

Computer Vision

David Vernon

Course Text:
Machine Vision
Automated Visual Inspection and Robot Vision

David Vernon
Prentice-Hall International
1991

- 1.1 Computer Vision: Image Processing or Artificial Intelligence ?
- 1.2 Industrial Vision Vs. Image Understanding.
- 1.3 Sensory Feedback for manufacturing systems:
Why Vision ?

1.4 Examples of Industrial Machine Vision problems and solutions.

1.4.1 Measurement of Steel Bars

1.4.2 Inspection of Computer Screens

1.5 Typical System Architecture

2. Illumination and Sensors

2.1 Illumination

2.2 Sensors

2.2.1 Image Formation : Elementary Optics

2.2.2 Camera Sensors

2.2.3 Camera Interfaces and Video Standards

2.2.4 Characteristics of Camera Sensors

2.2.5 Commercially-available Cameras

3. Image Acquisition and Representation

- 3.1 Sampling and Quantisation
- 3.2 Interpixel Distances
- 3.3 Adjacency Conventions
- 3.4 Image Acquisition Hardware
- 3.5 Speed Considerations

4. Fundamentals of Digital Image Processing

4.1 Local Pixel Operations

- 4.1.1 Contrast Stretching
- 4.1.2 Thresholding
- 4.1.3 Noise Suppression by Image Addition
- 4.1.4 Background Subtraction

4.2 Neighbourhood Operations

- 4.2.1 Convolution
- 4.2.2 Noise Suppression
- 4.2.3 Thinning, Erosion and Dilation

4.3 Geometric Operations

4.3.1 Spatial Warping

4.3.1.1 The Spatial Transformation

4.3.1.2 Grey-level Interpolation

4.3.2 Registration and Geometric Decalibration

4.4 Mathematical Morphology

4.4.1 Structuring Elements

4.4.2 Hit or Miss Transformations

4.4.3 Erosion and Dilation

4.4.4 Opening and Closing

4.4.5 Thinning and Extraction of End Points

4.4.6 *Application* : Identification of End-Points of
Electrical Wires

4.4.7 A brief introduction to Grey-Scale
Mathematical Morphology

5. The Segmentation Problem

5.1 Introduction: Region and Boundary-based approaches.

5.2 Thresholding

5.2.1 Global, Local and Dynamic Approaches

5.2.2 Threshold Selection

5.3 Edge Detection.

5.3.1 Gradient and Difference Operators

5.3.2 Template Matching

5.3.1 Edge Fitting

5.3.2 Statistical Techniques

5.4 Region Growing.

5.4.1 The Split and Merge Procedure using
Quad-Trees

5.4 Boundary Detection

5.4.1 Contour Following

6. Image Analysis

6.1 Introduction : Inspection, Location and Identification

6.2 Template Matching

6.2.1 Measures of Similarity

6.2.2 Local Template Matching

6.3 Decision-Theoretic Approaches

6.3.1 Components of Statistical Pattern
Recognition Process

6.3.2 Simple Feature Extraction

6.3.3 Classification

6.3.3.1 A Synopsis of Classification using Baye's
Rule

6.4 The Hough Transform

- 6.4.1 Hough Transform for Line and Circle Detection
- 6.4.2 The Generalised Hough Transform

7. An Overview of Techniques for Shape Description

7.1 A Taxonomy of Types of Shape Descriptors

7.2 External Scalar Transform
- Features of the Boundary

7.3 Internal Scalar Transform
- Features of the Region

7.4 External Space Domain

- Spatial Organisation of the Boundary

7.4.1 An Algorithm for Re-sampling the
Boundary Chain Code

7.5 Internal Space Domain

- Spatial Organisation of the Region

8. Robot Programming & Robot Vision

8.1 A brief review of Robot Programming

Methodologies

8.2 Description of Object Pose with

Homogenous Transformations

8.3 Robot Programming :

A Wire Crimping Task Specification

8.4 A simple Robot Programming Language

8.5 Two Vision Algorithms for Identifying Ends of Wires

- 8.5.1 A Binary Vision Algorithm
- 8.5.2 A Grey-Scale Vision Algorithm
- 8.5.3 The Vision/Manipulator Interface

8.6 The Camera Model and the Inverse Perspective Transformation.

8.6.1 The Camera Model

8.6.2 The Inverse Perspective Transformation

8.6.3 Recovery of the Third Dimension

8.7 3D Vision using Structured Light

9. Image Understanding

- 9.1 Representations and Information Processing : from Image Objects Models
- 9.2 Organisation of Visual Processes
- 9.3 Visual Representations
 - 9.3.1 The Raw Primal Sketch
 - 9.3.2 The Full Primal Sketch
 - 9.3.3 The 2½D Sketch

9.3.4 3D Models

9.3.4.1 Volumetric Representations

9.3.4.2 Skeletal Representations

9.3.4.3 Surface Representations

9.3.5 The Extended Gaussian Images

9.4 Visual Processes

9.4.1 Stereopsis

9.4.2 Camera Motion

9.4.3 Shading

9.5 Concluding Remarks

Computer Vision

Image Processing or Artificial Intelligence?

Why would anyone be interested in
studying Computer Vision ?

- All naturally-occurring intelligent life-forms exhibit an ability to interact with and manipulate their environment in a coherent and stable manner.
- This interaction is facilitated by on-going intelligent interplay between perception and motion control (*i.e.* action).

- Most manufacturers are concerned with the cosmetic integrity of their product; customers quite often equate quality of appearance with functional quality.

It is highly desirable therefore that :

- the product is checked visually before packaging and shipping,
- the Inspection Process be automated and effected without human intervention.

What is Computer Vision ?

The world we live in and experience is filled with an endless variety of objects and it is by looking and seeing that we come to know what is where in this world.

Vision is a means to an end :
to know the world by looking

Computer Vision is exactly the same except
that the medium by which the knowledge is
gained is now a computational instrument
rather than the brain of some living creature.

Machine Vision is a multi-disciplinary subject

- Optics
- Electronics
- Mechanical Engineering
- Computer Science
- Artificial Intelligence

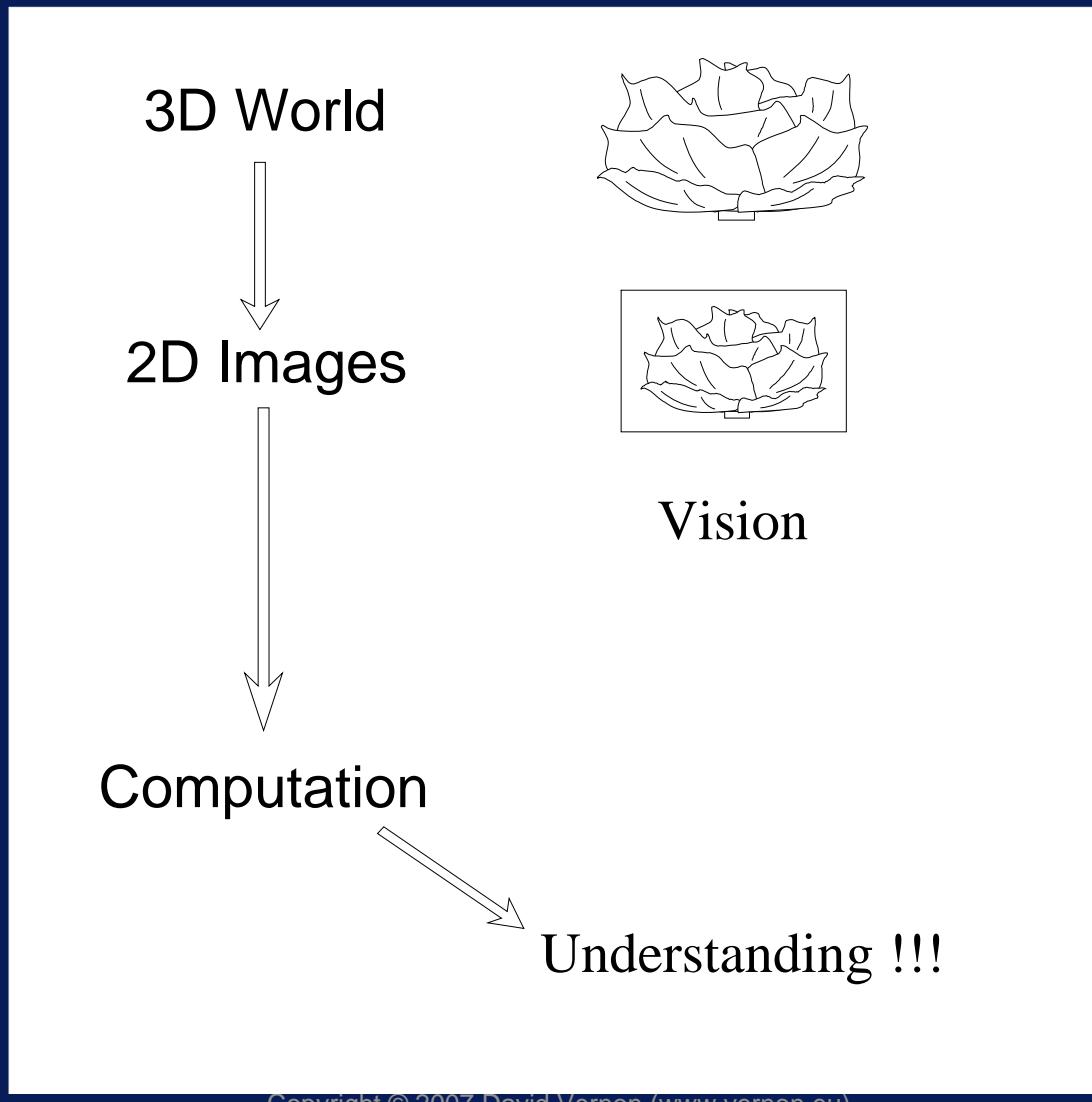
Computer Vision is concerned with the physical structure of a three-dimensional world by the automatic analysis of images of that world.

The image is two-dimensional. We inevitably lose information in the projection process, *i.e.*, in passing from a 3D world to a 2D image.

The images are digital images :

- they are a discrete representation (*i.e.* they have distinct values at regularly sampled points)
- they are a quantised representation
(*i.e.* each value is an integer value)

Computer Vision





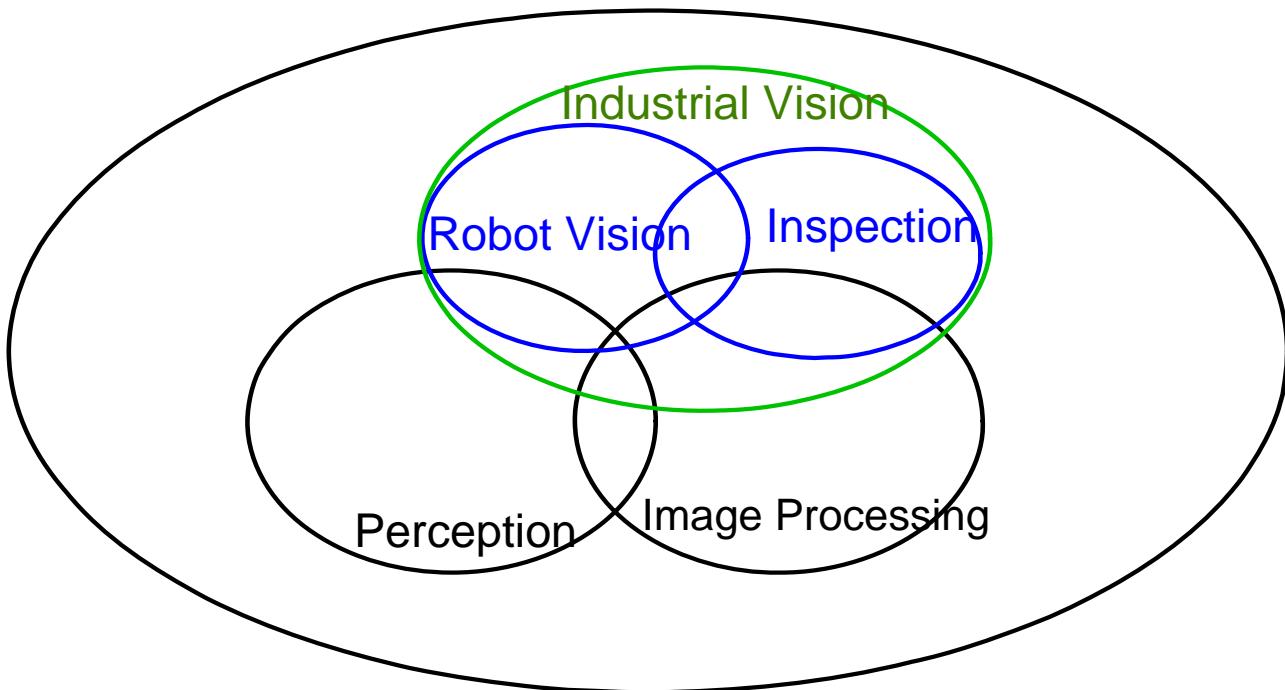
Copyright © 2007 David Vernon (www.vernon.eu)

Computer Vision includes many techniques

- Image Processing
- Pattern Recognition
- Description of Shape and of Volume
- Geometric Modelling
- Cognitive Processing

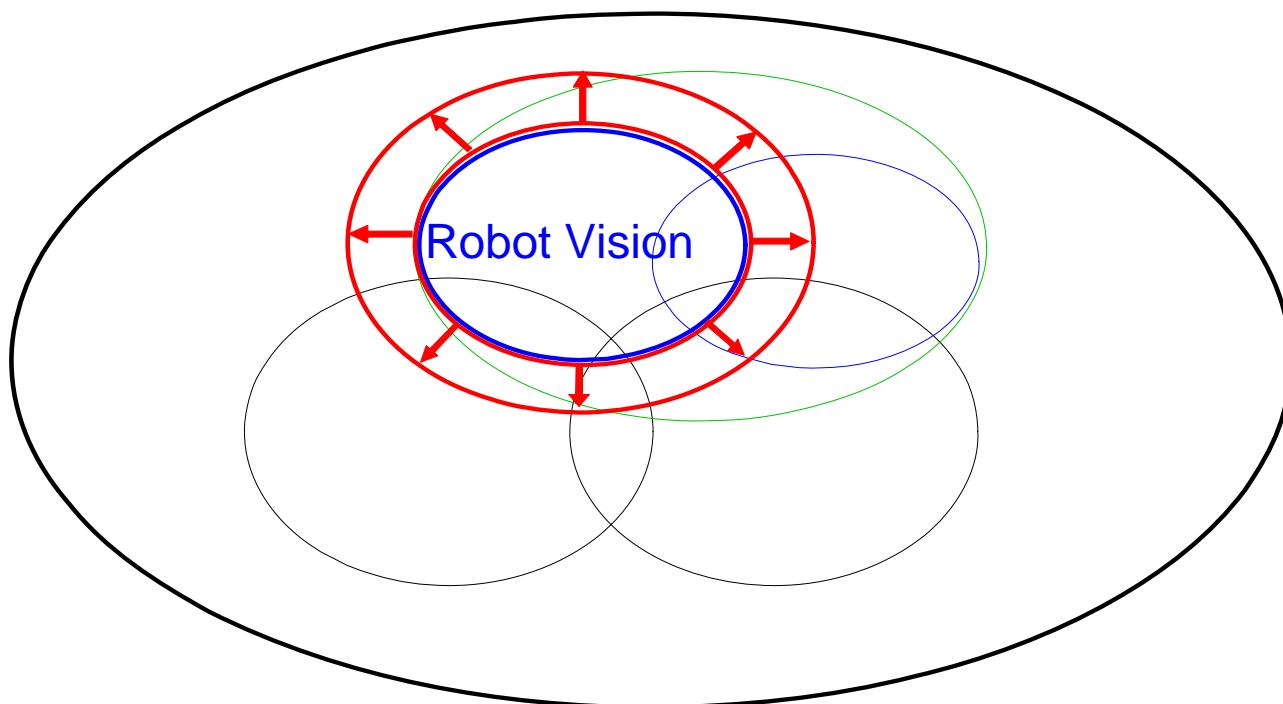
Although Computer Vision is certainly concerned with the processing of images, these images are only the raw-material of a much broader science which endeavours to emulate the perceptual capabilities of man.

