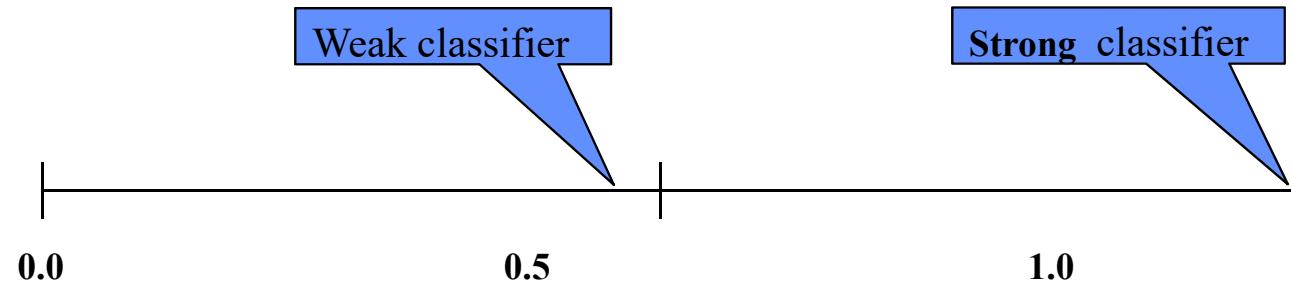


## Pedigree of Adaboost

- **AdaBoost**, short for "Adaptive Boosting", is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire who won the Gödel Prize in 2003 for their work. It can be used in conjunction with many other types of learning algorithms to improve their performance.
- Integral element of Viola-Jones Face Detection which won most significant paper over 10 years at CVPR in 2011
- It is easy and it actually works!
- See Patrick Winston talk at MIT  
<https://www.youtube.com/watch?v=UHBmv7qCey4>

# Binary Classifier

- Answers question such as “Is this a face or non-face?”
- Assume we have a set of classifiers of the form:
  - $h [-1,1]$ 
    - i.e. the classifier  $h$  looks at input and gives -1 or 1 as output
- Lets look at the error rates





# Voting Classifier

- Let's try to improve performance by creating a few classifiers and taking a vote

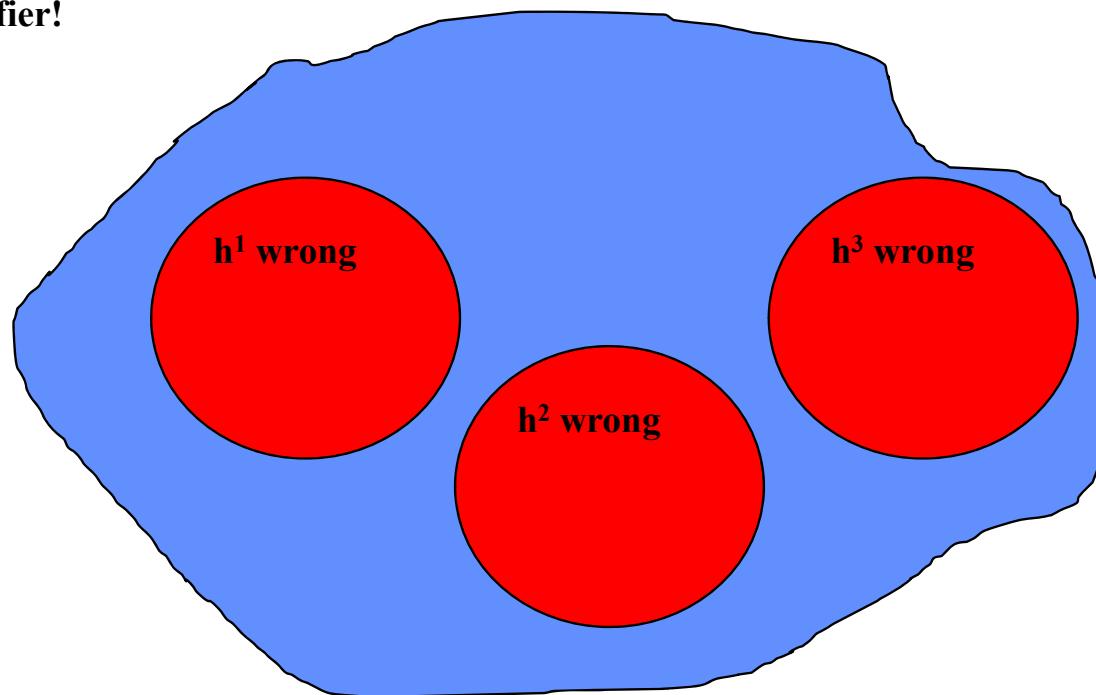
$$H(x) = \text{sgn}(h^1(x) + h^2(x) + h^3(x))$$