Computer Vision



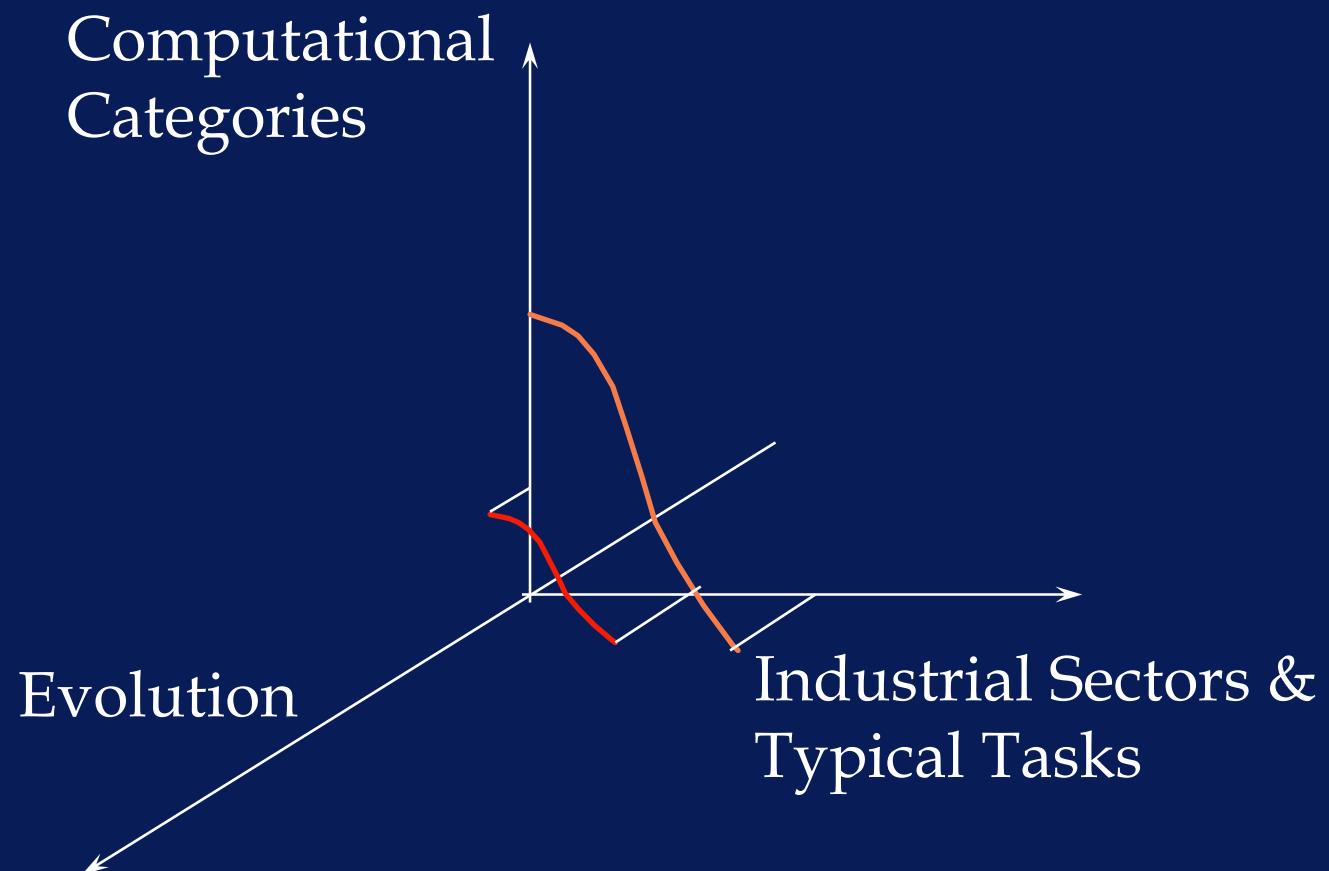
Trends in Robot Vision

Computer Vision



Applications and Techniques of Computer Vision

The Applications and Techniques of Computer Vision



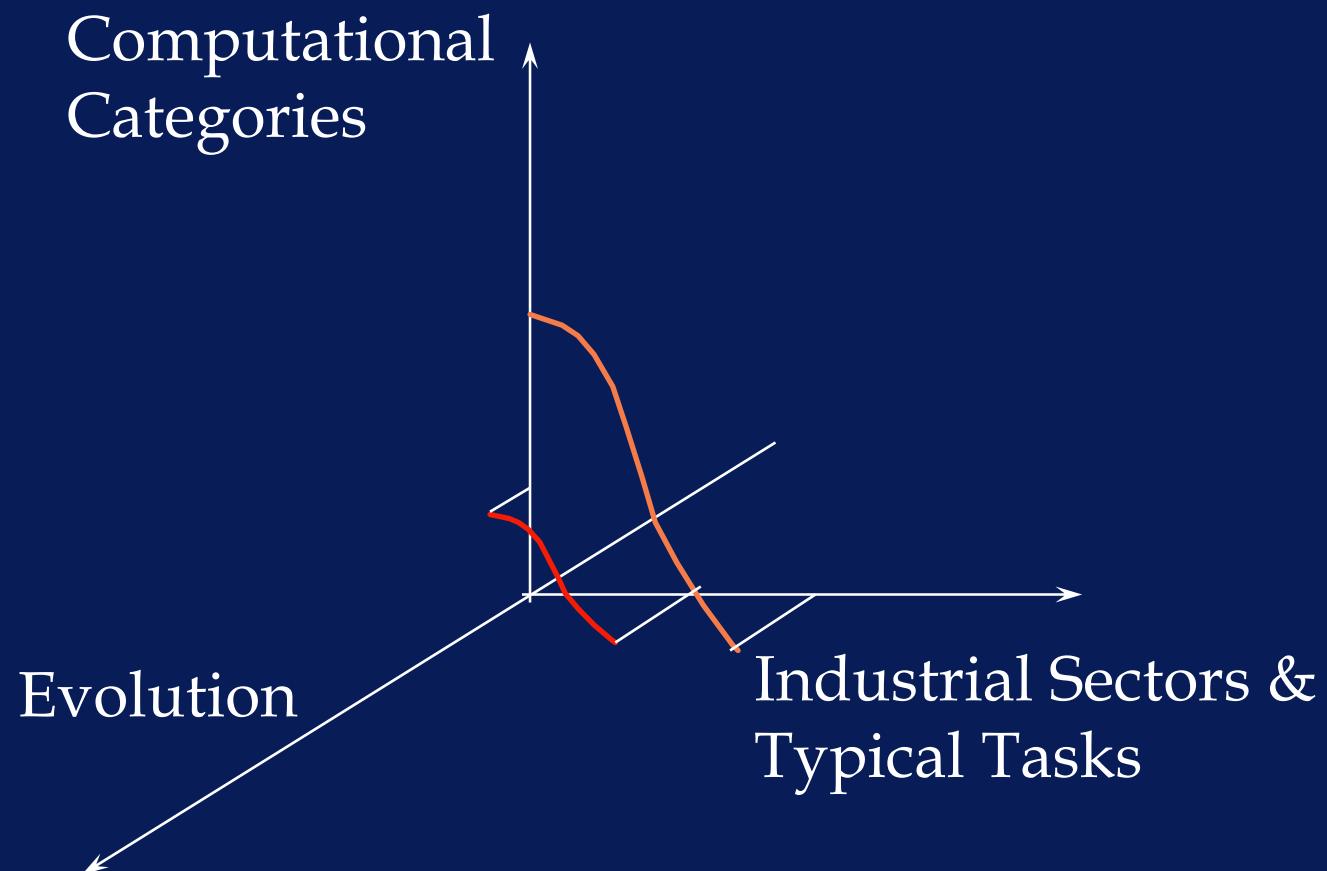
Industrial Sectors

- Telecommunications
- Entertainment
- Automation
- Transportation
- Agriculture
- Forestry
- Fisheries
- Slaughter
- Food processing
- Electronics
- Packaging
- Advertising
- Publishing and document processing
- Mining
- Mining
- Medical
- ❖ Pharmaceutics
- ❖ Health and Beauty
- ❖ Remote sensing & environmental surveillance
- ❖ Support for the disabled
- ❖ Security & surveillance
- ❖ Sports & recreation
- ❖ Energy
- ❖ Textiles
- ❖ Aerospace
- ❖ Education
- ❖ Domestic automation
- ❖ Construction
- ❖ Financial services
- ❖ Distribution and retailing
- ❖ Tourism and culture
- ❖ Post

Typical Tasks

- Inspection
- Localization, registration, metrology
- Document processing (OCR, bar code)
- Pattern, object, and event recognition
- Interpretation and recognition of motion, gestures, or facial expression
- Scene recognition, reconstruction, visualization, editing
- Visual control
- Image compression

The Applications and Techniques of Computer Vision



Computational Categories

- n-Dimensional Signal Processing
- Geometry and Shape
- Reflectance (including colour and texture)
- Time & Scene Dynamics
- Interpretation and Understanding
- Systems Design & Control

Scientific Topics

- Appearance-based vision (e.g. correlation and template matching)
- Visual signal processing (e.g 3-D and 4-D filtering)
- Image structure enhancement (e.g. morphological processing)
- Surface characteristics (texture, colour, ...)
- Coding and compression
- Model-based vision
- Shape description
- 3-D acquisition
- Metrology & non-contact measurement
- Theory of invariance
- Shape from shading
- Photogrammetry
- Dynamic vision (temporal phenomena)
- Active vision
- Vision-controlled action
- Visual motion

Scientific Topics

- Model-based vision
- Appearance-based vision
- Uncertainty analysis
- Recognition and interpretation
- Conceptual descriptions (e.g. knowledge based)
- Data fusion
- System design
 - Performance characterization
 - HCI for computer vision
- Sensors and computational platforms
- Co-design (of algorithm and implementation)

Scientific Topics	Geometry & Shape	Reflectance (including colour and texture)	Computational Categories			Interpretation & Understanding
			Time and Scene Dynamics	Systems Design and Control	n-Dimensional Signal Processing	
Model-based vision (CAR, VR)	X					X
Appearance-based vision					X	X
Dynamic vision (temporal phenomena...)			X			
Active vision (controllers)			X	X		
Vision-controlled action (closed loop)			X	X		
Image structured enhancement (filters)						X
Surface characteristics (texture, colour)		X				X
Shape description/analysis (3-D acquisition, geometry)	X					
Vision dynamics (temporal)			X			
Recognition and interpretation						X
Conceptual descriptions (knowledge, task-based)						X
Systems design (performance characterization, HCI for CV)				X		
Coding and compression						X
Metrology & contactless measurement	X					
Invariance	X	X				
Visual motion			X			
Visual signal processing					X	
Hardware/codesign/processors/storage/sensors/communications				X		
Uncertainty management and analysis						X
Colour		X				
Data fusion (multi-sensor, n-dimensional)					X	X

Industrial Machine Vision Vs. Image Understanding

Human live and work within a general three-dimensional world, pursuing

- many goals and objectives
- unconstrained and constantly changing environment
- many varied and, often, ill-defined objects.

Industrial Automation

- single repeated tasks
- relatively few objects
- all of which are known and defined
- manufacturing environments which are normally constrained and engineered to simplify those tasks.

Industrial systems do not yet work with
general three-dimensional environments.

Vision systems for manufacturing still exploit
many assumptions in order to facilitate
processing and analysis.

Approaches associated with general environments are frequently referred to as :

- *Image Understanding* or
- *Scene Analysis*

Vision systems specifically intended for the industrial environment are often referred to generically as *Industrial Machine Vision Systems*

Image Understanding Vision Systems

- Three-dimensional scenes,
- Partially constrained,
- Viewed from one (and often several) unconstrained viewpoints
- Illumination conditions may be known
- Usually one will have to contend with Shadows and Occlusion

- Scene Representation
- A two-dimensional image representation of a three-dimensional scene
- High spatial resolutions (*i.e.* it exhibits a large variation in grey-tone).
- Colour information is incorporated but not nearly as often as it should be.

Range data is sometimes explicitly available from active range sensing devices but a central theme of image understanding is the automatic extraction of both range data and local orientation information from several two-dimensional images using *e.g.*

- Stereopsis
- Motion
- Shading
- Occlusion
- Texture Gradients
- Focusing

Image Understanding utilises several redundant information representations

- Object Edges or Boundaries
- Disparity between objects in two stereo images
- Shading of the object's surface

It also incorporates different levels of representations in order to organise the information being made explicit in the representation in an increasingly powerful and meaningful manner.

Industrial Machine Vision System scenes are usually assumed to be two-dimensional, comprising known isolated rigid parts, frequently with a contrasting visual backdrop.

Lighting is almost always a critical factor and must be very carefully organised.

- Images are frequently two-dimensional binary images (pure black and white, with no intermediate grey-levels) of essentially two-dimensional scenes.
- Normally just one simple internal object representation or model; the analysis strategy being to extract salient features and to make some decision.

There are two complementary area of Industrial Machine Vision :

- Robot Vision
- Automated Visual Inspection

Visual Inspection tasks are, in general, not as difficult as those involved in visual perception for robotic parts manipulation, identification and assembly.

- Inspection Environment usually easier to control
- Accept decisions required for inspection are often easier to determine than the location and identification information needed for assembly.

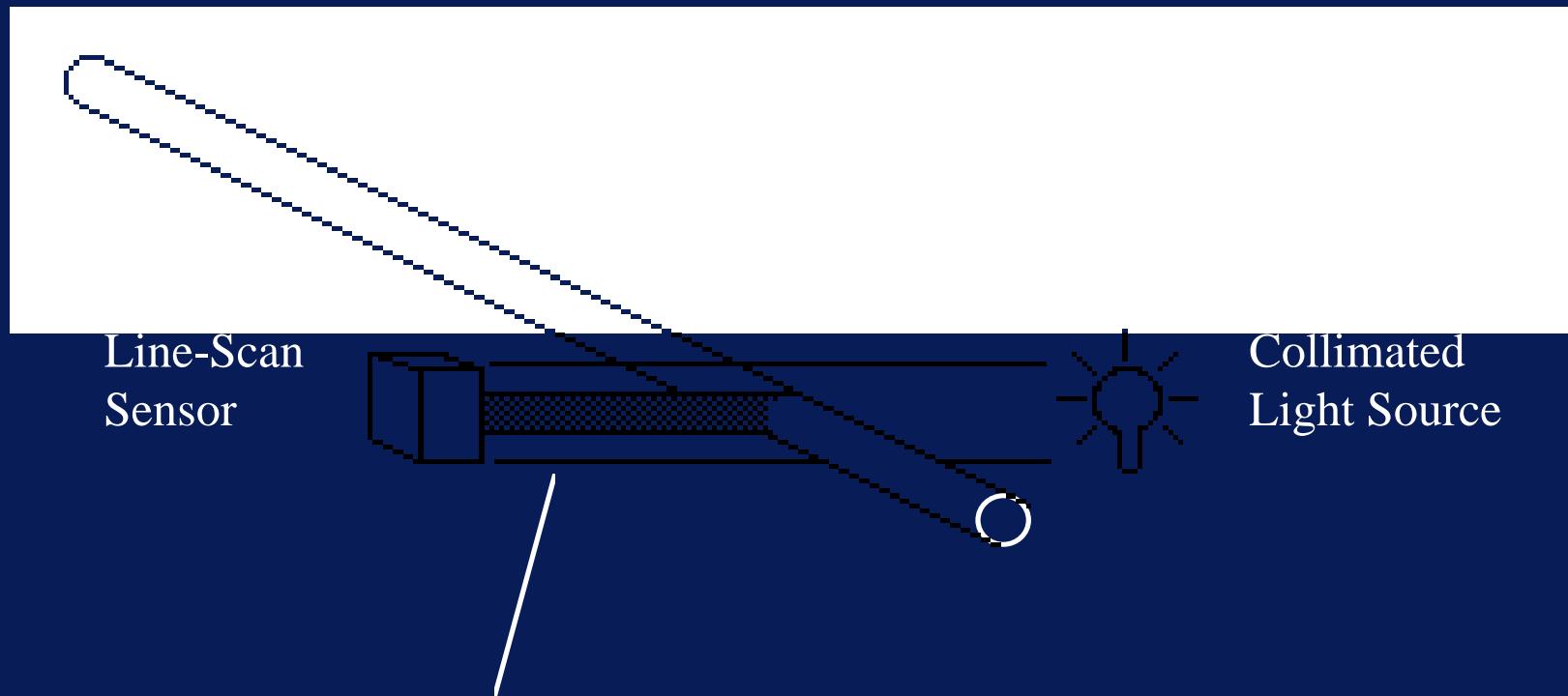
Visual Sensing is difficult

Outside Diameter Gauge

- Raw material for steel ingots to be used in a casting process is delivered in four-foot long bars with a circular cross-section.
- Individual ingots are then cut from the bar and stacked in a wooden box before being moved to the next stage of the casting process.

- To ensure high quality casting with minimal wastage of raw-material, it is essential that the weight of the charges fall within acceptable tolerances.
- If the cross-sectional area of the bar of raw material were constant, the required weight would be given by an ingot of fixed length.

- Ensure the correct weight of a cut charge, the cross-sectional area of the bar must be monitored.
- A length of bar is chosen such that the volume (and hence weight) falls within the required tolerance.



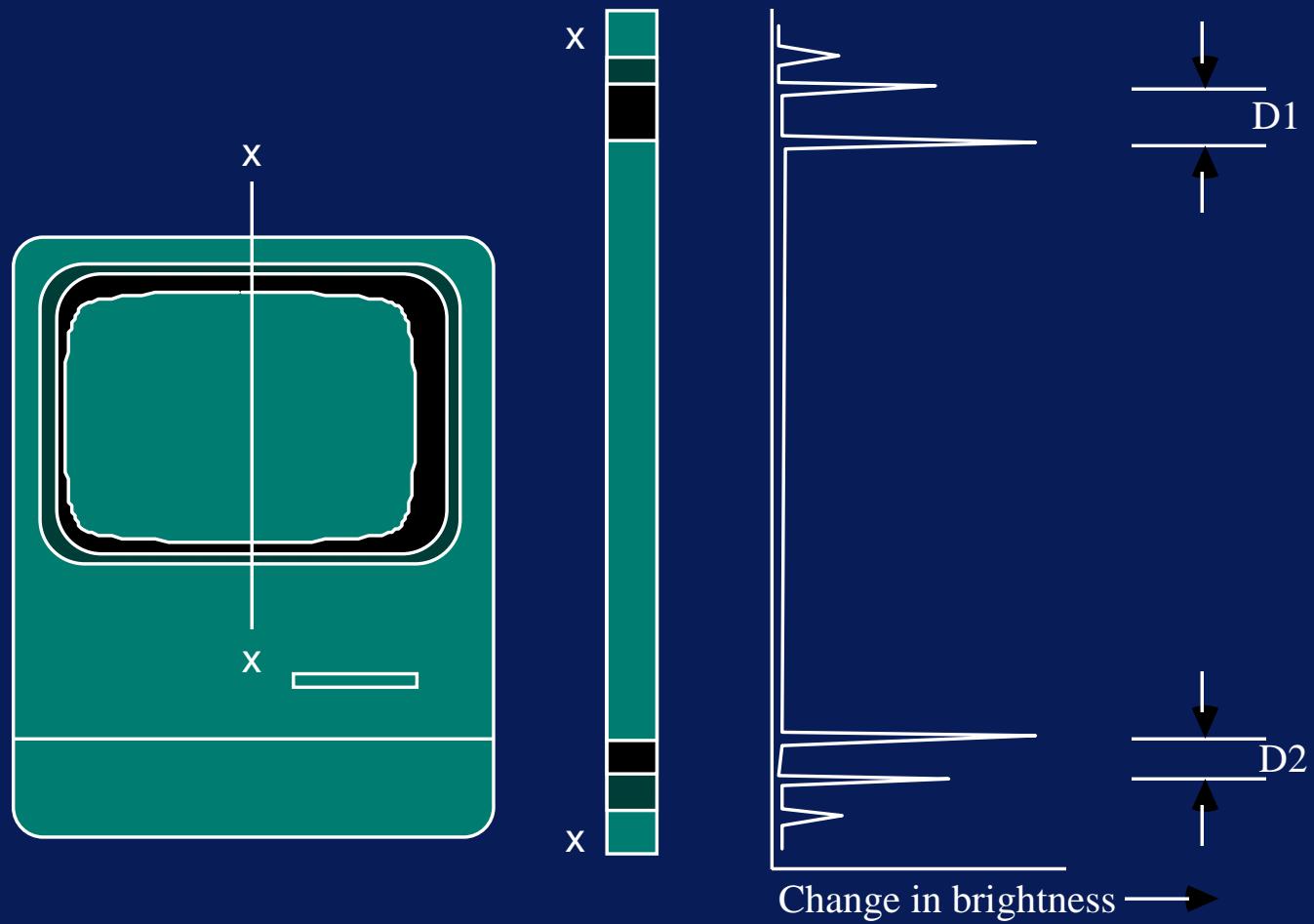
“Outside Diameter” gauge comprising a diametrically-opposed light source and Image sensor is mounted on robot end effector. This is moved along the axis of the bar.

One pass of the sensor generates a single signature describing how the diameter varies over the scanned length.

- Several passes of the sensor at different orientations (*i.e.* rotating the sensor about the axis of the bar after each pass) allow for non-circular cross-sections.

Inspections of Computer Screens

Cathode Ray Tubes (CRTs) must be inspected and calibrated during the assembly of personal computers. The picture position, width, height, distortion, brightness and focus need to be computed.



$$\text{Vertical eccentricity} = D1 - D2$$

Consider the problem of identifying the position of the picture on the screen

- Measure the distance between the edge of the picture (shown in white) and the inside edge of the computer face-plate, *i.e.* the width of the black border.
- We sample a small linear section.
- Identify the transition between white and black and between black and grey.

- This is accomplished by estimating the rate of change in intensity or colour along the strip.
- The required distances, D_1 and D_2 , correspond to the distance between the first two peaks we encounter as we journey from the centre of the strip to the periphery.

- The CRT is adjusted until these two distances are, to within specified tolerances, equal.

Typical System Architecture

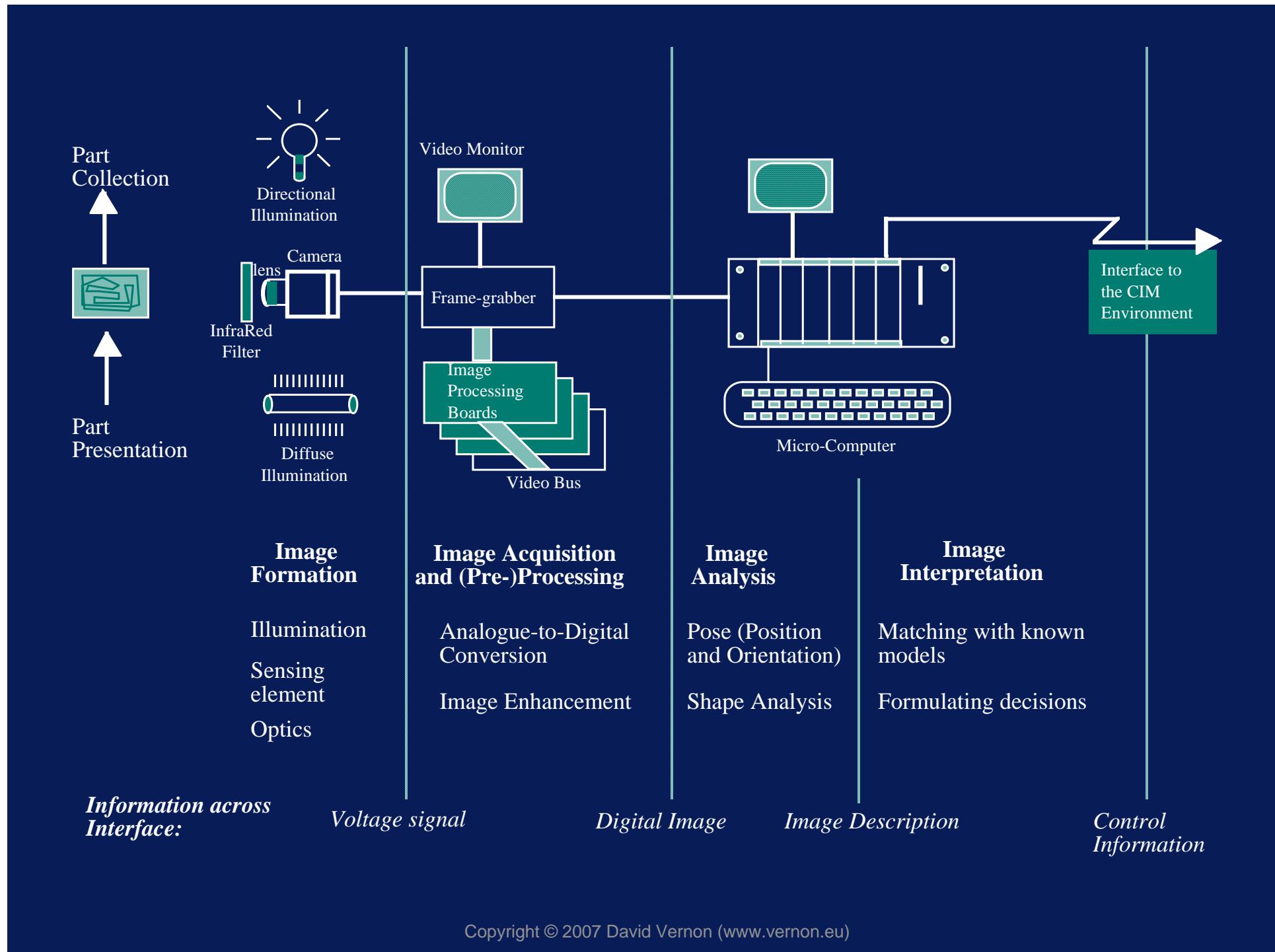


Image formation system

- part illumination
- sensing element
- associated optics

is critical to the successful deployment of the system. This system generates an analogue representation of the image scene, typically in the form of a conventional T.V. video signal.

The task of the image acquisition and processing sub-system is

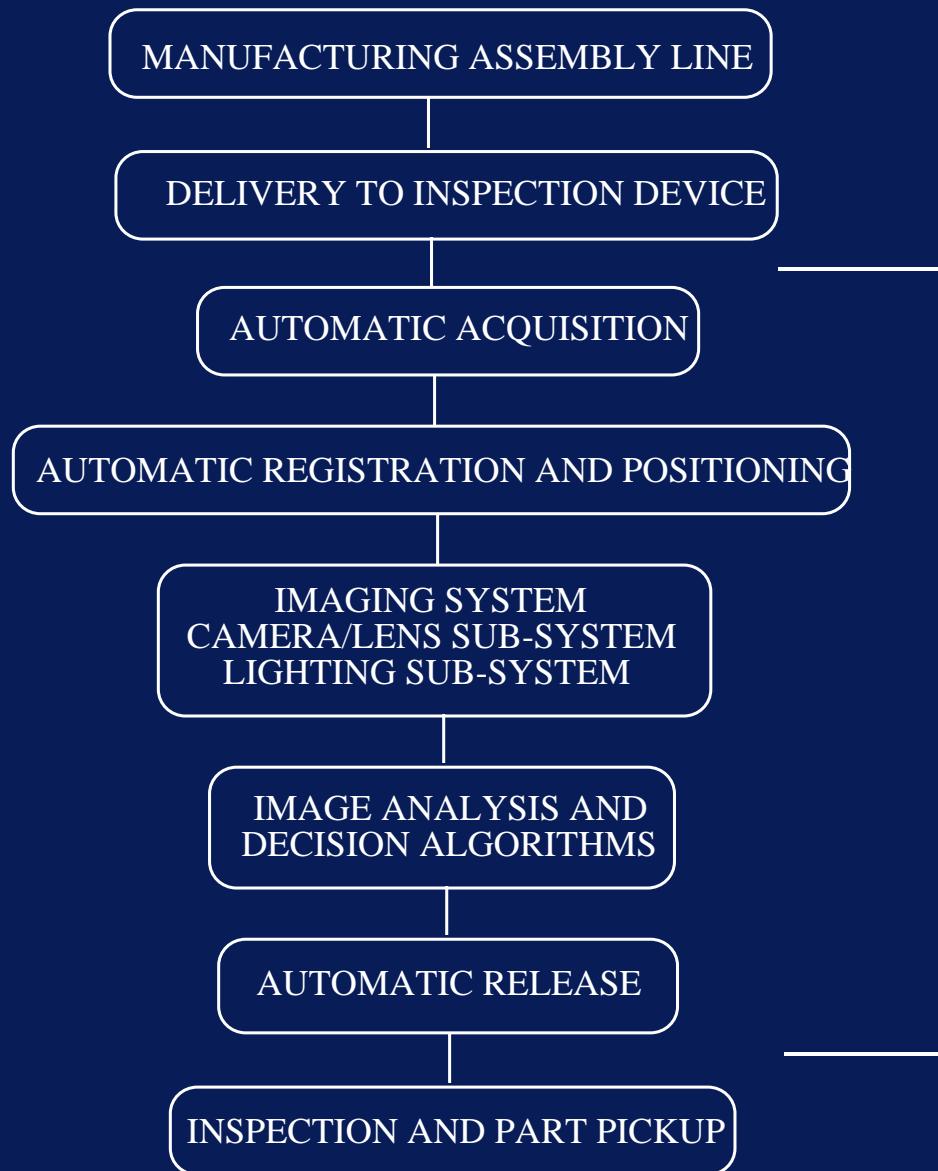
- to convert this signal into a digital image.
- to manipulate the resultant image to facilitate the subsequent extraction of information.

The image analysis phase is concerned with the extraction of explicit information regarding the contents of the image (*e.g.* object position, size, orientation).

There is a fundamental difference between image processing and image analysis.

- The former facilitates transformations of images to (hopefully, more useful) images.
- Image Analysis facilitates the transformation from an image to an explicit (symbolic) piece of information.

Image Interpretation is concerned with making some decision: it provides the link back to the environment to control the inspection process of the robot manipulator.



Automated
Visual
Inspection
System

Illumination & Sensors

Illumination

Scene and object illumination play a key role in the Machine Vision process.

The central purpose is to visually enhance the parts to be imaged so that their flaws, defects and features are highlighted.

The common is the incandescent bulb

- cost effective
- easily adjusted for light intensity
- generally provides directional illumination since it is, to an approximation, a point source of light
- incandescent bulbs cast strong shadows which invariably cause problems for machine vision software.

- Special bulbs are normally required as degradation in emitted light intensity with age is common
- Incandescent Bulbs emit considerable infra-red radiation
- CCD Cameras are sensitive and visual data can be washed out by the reflected IR rays.

For most machine vision applications, a diffuse source of light is the most suitable.

Diffuse lighting

- non-directional
- produces a minimum amount of shadows

Fluorescent lighting is the simplest and most common method of obtaining diffuse lighting.

*Back-lighting provides high contrast between the object and the background upon which the object rests and facilitates very simple **Object Isolation or Segmentation**.*

In some manufacturing environments, it is necessary to inspect moving objects

- “freeze” the motion of the object for an instant by the use of a strobe light or electronic flash
- the lamp emits a short (1 ms or less) burst of light, thus the moving object is illuminated for a very short period and appears stationary.

- the activation of the strobe must be synchronised with the acquisition of the image.
- exploit cameras with a very short “shutter speed” or, rather, the electronic equivalent, a very short exposure time.

This exposure is usually referred to as the integration time since it is the period over which the sensor integrates or averages the incident light.

One would normally choose the latter option of a short integration time, since it is more ergonomic and less disruptive for humans.

Control of Illumination and Light Levels

- If image processing and analysis decisions are being made on the basis of a fixed intensity datum (threshold) then some problems will occur if the illumination, and hence the reflected light, changes.
- If possible, the vision system should be able to adapt to such changes.

- This does not necessarily mean that it should be capable of dealing with dynamic changes.
- Most illumination systems degrade quite slowly over time and it would be quite satisfactory if the system were capable of self-calibration at the beginning of each day.

Alternatives

- Ensure that the illumination does remain constant
- Aperture of the camera lens might be altered

- Electronically-controlled aperture lenses (so-called auto-iris lenses) should not be employed directly:
 - function is to alter aperture so that the average amount of light passing through the lens remains constant
 - not appropriate for machine vision systems as the grey-tone shade of a particular feature would vary, depending on the intensity of the ambient lighting.

- Mains-powered lighting is inherently “flickery” due to the AC characteristics of the mains supply.
- Humans don’t notice this because they “integrate” (or average) the incident illumination over a short period of time.

- Machine Vision sensors do not integrate in quite the same way and, when they acquire the image, the flicker can become apparent.
- The use of an appropriate (DC, say) power supply can alleviate this problem, when it does occur.

Sensors

The task of a camera system is to convert an optical picture, into some form suitable for use with electrical systems.

Since the camera represents the direct link between the environment being examined and the information processing system, it plays a particularly significant role.

Lenses are required to focus part of the visual environment onto the image sensor.

It is possible to construct an imaging system without lenses, but such a configuration is not typical of the requirements of a vision system.

Lenses are defined by :

- their *Focal Length* (quoted in millimetres)
- their *Aperture* (the *f* number).

These parameters determine the performance of the lens in terms of light gathering power and magnification, and it often has a bearing on its physical size.

The Focal Length of a lens is a guide to the magnification it effects and its field of view.

Selecting the Focal Length which is appropriate to a particular application is simply a matter of applying the Basic Lens Equation :

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$$

where

v is the distance from the lens to the image,

u is the distance from the lens to the object,

f is the focal length

Noting the *Magnification Factor* M is

$$M = \frac{\text{image_size}}{\text{object_size}}$$

and equivalently,

$$M = \frac{\text{image_dist.}}{\text{object_dist.}}$$

Thus

$$f = \frac{uM}{M + 1}$$

Hence if we know the required magnification factor and the distance from the object to the lens, we can compute the required focal length.

For example, if a 10cm wide object is to be imaged on a common 8.8 * 6.6mm sensor from a distance of 0.5m, this implies a magnification factor of :

$$M = \frac{8.8}{100} = 0.088$$

So,

$$f = \frac{500 * 0.088}{1.088} = 40.44$$

Thus, we require a 40.44 mm focal length lens.

Typically, one would use a slightly shorter focal length (*e.g.* 35 mm) and accept the slight loss in resolution due to the larger field of view.

The ***minimum focal distance*** is the minimum distance between the front of the lens and the object at which the object can still be focused.

Generally speaking, lenses with short focal lengths have smaller minimum focal distances.

If the lens is required to focus a relatively small object at short distances, the minimum focal distance (typically 300mm) may be too large.

In that case, an extension tube can be used to increase the distance, v , of the lens from the sensor and hence decrease the distance, u , from the lens to the object.

For a given focal length, f , the lens equation stipulates that u *decreases* as v *increases*, if the image is to remain in focus.

The ***f-number*** is measure of the amount of the light allowed to pass through the lens and is a normalised measure of lens aperture.

It is defined by the focal length divided by the diameter of the aperture.

The standard scale is 1.4, 2, 2.8, 4, 5.6, 8, 11, 16; each increase reduces the amount of light passing through the lens by one half.

The ***depth of field*** is the distance between the nearest and the furthest points of a scene which remain in acceptable focus at one aperture setting.

In general, the depth of field gets larger as the focused point moves away from the lens (given constant aperture and focal length).

Also, the depth of field increases significantly as the aperture closes (*i.e.* as the f number increases) for any given focusing distance.

Lenses have many types of standard mounts, for example, the Pentax, Olympus, Nikon bayonet mounts, but the standard on television and CCTV (Closed Circuit TV) cameras is a screw mount called the C mount.

35 mm photographic lenses are usually much more bulky than the miniature lenses which are specifically designed for CCD cameras.

- photographic lenses are designed to image a 35mm format and, for a given focal length, the optical surface must be much larger.
- CCD sensors (as we will see in the next section) are typically less than 10 mm in width and hence the optical surface can be much smaller.

Many of the visual problems which cause difficulty interpreting a scene can often be solved by the use of simple filters on the camera lens.

Filters are often used to reduce, or remove artefacts; polaroid sun-glasses are probably the most widely used ‘filters’ and are used to reduce glare.

In machine vision, one of the most annoying and potentially disruptive problems is that due to specular reflections (mirror-like reflection on shiny objects).

- The use of a simple polarising filter on the camera lens can often reduce the effect of these reflections.

Some sensors are sensitive to segments of the electro-magnetic spectrum which do not convey visual data, *e.g.* IR radiation.

The use of a simple IR blocking filter can solve this problem in a simple and effective manner.

Camera Sensors

Two types of video camera available :

- the vidicon, is based on vacuum-tube technology
- the other is based on semi-conductor technology.

A Vidicon Tube

- is a photo-conductive device
- which employs a photo-sensitive sensor layer
- consisting of several million mosaic cells
- insulated from one another on a transparent metal film (refer to Slide 43).

Each cell represents a small capacitor whose charge is a function of incident light.

The sensor layer is scanned in a raster format with an electron beam over 625 lines in accordance with the television standard.