**Correct if 2 out of 3 are correct**



# Sample Set

**Perfect Classifier!**

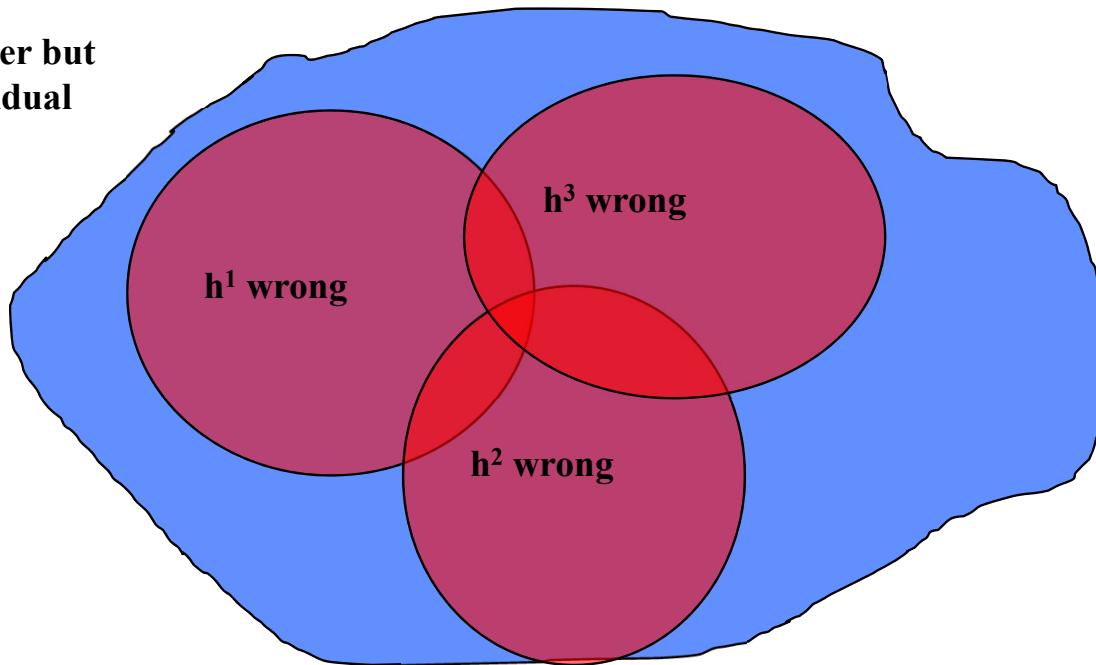




# Sample Set

**Imperfect classifier but  
better than individual  
Weak classifiers**

**Is this always  
True?**



**Always better or equal to the single  
worst classifier because intersection is a subset of the set**



## Choose Best Classifiers

- Now how do we choose the best classifiers for the vote?
- We test classifiers on the data to see which performs best
- This is fine for the first classifier, but we need the second classifier to fix the errors in the first etc.

IDEA 1

**Data  $\rightarrow h^1$**

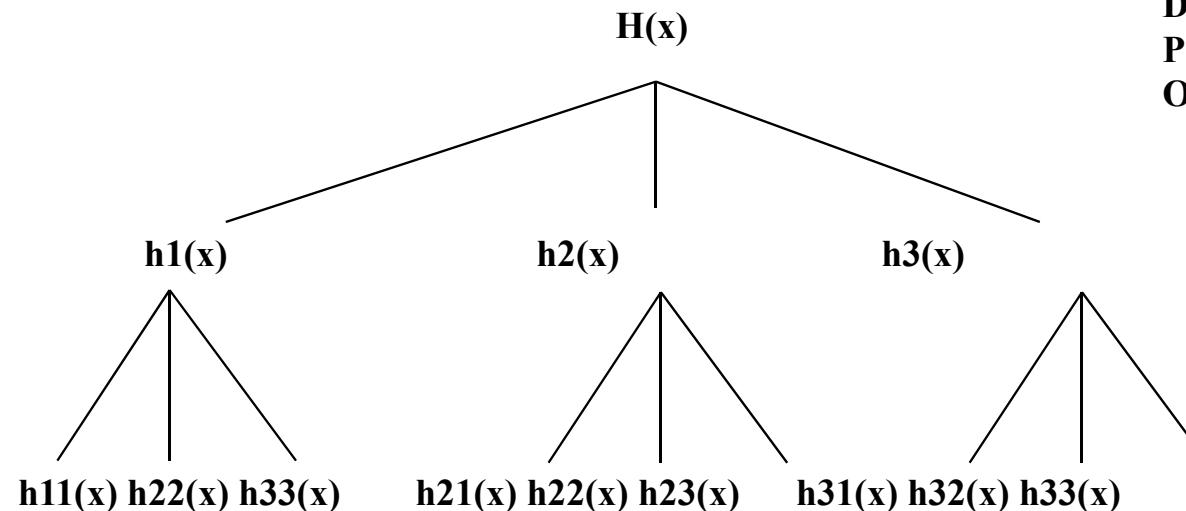
**Data with exaggerated  $h^1$  errors  $\rightarrow h^2$**