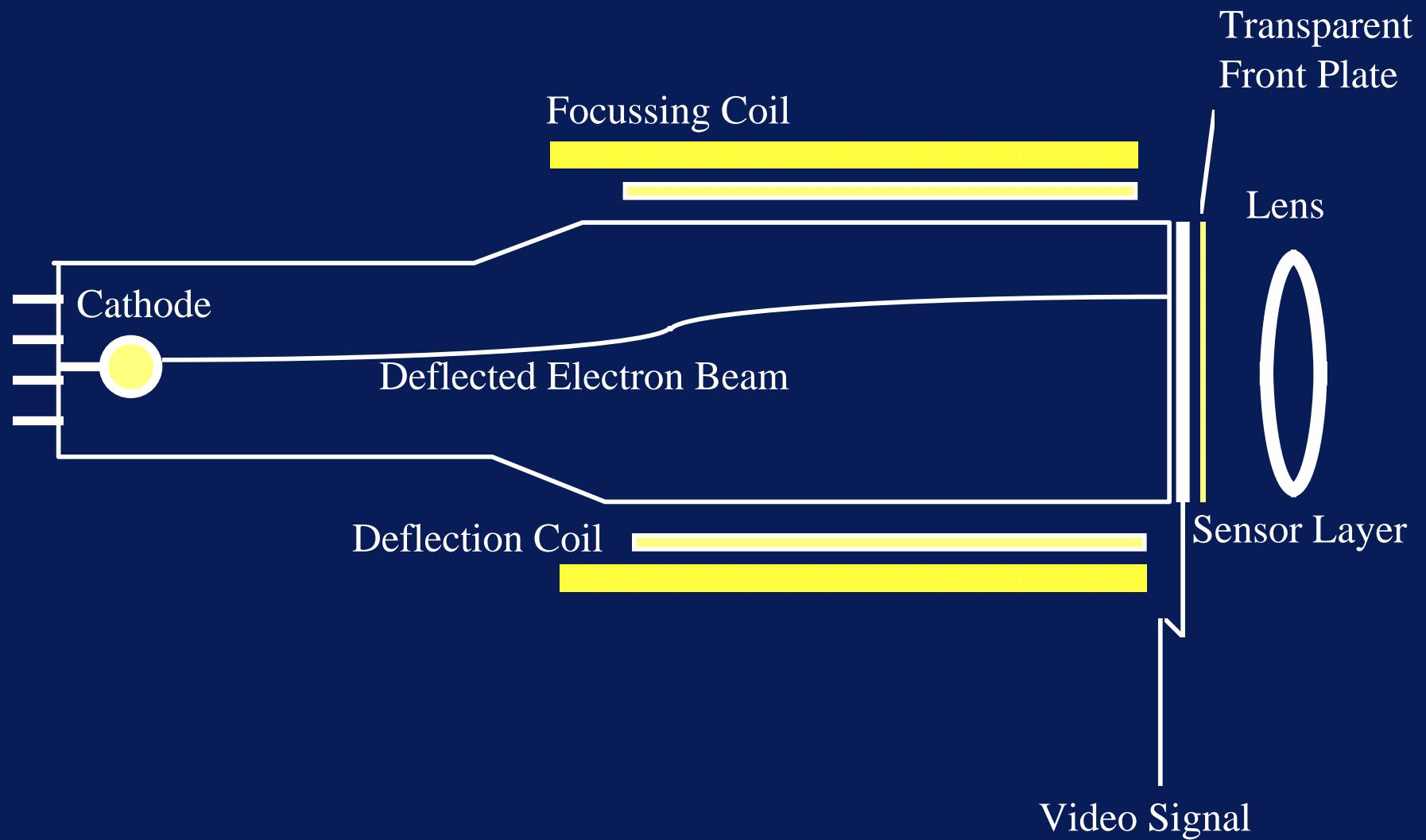
This beam is deflected magnetically by a set of coils outside the tube bulb.

The electron beam makes up charge lost through the incident light in individual mosaic cells and so generates the video signal at the sensor element.

This video signal is simply a continuous analogue signal proportional to the light intensity of the focused image.

The camera electronics insert synchronisation pulses (syncs.) to indicate scan lines, fields and frame ends.

- The standard vidicon has a sensor element comprised of Antimony Sulphide, Sb_2S_3
- The silicon diode vidicon has a sensor element made from Silicon, Si.
- the plumbicon has a sensor element made of Lead Oxide, PbO.



*Most solid-state cameras are based on
Charge-Coupled Device (CCD) technology,
though there are several variations on the
theme.*

- The basic structure of CCD technology is that of an analogue shift register consisting of a series of closely spaced capacitors.
- Charge integration (accumulation) by the capacitors, photosites, caused by the photons comprising the incident light, provides the analogue representation of light intensity.

- At the end of the integration period (exposure time) these charges are read out of the sensor.

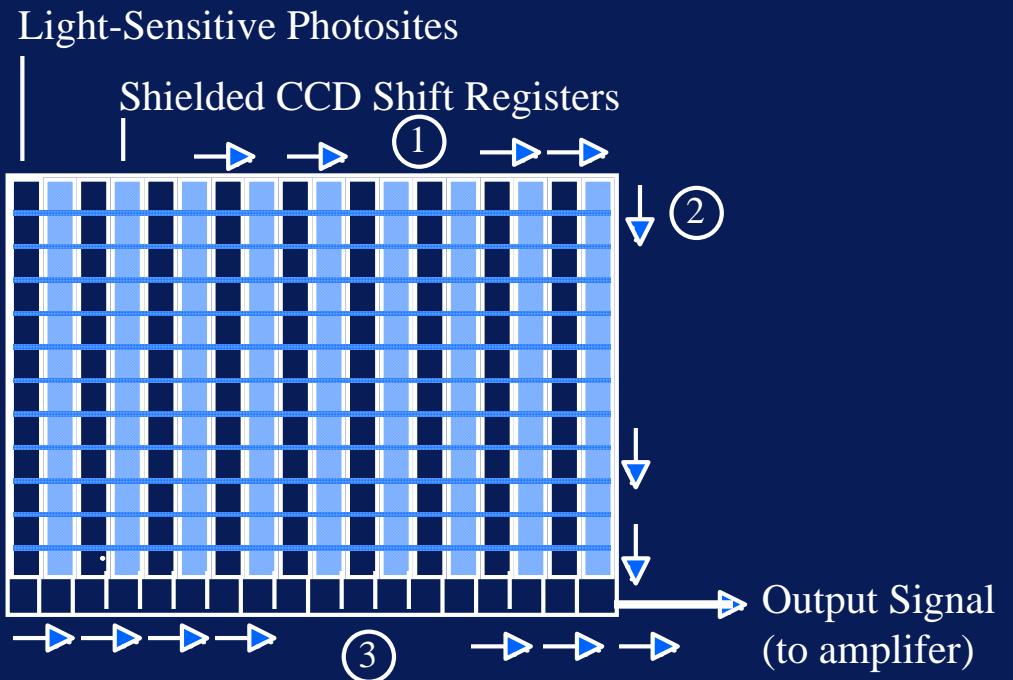
CCD sensors most commonly use one of three addressing strategies :

- Interline transfer
- Frame transfer
- Column-row transfer

Interline Transfer CCD

- Organised into column pairs of devices
- An imaging column of photosensors is adjacent to an opaque vertical shift register (see Slide 49)
- Charge accumulates in the imaging column until the end of the integration period.

- ① Charges are shifted to the shielded area
- ② The charges in the column are shifted down one cell
- ③ A row of charge is then shifted out



- when it is transferred to the opaque column
- The signal then shifts vertically into a horizontal shift register that represents the picture sequentially line by line.

The advantage of the interline transfer is that the transfer time (to opaque storage) is short compared to the integration period.

This is desirable because when transfer time approaches the integration time, solid-state sensors tend to exhibit a locally-contained spreading of the image response, called *Smear*.

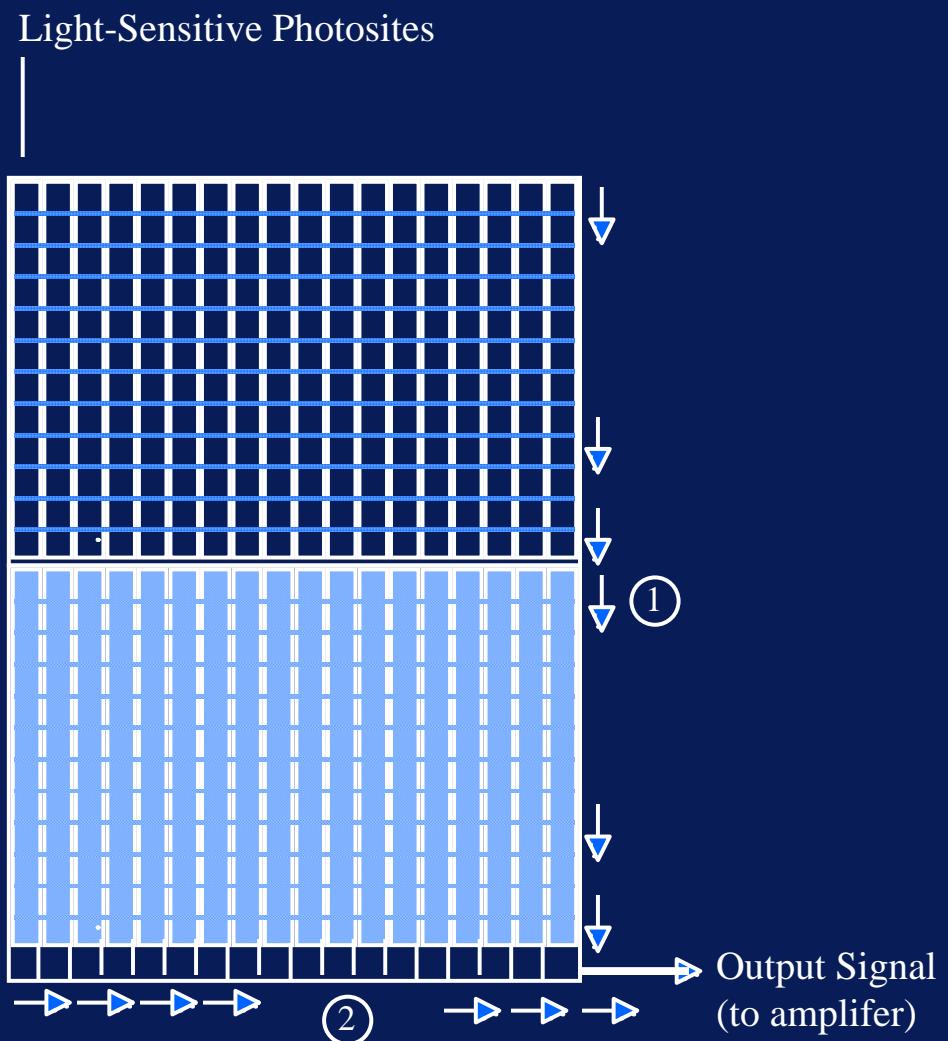
Thus, the Interline Transfer minimises Smear.

Frame Transfer Organisation

- Sensor consists of vertical columns of CCD shift registers divided into two zones
- One zone, where charge accumulates during integration time, is photosensitive.
- When integration is complete, the whole array is transferred in parallel to the opaque storage area of the second zone.

- ① All charges are shifted down to the shielded area
- ② Each row of charge is then shifted out

Shielded CCD Shift Registers —



X-Y Addressing

- The sensor elements are addressed by selecting individual column and row electrodes.
- Charge collected under the column electrode is transferred to the row electrode and amplified for output.

Linear Array Sensor or Line-Scan Sensor

- These sensors are simply a 1D array (row) of photosites.
- Use exactly the same technology as the 2D array sensors.

They differ in two important characteristics :

- These sensors can have between 256 and 4096 photosites in a row and, hence, can achieve much greater resolution than state-of-the-art array cameras.

- » They are inherently 1D devices. They can only take pictures of slices of a 2D scene and if a 2D image is required, several such slices must be acquired.
- » These sensors are best suited to the inspection applications in which the scene to be scanned is in continuous linear motion (or, indeed, where the camera itself can be translated)

- The video signal that they generate does not correspond to any particular video standard and what is produced is essentially a time varying analogue voltage which represents the incident light along the line of photosites.

- » most systems which use line-scan cameras tend to have custom designed computer interfaces
- » matched line-scan cameras and digitisers, and indeed, line-scan digitisers are now appearing on the market.

Camera Interfaces and Video Standards

The monochrome video signal standard used in the USA and Japan is RS-170, a subset of the NTSC (National Television Systems Committee) standard.

Europe uses the international CCIR (International Radio Consultative Committee) standard, which is similar to, but incompatible with RS-170.

Approximately fifty pictures per second are necessary for a flicker-free television picture.

As the generation of a complete new picture every 1/50th of a second (20ms) would place a severe load on the capabilities of the video amplifier so, in an attempt to obtain a flicker free image :

- the odd lines of a picture are scanned first
- the following 1/50th of a second, the even lines are scanned.

This means that a complete picture is established in 1/25th of a second.

This procedure whereby every alternate line is skipped in each field is called ***Interlaced Scanning***.

Thus, the number of complete pictures per second, *i.e.* the *Picture Frame Frequency* is 25Hz.

The *Raster Field Frequency* is 50Hz.

Each frame comprises
625 lines in the CCIR standard
(525 in the NTSC RS-170 standard)
thus, with 25 pictures of 625 lines,
 $625 * 25 = 15625$ lines will be scanned in a
second.

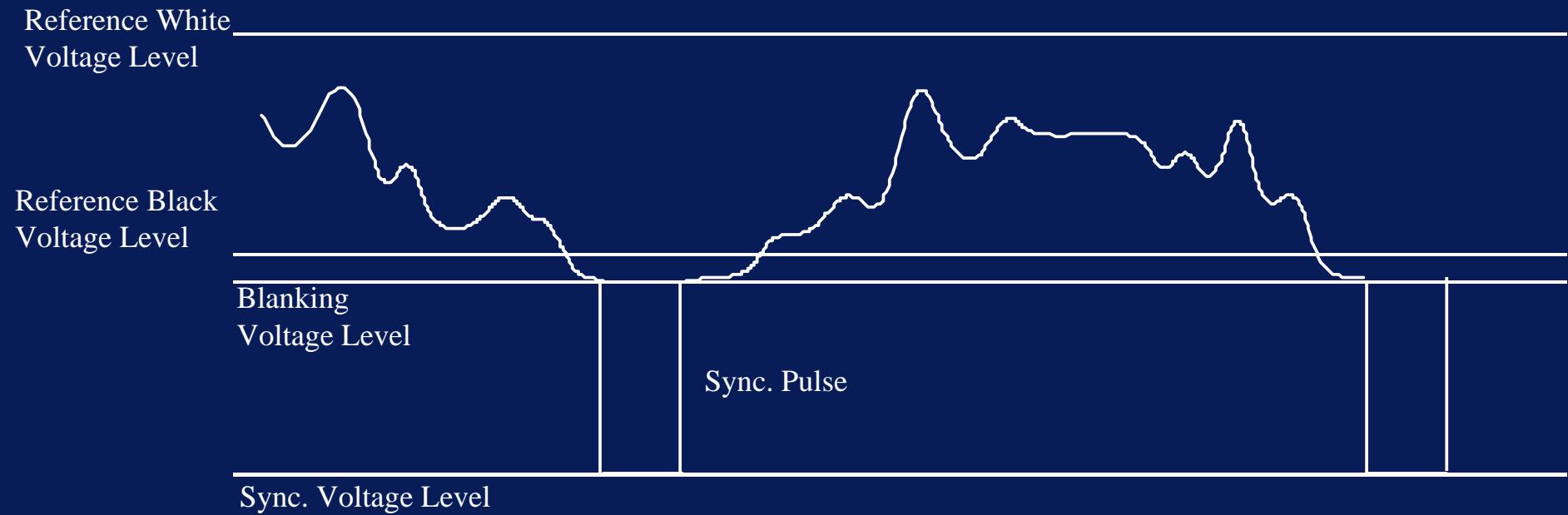
The so-called ***Line Frequency, f*** , is, hence,
15625 Hz.

Signal from a Line comprises

- picture information
- a synchronising pulse marking the end of a line
(horizontal sync. pulse)
- and a blanking period.

The time available for one line is :

$$T = \frac{1}{f} = \frac{1}{15625} s = 64 * 10^{-6} s = 64 \mu s$$



Composite Video Signal

11.5 μ s of this are used for the blanking and synchronising signal.

- This consists of a 6.5 μ s *porch* blanking signal
- a 5 μ s sync.

The sync. pulse signals the beginning of one line and the end of another.

The porch represents a quiescent voltage level which prevents the beam appearing as a bright line during the line flyback (when being displayed on a video monitor).

The end of the picture consisting of 625 lines,
i.e. the frame, is characterised by several
picture pulses which are significantly different
from the line pulses.

During these pulses, the picture is blanked
and the beam returns to the top of the picture
for the next frame.

However, image acquisition devices often have a 1:1 aspect ratio

- the rectangular video picture is effectively squeezed into a square with a consequent geometric distortion.

This has a very severe repercussions for vision based gauging applications.

Unfortunately for machine vision, the CCIR standard specifies a 4:3 horizontal-to-vertical aspect ratio for video signals.

The television monitor has a similar aspect ratio and, thus, an image of a square will appear square.

Characteristics of Camera Sensors

Resolution

- the purpose of the camera sensor is to convert an incident analogue optical image to some electrical signal
- the resolution of a sensor can be defined as the number of optical image elements which can be discriminated by that sensor and represented by the resulting signal.

- The resolution is limited by the number of photosites in the sensor since this defines the frequency with which the optical image is sampled.

Note that in solid-state cameras the effective resolution of the sensor is approximately half that of the number of photosites in any given direction.

Resolution is also limited by the array geometry and by how much opaque material separates the photosites.

- Interline transfer sensors have poorer effective resolution than frame transfer sensors due to the presence of shielded buffers between each photosensitive line.

The resolution of the camera system is also constrained

- by the limitations imposed by the CCIR video standard (see section 2.2.3)
- by the sampling frequency of the analogue-to-digital converter which converts the video signal to a digital image.

Blooming

- If the sensor of a television camera is subjected to intensive brightness, then the excess charge carriers spread into neighbouring zones and light is registered there also.

- *Thus, a thin beam of very bright light will be sensed over an area considerably larger than the cross-sectional area of the beam.*
- This effect is called **Blooming** and is especially noticeable with specular (mirror-like) reflections

Noise

- Noise, unwanted interference on the video signal, is defined quantitatively by the signal-to-noise ratio (S/N),
- the ratio of the amplitude of the wanted signal to the average amplitude of the interference

- For television cameras, the signal-to-noise ratio is defined as the peak-to-peak video signal divided by the effective amplitude of the noise.

If one follows general practice in telecommunications and expresses the ratio of electrical variables of the same unit in logarithmic terms

- one can compute the level, defined as 20 times the decimal log of the ratio of the linear variables and expressed in decibels (dB).

Thus, a signal to noise ratio of 20dB means that the picture quality is very bad (20dB implies that $\log_{10}(S/N)$ equals 1 and, thus, the signal to noise ratio is 10:1).

For satisfactory quality, a signal to noise ratio of 40dB or more is required.

Image Acquisition and Image Representation

Any visual scene can be represented by a continuous function (in two dimensions) of some analogue quantity.

This is typically the reflectance function of the scene: the light reflected at each visible point in the scene.

Such a representation is referred to as an image:

the value at any point in the image corresponds to the intensity of the reflectance function at that point.

Digital images represent the reflectance function of a scene but they do so in a

- *Sampled* and *Quantised* form

They are typically generated with

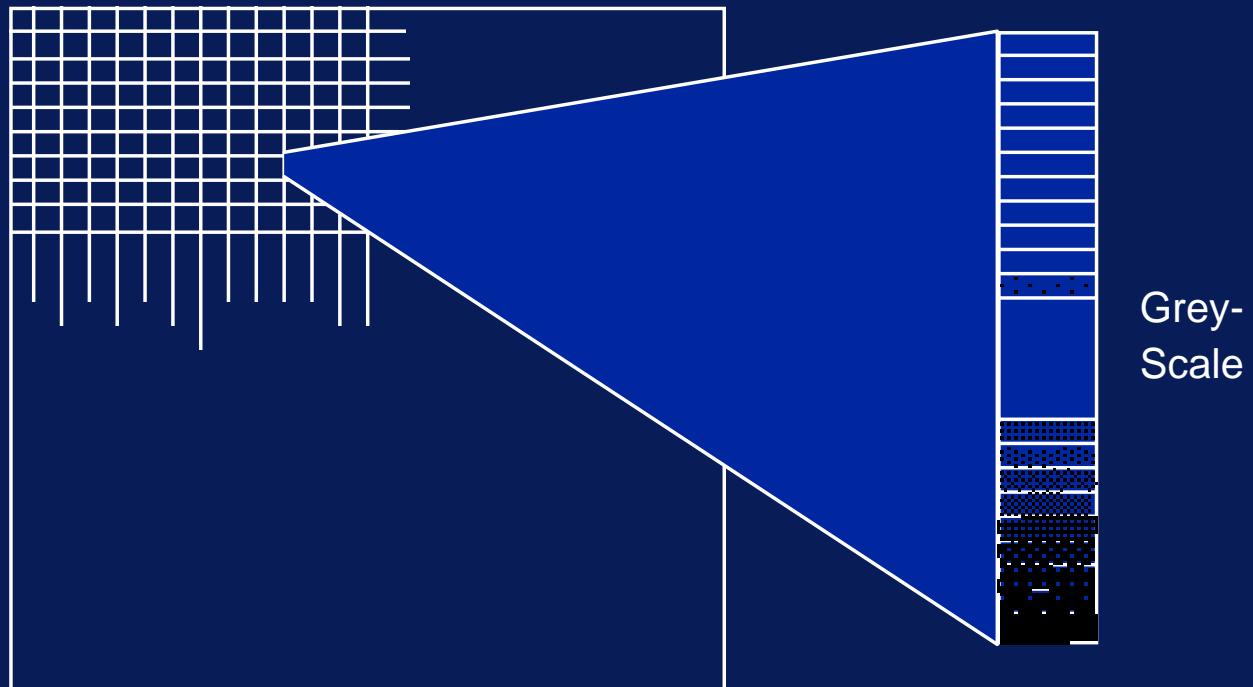
- some form of optical imaging device (*e.g.* a camera) which produces the analogue image (*e.g.* the analogue video signal discussed in the previous chapter)
- an analogue-to-digital converter: this is often referred to as a *digitiser*, a *frame store* or *frame-grabber*.

The Frame-grabber

- samples the video signal in some predetermined fashion (usually in an equally-spaced square grid pattern)
- quantises the reflectance function at those points into integer values called grey-levels

*Each Integer grey-level value is known as a **Pixel** and is the smallest discrete accessible sub-section of a digital image.*

Rectangular Sampling Pattern



Sampling and Quantisation

The number of grey-levels in the (equally-spaced) grey-scale is called the *Quantisation* or *Grey-scale Resolution* of the system.

In all cases, the grey-scale is bounded by two grey-levels, black and white, corresponding to zero intensity and the maximum measurable intensity, respectively.

Most current acquisition equipment quantises the video signal into 256 discrete grey-levels, each of which are conveniently represented by a single byte.

In certain cases, a grey-scale of -127 to +128 is more convenient.

The sampling density, the number of sampling points per unit measure, is usually referred to as the (spatial) resolution.

It is measured in terms of the number of sampling elements along each orthogonal axis.

This normally corresponds to the extent of the number of pixels in both the horizontal and vertical directions.

Most current commercial frame-grabbers have spatial resolutions of 768*576 pixels.

In summary, digital image acquisition equipment is essentially concerned with

- the generation of a two dimensional array of integer values
- representing the reflectance function of the actual scene
- at discrete spatial intervals
- this is accomplished by processes of *Sampling* and *Quantisation*

Thus, there are 3 factors which can limit the effective resolution and fidelity of the final digital image :

1. the sensor sampling frequency;
2. the bandwidth of the video signal;
3. the sampling frequency of the analogue-to-digital converter, *i.e.* the frame-grabber.

Spatial Frequencies

High frequency signals, such as high pitch soundwaves, periodically change their value over a very short period of *time*.

Spatial Frequencies

Similarly, a high spatial signal, such as an image, periodically changes its value (*e.g.* its intensity or grey-level) over a very short *distance*,

- it changes abruptly from one grey-level to another.

Spatial Frequencies

Conversely, low spatial frequencies correspond to “slower” changes in intensity where the change occurs gradually from one position in the image to another.

Spatial Frequencies

Consider, a spatially-unbounded analogue optical image $g(x,y)$. The ***Fourier Transform*** of the image $g(x,y)$ is defined $G(f_x, f_y)$

$$\begin{aligned} G(f_x, f_y) &= \mathfrak{I}(g(x, y)) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-2\pi i(f_x x + f_y y)} dx dy \end{aligned}$$

Spatial Frequencies

The Inverse Fourier Transform of $G(f_x, f_y)$
is defined by :

$$g(x, y) = \mathcal{I}^{-1}\left(G(f_x, f_y)\right)$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(f_x, f_y) e^{2\pi i (f_x x + f_y y)} df_x df_y$$

Spatial Frequencies

The variables f_x and f_y which identify the domain over which $G(f_x, f_y)$ is defined are known as the spatial frequencies in the x and y directions, respectively,

- domain is known as the *Spatial Frequency* domain.
- What do these variables and this domain represent?

The integral given by equation 3.2 can be viewed as a ‘generalised sum’ of complex exponentials, defined in terms of the spatial frequencies f_x and f_y and in terms of the spatial co-ordinates x and y .

Each exponential is weighted by a value given by $G(f_x, f_y)$ and, thus, equation 3.2 is an expansion (or expression) of the analogue optical function $g(x,y)$ in terms of this weighted generalised sum of exponentials.

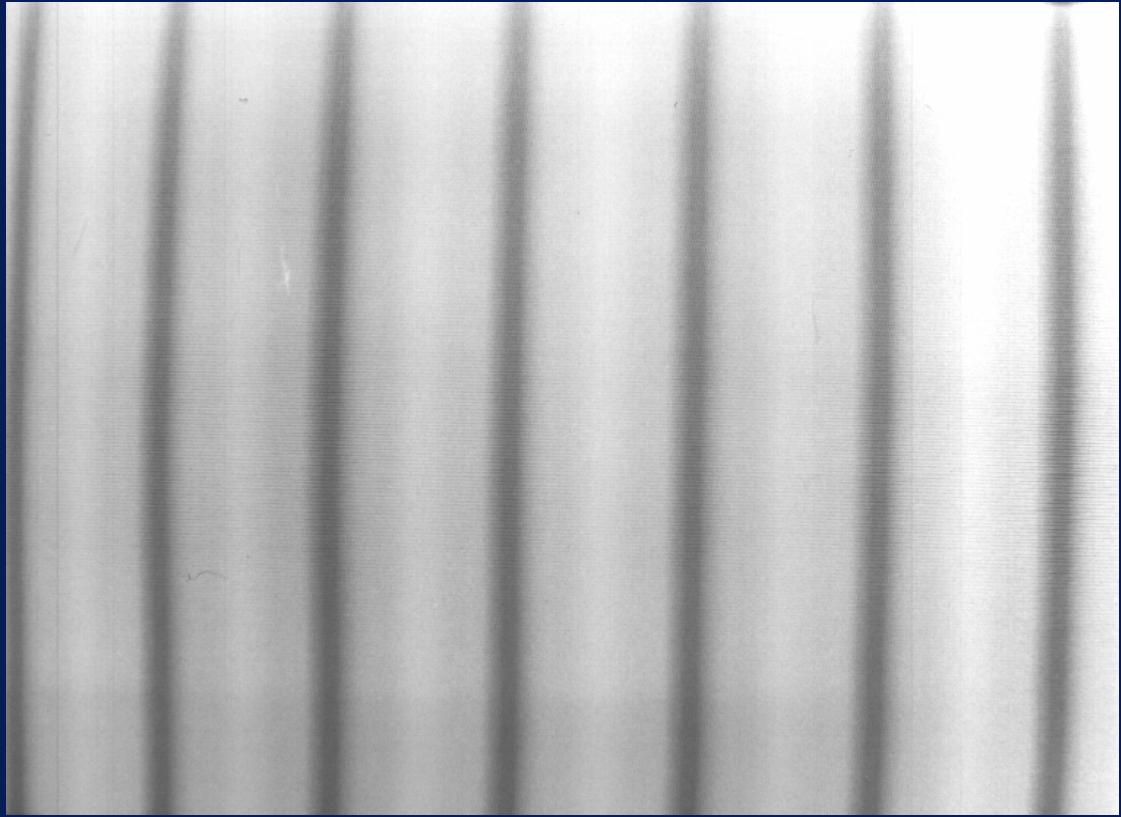
These weights are, in fact, given by equation 3.1, *i.e.* the Fourier Transform of the image function, and will, of course, vary with content.

Since these complex exponentials can also be expressed in terms of sine functions*, a spatial frequency domain which has, for example, just two non-zero values at, say,

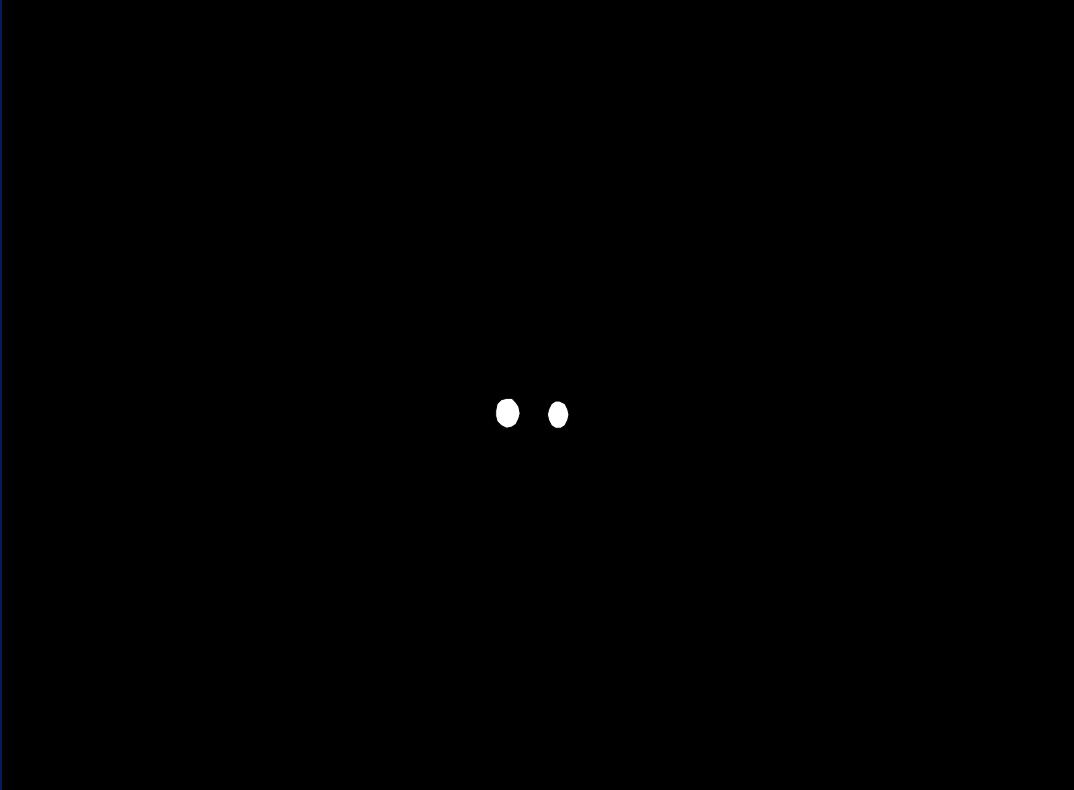
$$(f_{x1}, f_{y1}) \text{ and } (-f_{x1}, -f_{y1})$$

corresponds to a sinusoidal variation in intensity in the spatial domain, *i.e.*, to an image $g(x, y)$ which comprises sinusoidally undulating “stripes” of alternating light and dark intensity.

$$* e^{i\theta} = \cos \theta + i \sin \theta$$



(a) An image comprising a sinusoidal variation in intensity along the x-axis and ...



(b) its Fourier transform, comprising two spatial frequency components (f_{x1}, f_{y1}) and $(-f_{x1}, -f_{y1})$ both of which are spatial frequencies in the x direction.

The period and orientation of these stripes depends on the exact values of f_{x1} and f_{y1} .

Conversely, an image $g(x, y)$ which comprises a sinusoidal variation in intensity can be expressed in term of the spatial frequencies

$$\left(f_{x1}, f_{y1} \right) \quad \left(-f_{x1}, -f_{y1} \right)$$

The ‘quicker’ these sinusoidal variations, *i.e.* the greater the frequency of variation in the spatial domain, the further (f_{x1}, f_{y1}) and $(-f_{x1}, -f_{y1})$ are from the origin in the $G(f_x, f_y)$ domain.

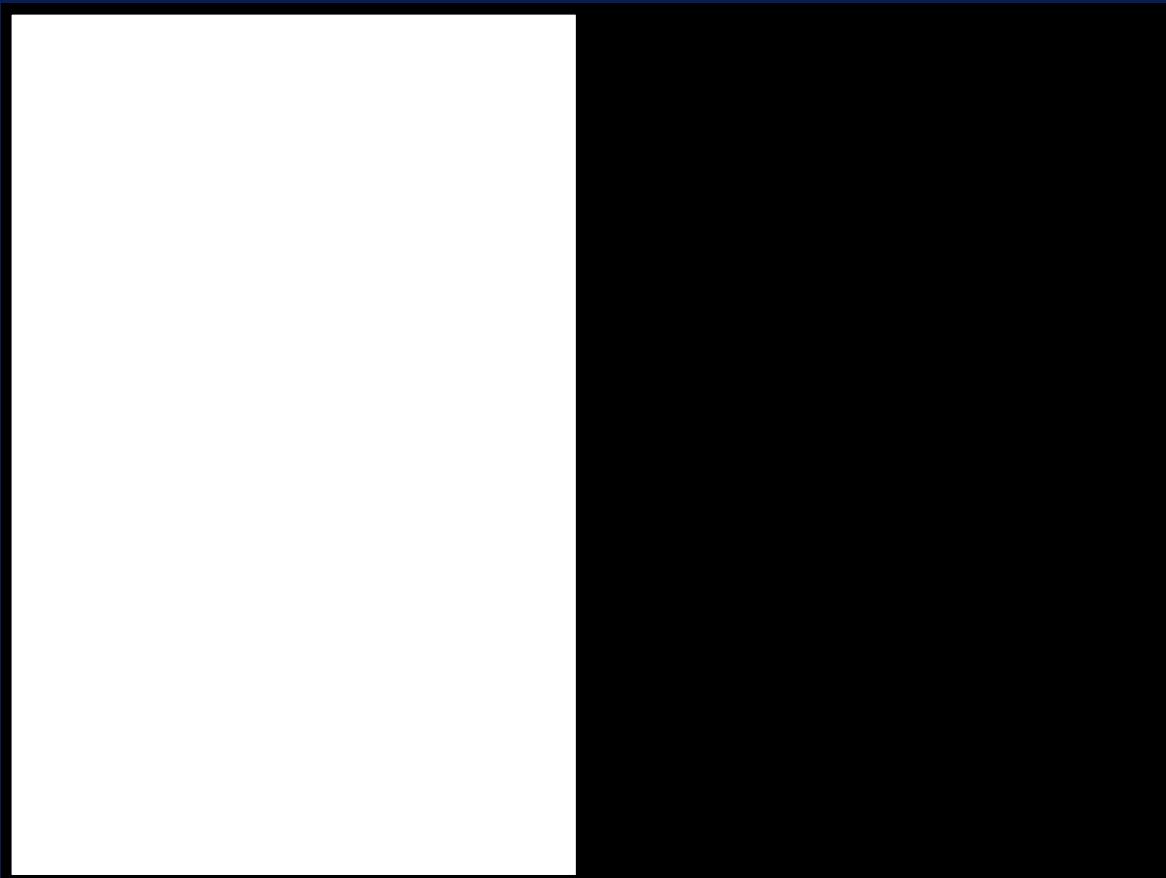
Of course, a sinusoidal variety in intensity is not a particularly common image.

However, Fourier analysis tells us that more complex functions can be constructed by including more terms of varying weight in the ‘generalised sum’ of exponentials, *i.e.* by including further spatial frequency components.