**Data with exaggerated  $h^1 \neq h^2 \rightarrow h^3$**



# Extend the Vote

**Wisdom of a crowd**

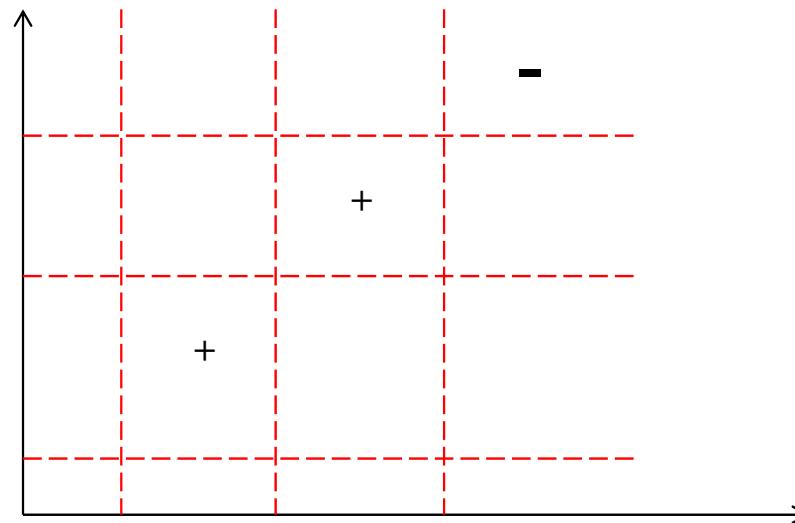


**Representative  
Democracy or  
Political Convention  
One vote one value**

**IDEA 2**



# Decision Tree Stumps



How many tests?

12

Tests = Number of lines x 2 x number of dimensions



# Why use Decision Tree Stumps?

- Could use any class of classifiers
  - Neural nets
  - Hyperplanes (linear functions)
- Decision trees are easy to explain and enumerate
- Boosting works on all classifiers



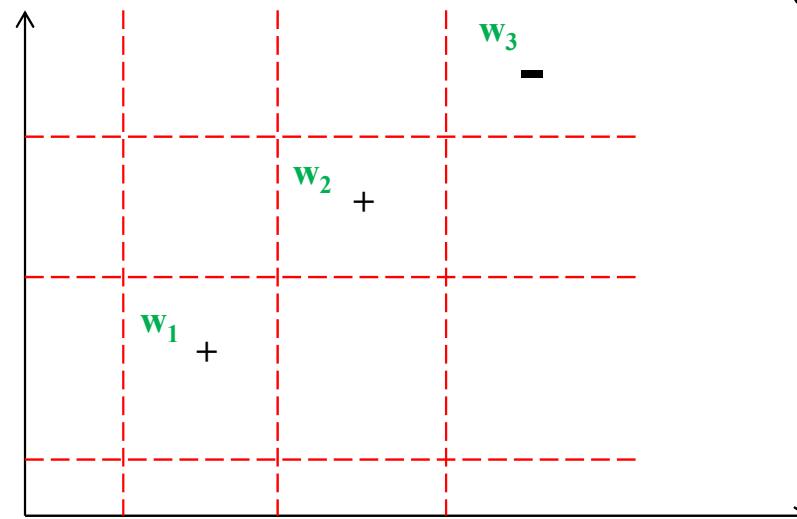
## Error Rate

$$\text{Error Rate, } \epsilon = \sum_{\text{wrong}} \frac{1}{N}$$

**So for N=50 samples, if we get 2 examples wrong,  $\epsilon$  is  $2/50 = 4\%$**



# How do we Exaggerate Errors?



**Use error weights**

$$w_i^1 = \frac{1}{N}$$

Enforces a distribution

$$\epsilon = \sum_{\text{wrong}} w_i \text{ and } \sum w_i = 1$$

IDEA 3



## Another Way to Extend the Vote

$$H(x) = \text{sgn}(h^1(x) + h^2(x) + h^3(x) + \dots)$$

But some voters are better than others!