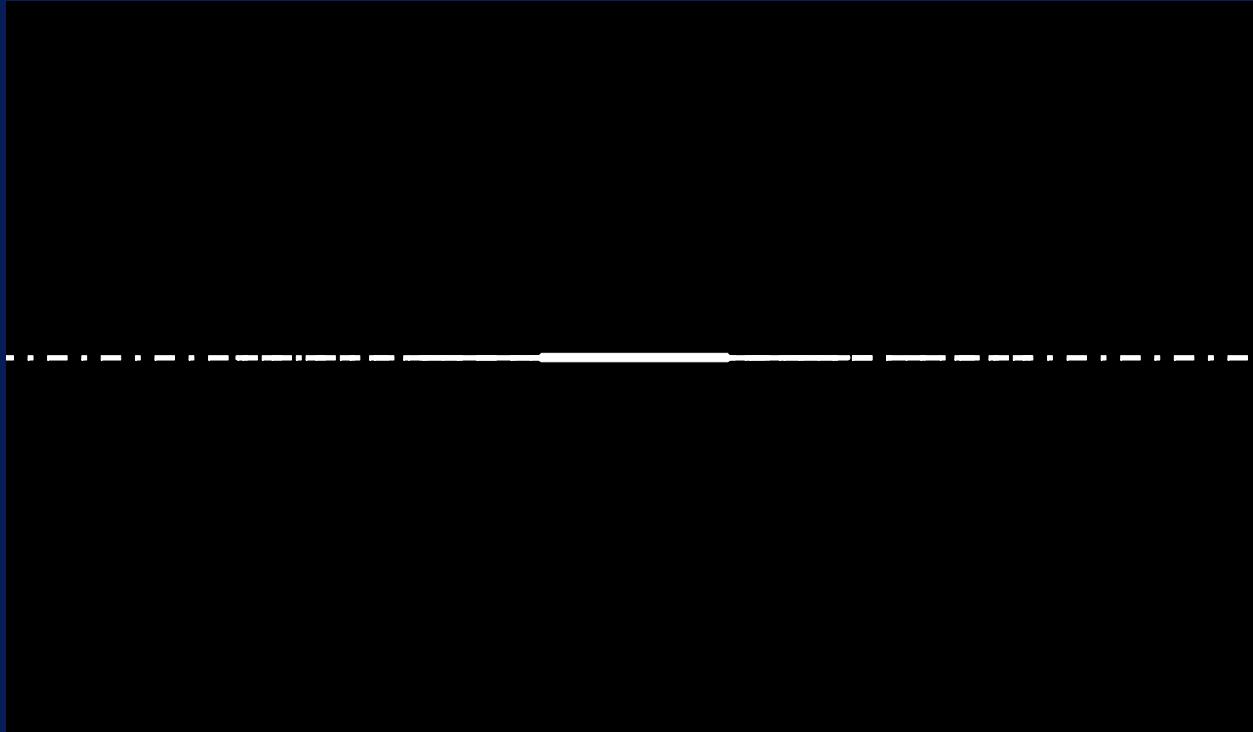
The exact weight is, again, determined by equation 3.1, *i.e.* the Fourier Transform.

An abrupt change in intensity will require the presence of a large number of terms which will correspond to high spatial frequencies, many of which are far removed from the origin of the $G(f_x, f_y)$ domain.

High spatial frequencies correspond to the presence of abrupt, or sharp, changes in the intensity of the image.



(a) An image comprising a step discontinuity in intensity along the x axis and



(b) its Fourier transform, exclusively comprising spatial frequency components $\$_x$, i.e. spatial frequencies in the x direction.

Sampling

*What sampling frequency is required, i.e.
how often does one need to sample in the
spatial domain in order to faithfully
represent an image?*

Shannon's sampling theorem tells us that a band-limited image (*i.e.* an image which does not comprise infinitely-high spatial frequencies) can be faithfully represented (*i.e.* reconstructed) *if the image is sampled at a frequency twice that of the highest spatial frequency present in the image.*

This sampling frequency is often referred to as the *Nyquist frequency*.

1. A sensor which has, for example, 756 photosites in the horizontal direction, *i.e.* along a line, will only be capable of representing a sinusoidal variation in intensity which has a spatial frequency of 378 cycles per unit distance.

A pattern of 378 alternately light and dark bars with abrupt edges, *i.e.* discontinuities in intensity, would obviously require a much higher sensor sampling frequency to faithfully represent the image.

2. The resolution of a television picture is also limited by the number of lines and the frequency bandwidth of the video signal.

The line frequency for the CCIR standard video signal is 15625 Hz. In addition, the nominal bandwidth of the CCIR standard is 5.0 MHz, meaning that a signal can transmit a video image with five million periodic variations in the signal (brightness) levels.

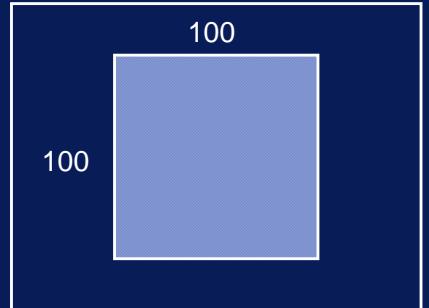
This results in an absolute maximum of $5 * 10^6 \div 15625 = 320$ periodic (or sinusoidal) variations per line, that is, the maximum spatial frequency which can be faithfully represented by a video signal is 320 cycles per line.

3. A frame-grabber which has a sampling frequency of 512 pixels in the horizontal direction, will only be capable of faithfully representing a sinusoidal variation in intensity which has a spatial frequency of 256 cycles per unit distance.

Again, a pattern of 256 alternatively light and dark bars with abrupt edges would require a much higher sampling frequency to faithfully represent the image.

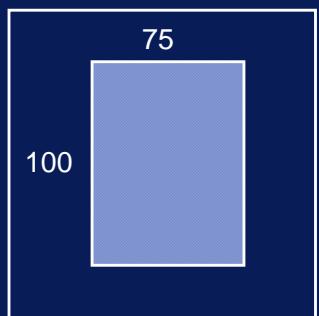
Inter-pixel Distances

- Video signals assume an aspect ratio of 4:3 (horizontal-to-vertical)
- although some framestores assume a square ratio of 1:1.



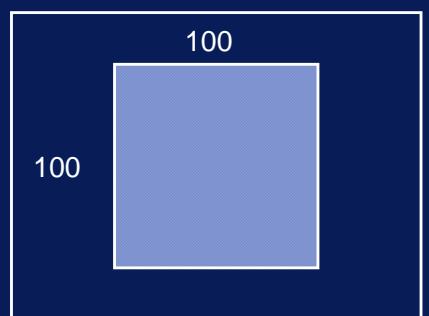
4:3 Aspect Ratio
(Video Camera Signals)

Image Acquisition



1:1 Aspect Ratio
(Framestores)

Image Display

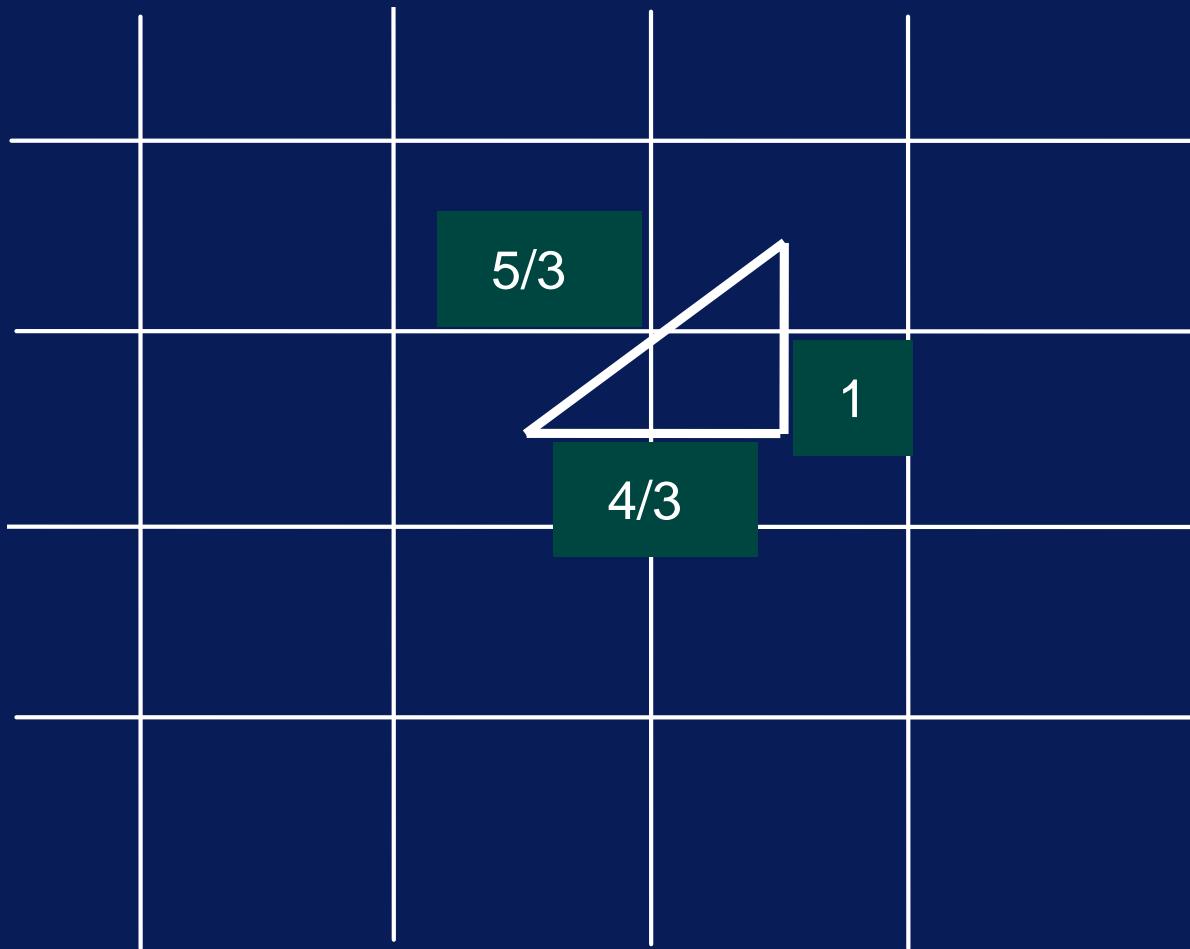


4:3 Aspect Ratio
(Video Monitor Signals)

The aspect ratio mis-match has serious repercussions for gauging or measuring functions of an inspection system

- the rectangular picture is being squeezed into a square image
- the effective distance between horizontal pixels is $4/3$ times greater than that between the vertical pixels.

The length of the hypotenuse completing the right-angle triangle formed by the horizontal $\frac{4}{3}$ inter-pixel interval of length and the vertical inter-pixel interval of length 1, is $\frac{5}{3}$.



Inter-pixel Distances

Most discussions of inter-pixel distances usually assume the diagonal inter-pixel interval to be $\sqrt{2}$

However, if we are working with a framestore which has a square aspect ratio and a video signal which has a 4:3 aspect ratio, then the diagonal inter-pixel interval is, in fact, $\sqrt{\frac{5}{3}}$.

Adjacency Conventions

This problem is one of defining exactly which are the neighbours of a given pixel.

Consider the 3*3 neighbourhood in an image where the pixels are labelled 0 through 8;

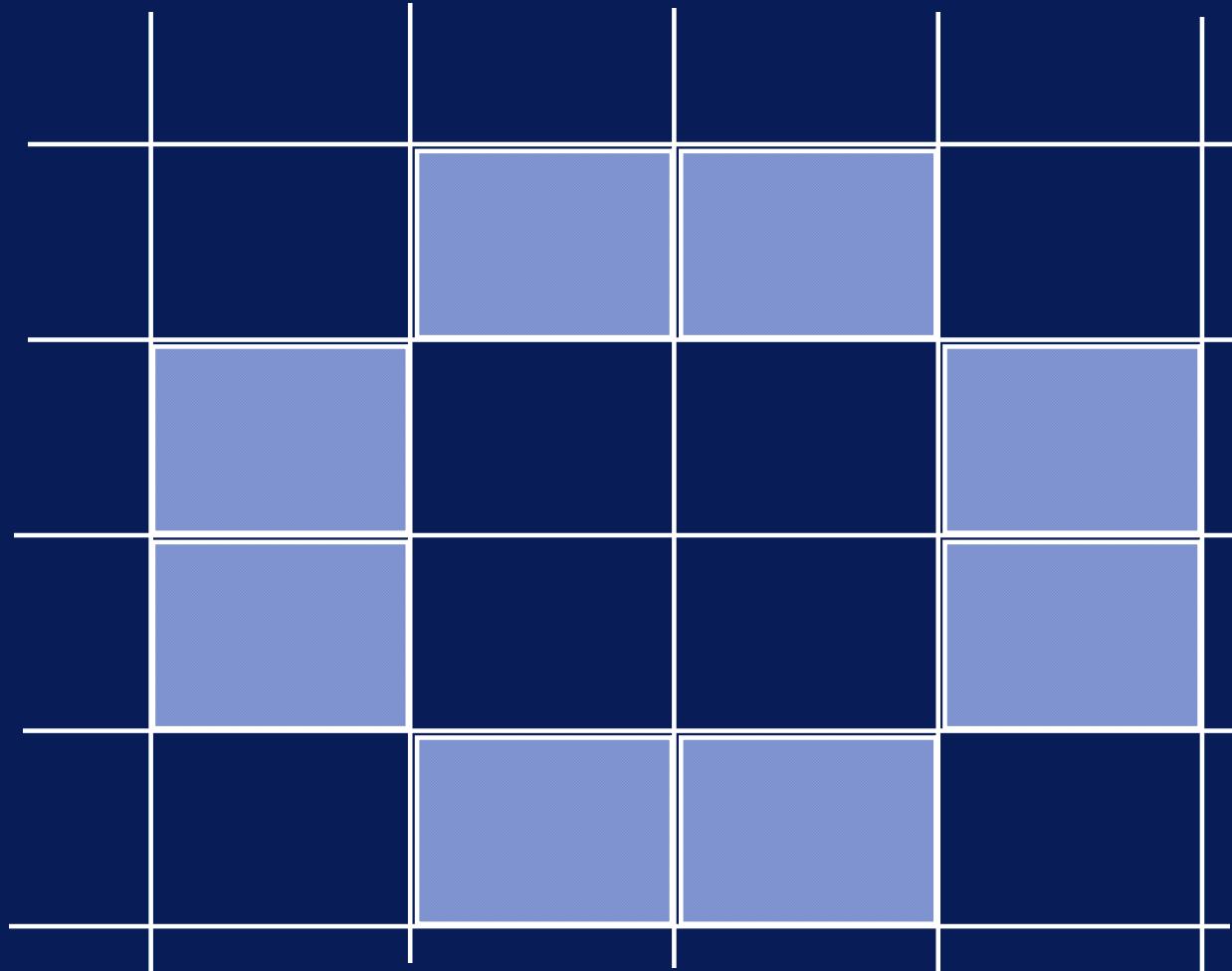
which pixels does pixel 8 touch?

One convention, called ***4-Adjacency***, stipulates that pixel 8 touches (*i.e.* is connected to) pixels 2, 4, 6 and 0, but does not touch (*i.e.* is not connected) to pixels 1, 3, 5 and 7.

An alternative convention, called ***8-Adjacency***, stipulates that pixel 8 is connected to all eight neighbours.



*A 3*3 Pixel Neighbourhood*



*Adjacency Conventions :
A Dark Doughnut on a White Table*

It is normal practice to use both conventions

- one for an object
- one for the background on which it rests.

In fact, this can be extended quite generally

- adjacency conventions are applied alternatively to image regions which are recursively nested (or embedded) within other regions as one goes from level to level in the nesting.

- Thus, one would apply 4-adjacency to the background region
- 8 Adjacency to the objects resting on the background
- 4-adjacency to holes or regions contained within the objects
- 8-Adjacency to regions contained within these, and so on.

This convention means that one never encounters topological anomalies such as the one described previously.

Because the 8-adjacency convention allows diagonally-connected neighbours, measurements of object features (*e.g.* its perimeter) will be more faithful with 8-adjacency.

Consequently, if it possible to stipulate which convention is to be applied to a particular object of interest, one should opt for 8-adjacency convention.

Fundamentals of Digital Image Processing

- Image processing
 - Image to image transformation
 - It starts with an image and produces a modified (enhanced) image
 - Iconic to iconic transformtion

Fundamentals of Digital Image Processing

- Image analysis

- Image to information transformation
 - It starts with an image and produces information representing a description or a decision
 - Iconic to symbolic transformation

Digital Image Analysis techniques are usually applied to previously processed (enhanced) images.

Since the analysis of enhanced images is accomplished quickly and easily by human observers, it is often erroneously assumed that the analysis phase is easy, if not trivial.

IT'S NOT!

The interpretation of such emergent features is simple only in the context of the powerful perceptual abilities of the human visual system.

For Machine Vision Systems,

*the sole purpose of Image Processing
is to produce images which make
subsequent analysis more simple and more
reliable*

*We are not concerned with their ‘aesthetic’
appearance*

The Image Processing Phase should :

- facilitate the extraction of information,
- compensate for non-uniform illumination,
- re-adjust the image to compensate for distortions introduced by the imaging system.

Historically, Digital Image Processing had two main thrusts to its development.

1. The natural extension of 1-dimensional
(temporal) Digital Signal Processing to
two (spatial) dimensions.

- Mathematical basis
- Rigorous Manipulation
- Classical Linear System Theory

2. The second, more heuristic, thrust considers digital images as a set of discrete sample points :
 - performing logical operations on the individual points.
 - this contrasts with the signal processing approach which treats images as a discrete representations of a continuous 2D function.

There are 3 distinct classes of operations

- Point Operations
- Neighbourhood Operations
- Geometric Operations

Point and Neighbourhood Operations are typically the type of operation that are performed by these frame-grabber sister boards.

Point Operations

Each pixel in the output image is a function of the grey-level of the pixel at the corresponding position in the input image and only of that pixel.

Point Operations

Point Operations are also referred to as grey-scale manipulation operations.

They cannot alter the spatial relationships of the image.

Point Operations

Typical uses of Point Operations include :

- Photometric Decalibration, to remove the effects of spatial variations in the sensitivity of a camera system.
- Contrast Stretching (*e.g.* if a feature or object occupies a relatively small section of the total grey-scale image, these local operations can manipulate the image so that it occupies the entire range).

Point Operations

- Thresholding, in which all the pixels having grey-levels in specified ranges in the input image are assigned a single specific grey-level in the output image.

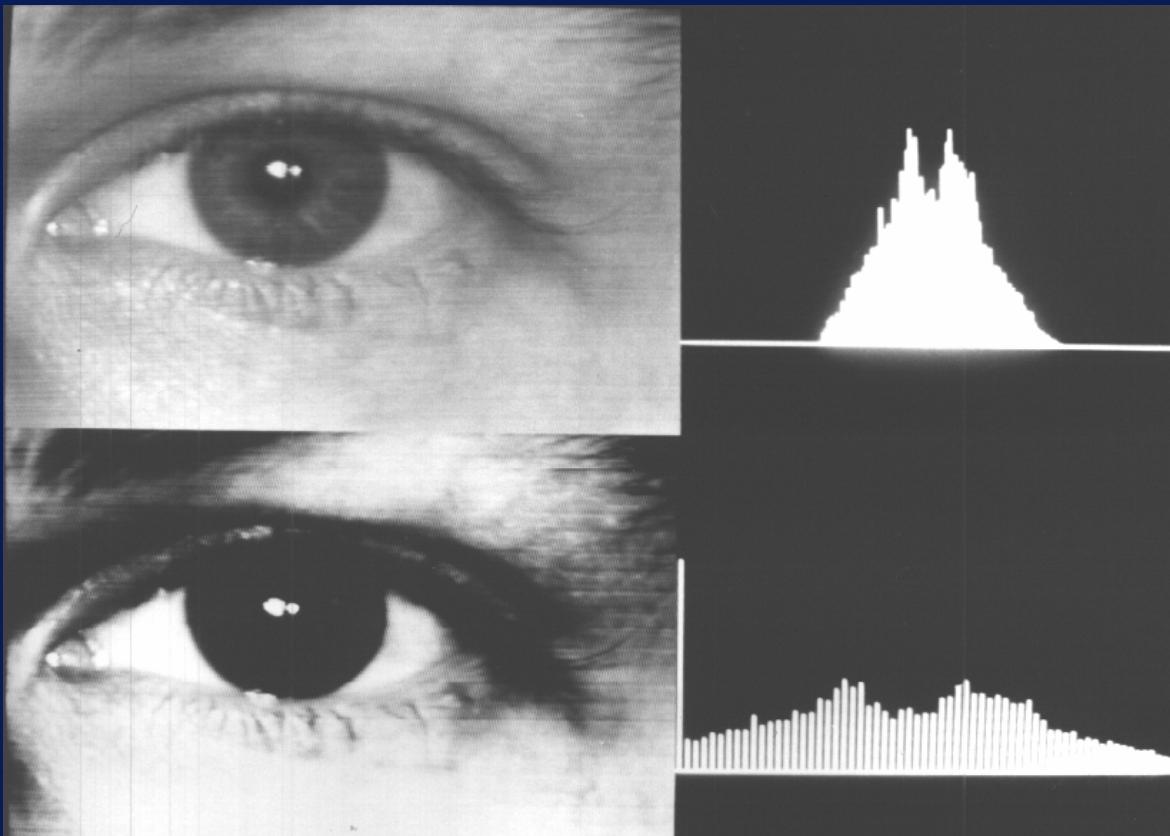
Point Operations

As we shall see, these operations are most effectively accomplished using the hardware input look-up tables (LUTs) which are provided by most framegrabbers.

Contrast Stretching

101	100	103	105	107	105	103	110
110	140	120	122	130	130	121	120
134	134	135	131	137	138	120	121
132	132	132	133	133	150	160	155
134	140	140	135	140	156	160	174
130	138	139	150	169	175	170	165
126	133	138	149	163	169	180	185
130	140	150	169	178	185	190	200

Contrast Stretching



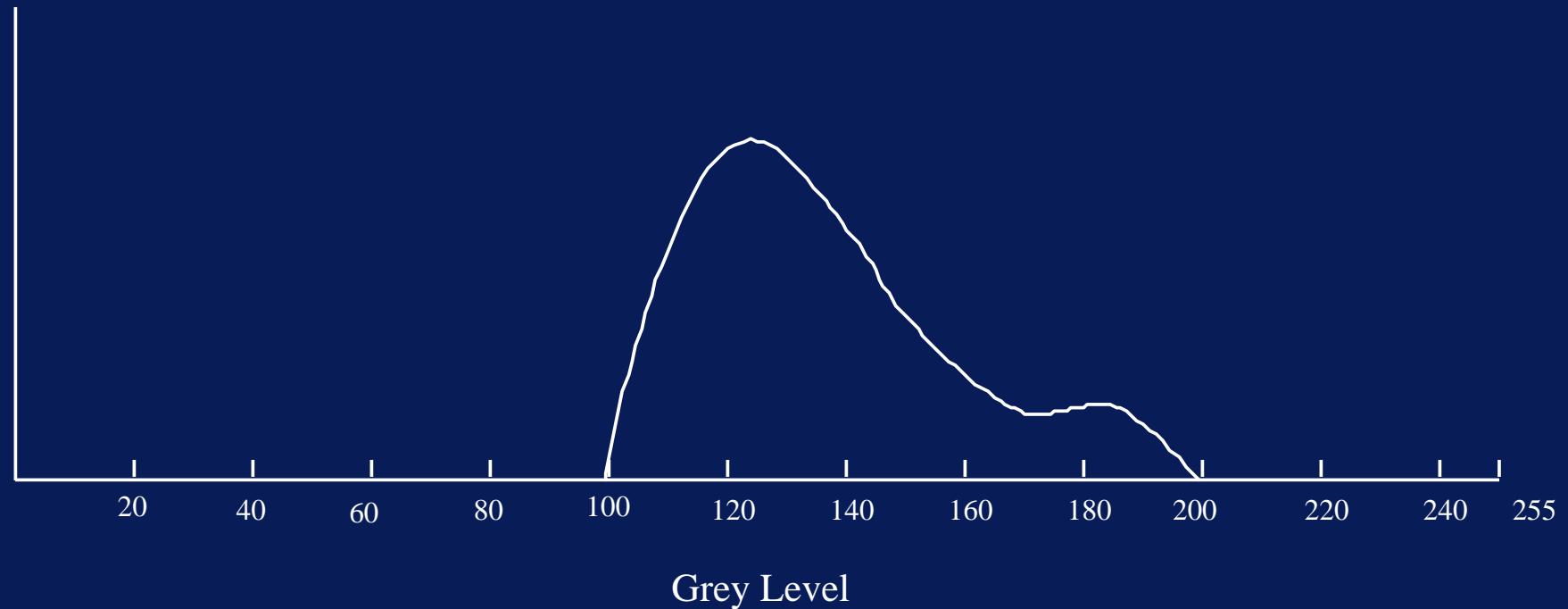
Contrast Stretching

Contrast Stretching is achieved by first shifting all the values so that the actual pixel grey-level range begins at 0 :

i.e. Add to every pixel the difference between the final low value (0) and the initial low value (100) : $0 - 100 = -100$.

Contrast Stretching

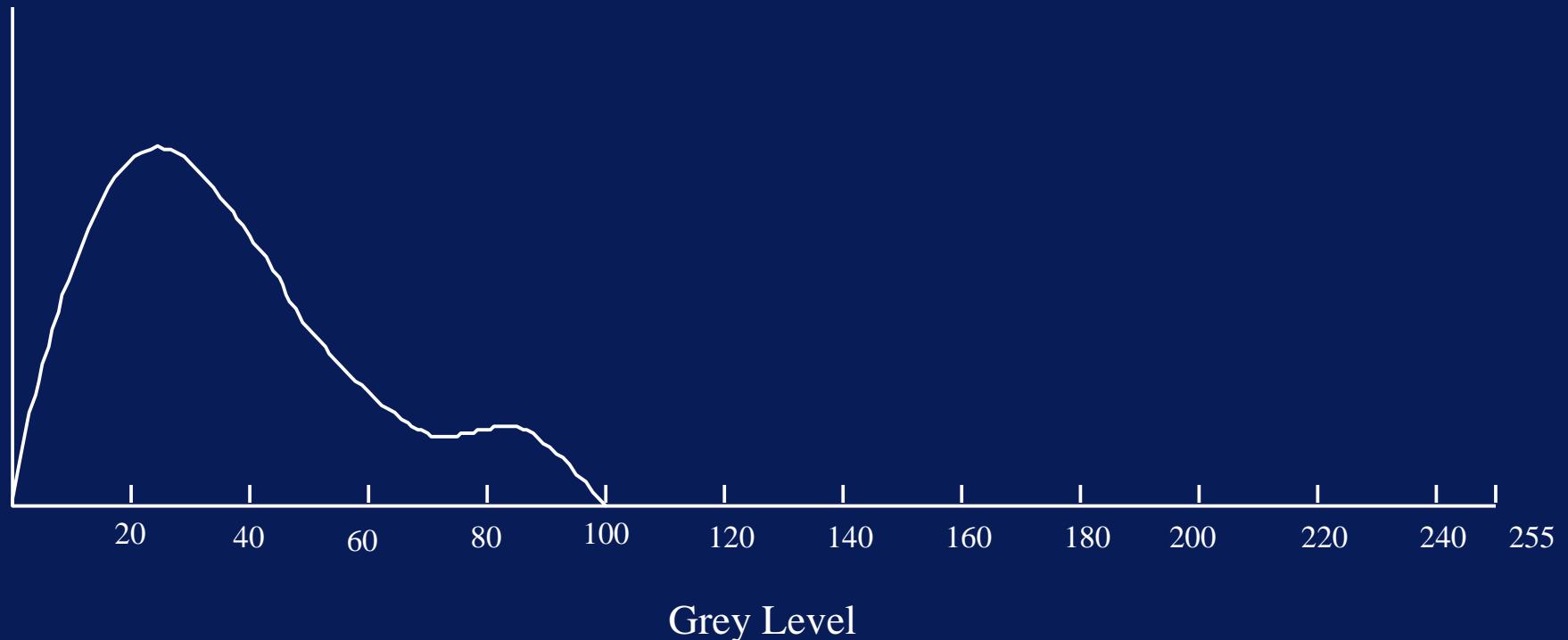
Number of Pixels



Grey-Level Histogram

Contrast Stretching

Number of Pixels



Shifted Grey-Level Histogram

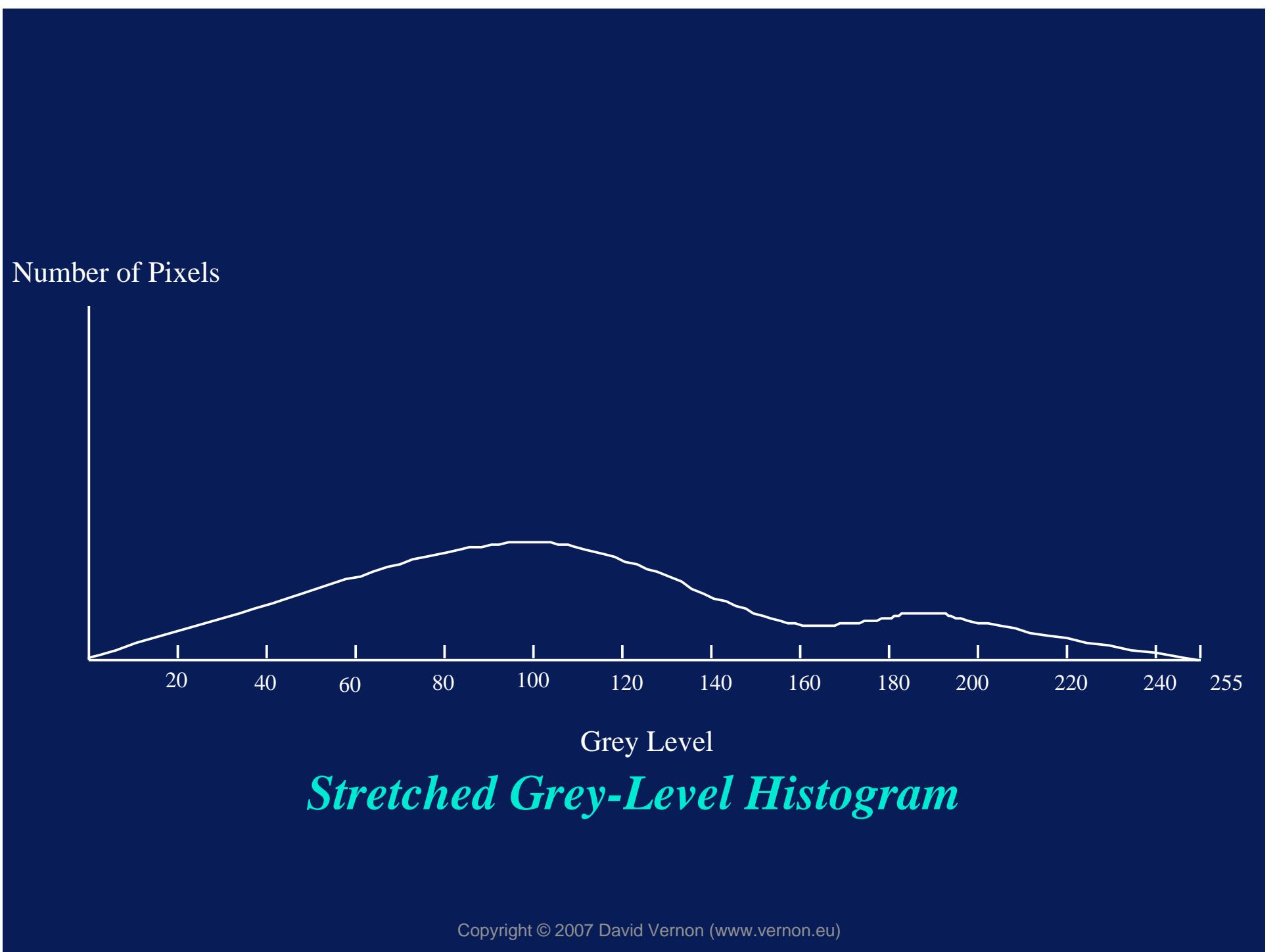
Contrast Stretching

- Next, we scale everything, reassigning values in the range 0 - 100 to the range 0 - 255.
i.e. We scale by a factor $= (255-0)/(100-0)$
 $= 2.55$

Contrast Stretching

Thus, in this instance, to stretch the contrast so that all grey levels in the grey-scale are utilised, one must simply apply the following operation

*new pixel value :=
(old pixel value - 100) * 2.55*



An Example of Contrast Stretching

By way of example, let us consider the application of this stretching to any pixel having either the lowest grey level (100) in the original image or the highest grey level (200) in the original image.

If old pixel value = 100

*new pixel value := (100-100)*2.55*
= 0

If old pixel value = 200

*new pixel value := (200-100)*2.55*
= 255

Algorithm expressed in pseudo-code

```
/*Contrast Stretching in a 512*512 pixel  
image
```

```
HIGH and LOW are assumed to be the  
highest and lowest grey-levels,  
respectively, in the unstretched image  
*/
```

```
scale_factor := 255 / (HIGH-LOW)
FOR i := 1 TO 512 DO
    FOR j = 1 TO 512 DO
        IF image[i, j] < LOW THEN
            image[i, j] := 0
        ELSE
            IF image[i, j] := 255 THEN
                image[i, j] := 255
            ELSE
                image[i, j] :=
                    (image[i, j] - LOW) * scale_factor
```

```
/* Look Up Table Version */

scale_factor := 255/(HIGH-LOW)

/* Initialise LUT */
FOR I := 0 TO LOW-1 DO
    LUT[I] := 0

FOR I := LOW TO HIGH DO
    LUT[I] := (I - LOW) * scale_factor

FOR I := HIGH+1 TO 255 DO
    LUT[I] := 255

FOR i := 1 TO 512 DO
    FOR j := 1 TO 512 DO
        image[i, j] := LUT[image[i, j]]
```

There are many other approaches which can be used for Contrast Enhancement

e.g. *Histogram Equalisation* is a technique which

- computes the histogram of the grey-level distribution in the image,
- re-assigns grey-levels to pixels in an effort to generate a histogram where there are equally many pixels at every grey-level.

Thresholding

Some scenes can be very simple, from a visual point of view, in that they comprise just one or two types of objects.

Thresholding

Suppose we wish to process an image of such a scene so that it exhibits extremely high contrast and so that *e.g.* the solder tracks and circuit board are very easily distinguishable.

Thresholding

The problem is to convert a grey-scale image (typically with 255 grey-levels) into an image with just two levels of grey;

- all the image pixels representing the circuit board *probably* have grey-levels in the range 0 to 160
- while the solder tracks *probably* have grey-levels in the range 160 to 255

Thresholding

- assign all the pixels in the range 0 to 160 to be black
- assign all pixels in the range 160 to 255 to be white.

Thresholding

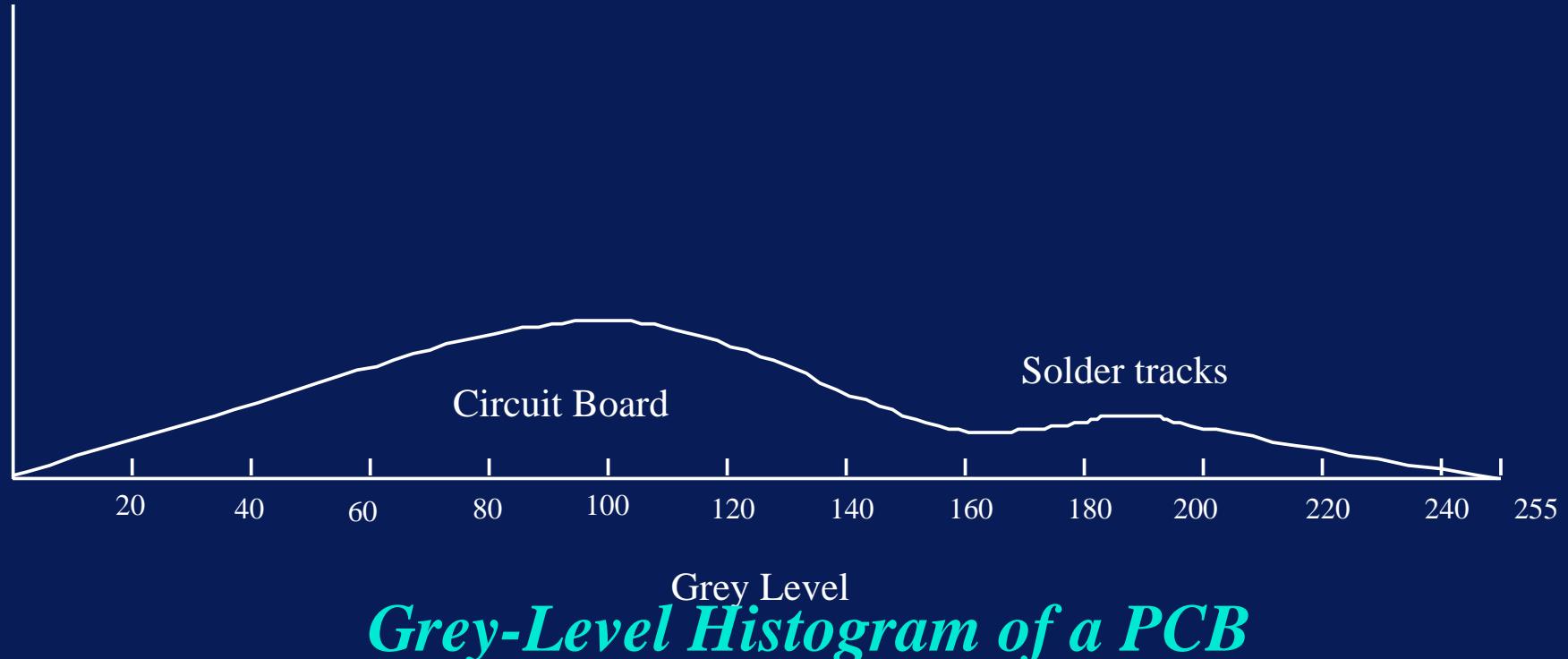
- We now have an explicit labelling in our image
 - the circuit board pixels are labelled (identified) with a grey level of 0
 - the solder is labelled with a grey level of 255.