IDEA 4

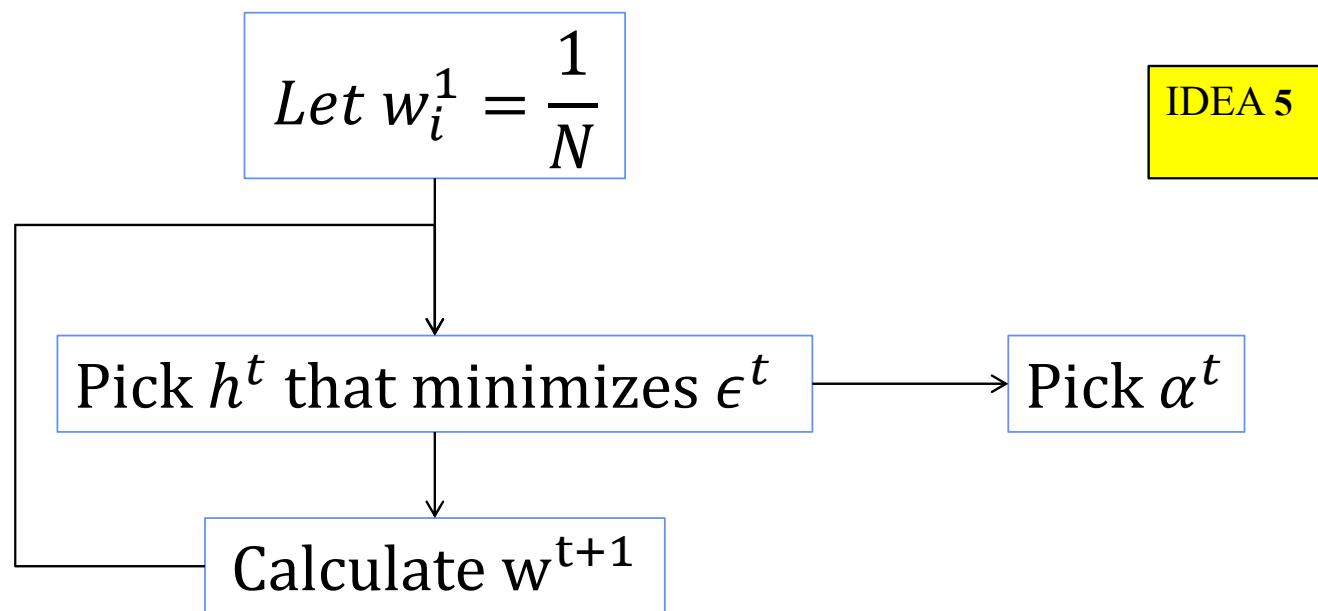
$$H(x) = \text{sgn}(\alpha^1 h^1(x) + \alpha^2 h^2(x) + \alpha^3 h^3(x) + \dots)$$

**Wisdom of a weighted crowd of experts**



# Method Outline

1. Pick the best classifier from the set
2. Reweight errors and repeat





# Reweighting

$$w_i^{t+1} = \frac{w_i^t}{Z} e^{\alpha^t h^t(x) y(x)}$$

Z is normalisation factor

IDEA 6

y(x) is function that is [-1,+1]  
depending if  $h^t$  answer is wrong or right

How was this arrived at?

It took a top mathematician over a year of contemplation  
Almost led to divorce!



## Error Bound

**Minimum error bound for whole thing if we choose**

$$\alpha^t = \frac{1}{2} \frac{\ln(1 - \epsilon^t)}{\epsilon^t}$$

**Substitute into previous equation and the new weights are**

$$w_i^{t+1} = \frac{w_i^t}{Z} * \begin{cases} \sqrt{\frac{\epsilon^t}{1 - \epsilon^t}} & \text{correct} \\ \sqrt{\frac{1 - \epsilon^t}{\epsilon^t}} & \text{wrong} \end{cases}$$



## Simplification

$$\sqrt{\frac{\epsilon^t}{1-\epsilon^t}} \sum_{\text{correct}} \omega_i^t + \sqrt{\frac{1-\epsilon^t}{\epsilon^t}} \sum_{\text{wrong}} \omega_i^t = Z = 2\sqrt{\epsilon^t(1-\epsilon^t)}$$

$$\begin{aligned}\omega_i^{t+1} &= \frac{\omega_i^t}{2} \frac{1}{1-\epsilon} \quad \text{correct} = \frac{1}{2} \frac{1}{1-\epsilon} \sum_{\text{correct}} \omega_i^t = \frac{1}{2} \\ &= \frac{\omega_i^t}{2} \frac{1}{\epsilon} \quad \text{wrong} = \frac{1}{2}\end{aligned}$$

**Easy: Sum of all weights for correct/wrong is always  $\frac{1}{2}$ !!  
No need for log or exp, simply rescale the weights**



## Example