Thresholding

We have nominated the grey-level 160 as the *threshold* and the re-assignment of pixel values is known as *Thresholding*.

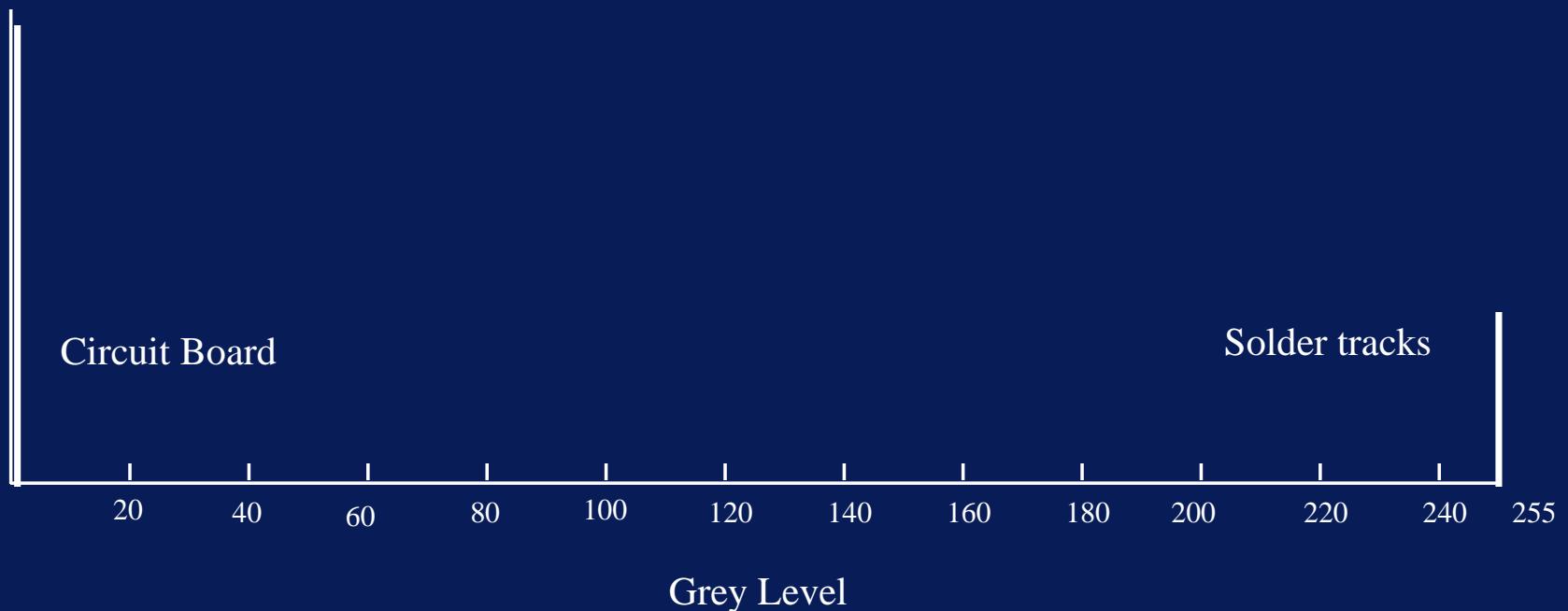
Thresholding

Number of Pixels



Thresholding

Number of Pixels



*Grey-Level Histogram of a PCB after
Thresholding*

Pseudo-Code for Thresholding

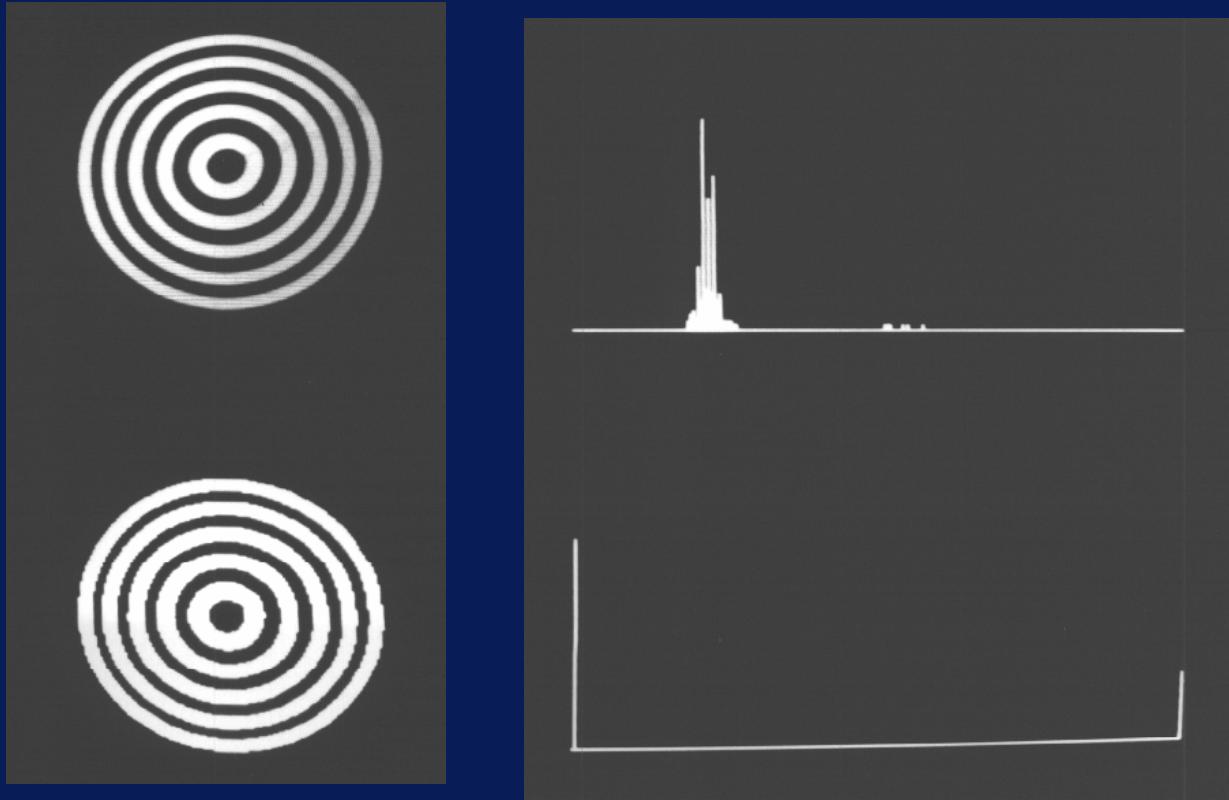
```
/* Threshold an image in place  
(i.e.. without an explicit destination  
image) */  
  
FOR i := 1 TO 512 DO  
    FOR j := 1 to 512 DO  
        IF image[i,j] > threshold  
            THEN image[i,j] := 255  
        ELSE image[i,j] := 0
```

```
/* Threshold an image in place (i.e.. without an
explicit destination_image) using a LUT */

/* initialise LUT */

FOR i := 0 TO threshold DO
    LUT[i] := 0
FOR i := threshold + 1 TO 255 DO
    LUT[i] := 255

FOR i := 1 TO 512 DO
    FOR j := 1 to 512 DO
        image[i,j] := LUT[ image[i,j] ]
```



Thresholding

A grey-scale image (top-left) with its histogram (top-right) is thresholded at a grey-scale value of 128 (bottom-left); the resultant histogram is shown in the bottom left quadrant.

Noise Suppression by Image Addition

- If it is possible to obtain multiple images of a scene, each taken at a different time.
- If the scene contains no moving objects,
- then the noise in the image can be reduced by averaging these images.

Noise Suppression by Image Addition

- In the averaging process the constant part of the image (that which is due to light reflected by stationary objects) is unchanged.
- the noise, which will in general change from image to image, will accumulate more slowly.

Noise Suppression by Image Addition

The assumptions inherent in this approach are

- a) The noise in each image is indeed uncorrelated, and
- b) the noise has a zero mean value; *i.e.* it averages out to a grey level of zero which contributes nothing to the real image.

Noise Suppression by Image Addition

With these assumptions, it is possible to show that averaging N images increases the signal to noise ratio by \sqrt{N} .

Many commercial framestores incorporate facilities to accomplish this averaging in real-time (*i.e.* as the image is acquired).

Noise Suppression by Image Addition

However, bear in mind the assumptions upon which this technique is based; not all noise has these desirable properties.

Background Subtraction

Digital Image Subtraction

- pixel values of two images are subtracted on a point by point basis.
- subtraction of a *known* pattern (or image) of super-imposed noise
- Motion Detection: stationary objects cancel each other out while moving objects are highlighted.

Photometric Decalibration

Photometric Decalibration is one of the most important applications of Background Subtraction.

- In some circumstances, camera systems exhibit non-uniform sensitivity to light.
- Quite often, this is caused by the lens and is manifested as an image centre which is somewhat brighter than the periphery.

Photometric Decalibration

- Problems if one is using Thresholding techniques to isolate objects,
- since the grey-level which is assumed to correspond to the object may change depending on the position in the image.

Photometric Decalibration

One solution (apart from replacing the lens) is to model this non-uniform response by,

- taking an image of a uniformly-shaded surface
- identifying the minimum grey-level of this image

Photometric Decalibration

- subtract this value from each pixel to generate an image which represents the effective response of the sensor.

Images which are subsequently acquired can then be processed by subtracting this calibration image from them.

Neighbourhood Operations

Generates an ‘output’ pixel

- on the basis of the pixel at the corresponding position in the input image
- *and on the basis of its neighbouring pixels.*

Neighbourhood Operations

The size of the neighbourhood may vary :

- $3 * 3$
- $5 * 5$
- $63 * 63$ pixels

Neighbourhood Operations

Often referred to as *Filtering Operations*.

- *Convolution* of an image with a filter kernel or mask.
- Such filtering often addresses the removal (or suppression) of noise or the enhancement of edges;
- most effectively accomplished using convolver (or filtering) hardware.

Neighbourhood Operations

Other Neighbourhood Operations

- Applying some logical test based on,
e.g. the presence or absence of object pixels in a local neighbourhood surrounding the pixel in question.
- Object Thinning (or Skeletonising)
- Erosion and Dilation (contract an object in an orderly manner)

Convolution

The 2D Convolution Integral :

$$g(x,y) = f * h = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(i-m, j-n)h(m,n)dmdn$$

- output g of a shift-invariant linear system
- is given by the convolution or application of the input signal f with a function h which is characteristic of the system.

Convolution

The function h is normally referred to as the filter, since it dictates what elements of the input image are allowed to pass through to the output image.

By choosing an appropriate filter, we can enhance certain aspects of the output and attenuate others.

A particular filter h is often referred to as a filter kernel.

Convolution

Thus, the form of g depends on

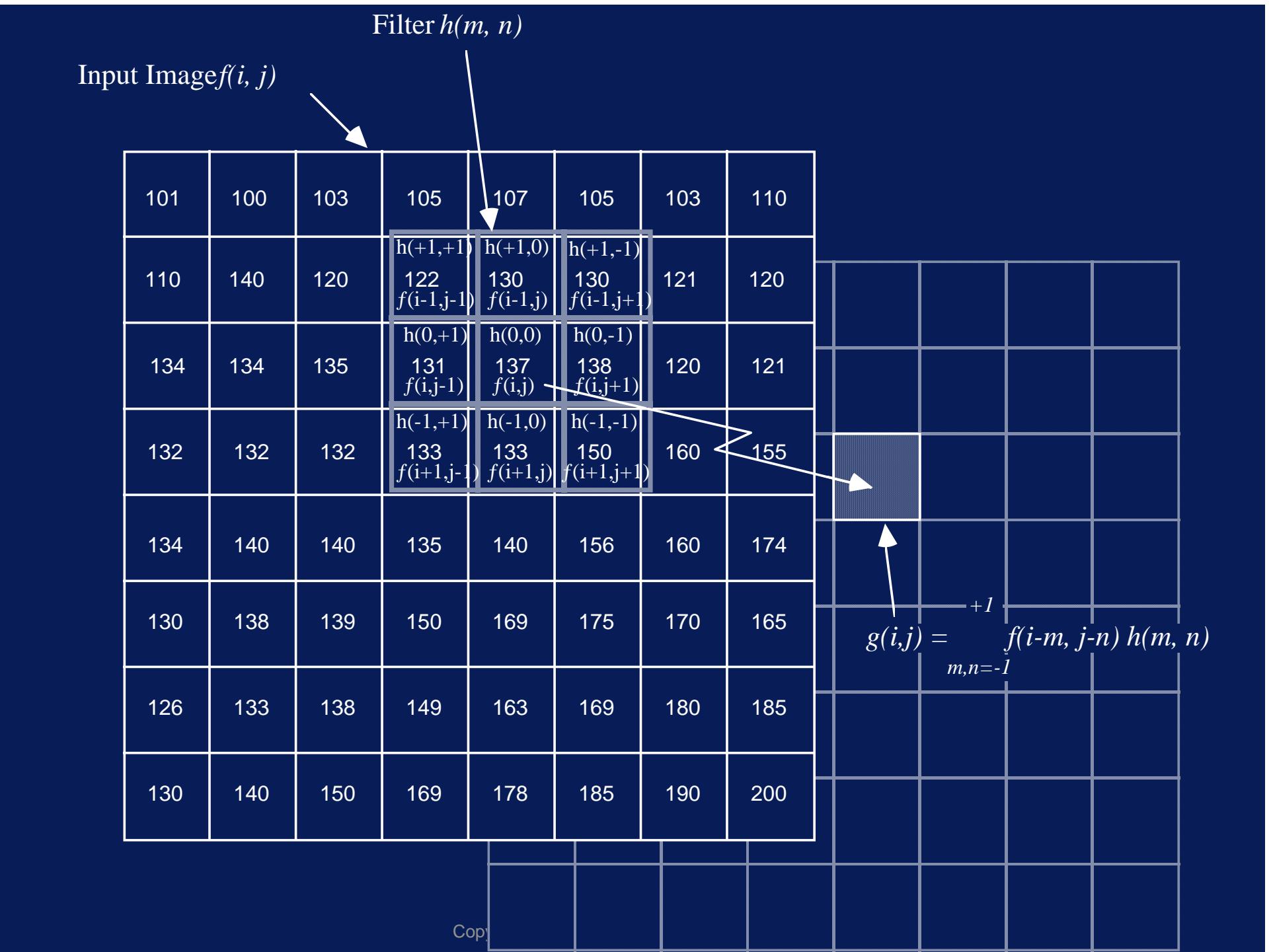
- the input f (obviously) and
- on the form of the system h through which it is being passed

The relationship is given by the Convolution Integral.

Convolution

$h(-1, -1)$	$h(-1, 0)$	$h(-1, +1)$
$h(0, -1)$	$h(0, 0)$	$h(0, +1)$
$h(+1, -1)$	$h(+1, 0)$	$h(+1, +1)$

*$3 * 3$ Convolution Filter h*



Convolution

In the discrete domain of digital images the Convolution Operation is given by:

$$g(i, j) = f * h = \sum_m \sum_n f(i - m, j - n)h(m, n)$$

The summation is taken only over the area where $(i-m, j-n)$ is defined, *i.e.* over the area where f and h overlap.

Convolution

Here

- a Filter Kernel is a $3 * 3$ Pixel Image
- with the origin $h(0, 0)$ at the centre
- representing a mask of nine distinct weights.

$$[h(-1, -1) \dots \dots h(+1, +1)]$$

	-1	0	1
-1			
0			
1			

Convolution

The Kernel or Mask is superimposed on the Input Image

- the input image pixel values are multiplied by the corresponding weight,
- the nine results are summed,
- and the final value of the summation is the value of the Output Image Pixel at a position corresponding to the position of the centre element of the kernel.

Convolution

Note that the mask is first rotated by 180° since :

- $f(i-1, j-1)$ must be multiplied by $h(1, 1)$,
- $f(i-1, j)$ must be multiplied by $h(1, 0)$,
-,
- and $f(i+1, j+1)$ must be multiplied by $h(-1, -1)$

Quite often the rotation by 180° is omitted if the mask is symmetric.

Convolution

```
FOR i := (low_limit_of_i + 1) TO (high_limit_of_i - 1) DO
  FOR j := (low_limit_of_j + 1) TO (high_limit_of_j - 1)
    DO
      /* for each feasible point in the image
         ... form the convolution */
      temp := 0
      FOR m := -1 TO +1 DO
        FOR n := -1 TO +1 DO
          temp := temp + F[i-m, j-n] * H[m, n]
      G[i,j] := temp
```

Filtering

All Image Systems introduce some amount of distortion into an image,

- image blur due to camera shake.

Since optical and electrical systems are (to an approximation) linear, the imaging system may also be assumed to be linear.

This implies that it has some transfer function $H(\omega_x, \omega_y)$.

Convolution

This is the Frequency Domain equivalent of the system impulse response $h(x, y)$ in the Spatial Domain;

- $h(x, y)$ typifies how the system would respond if a single unit spike were input to it.

Convolution

Alternatively :

- the image may be transformed to the frequency domain using the Digital Fourier Transform,
- multiplied by the inverse of the transfer function,
- transformed back to the spatial domain by use of the Inverse Fourier Transform.

Convolution

Thus, the transfer function describes how some input image is transformed into some output image.

Convolution

Since the output of this system is a distortion of the correct image we would expect that :

- if we could model the system, (define its transfer function),
- compute the inverse
- apply it to the distorted image,

Convolution

we would arrive at a much better approximation to the original.

This is accomplished by convolving the inverse impulse response with the image: this operation is referred to as *Deconvolution*.

Noise Suppression

Noise : any unwanted contamination of an image.

The mechanism by which noise is removed depends on the assumption we make regarding the form of this unwanted contamination.

Noise Suppression

One of the most common (and realistic) assumptions made about noise is that it has a high spatial frequency.

- apply a low-pass spatial filter which will
- attenuate the higher spatial frequencies
- allow the low spatial frequencies component to pass through to the resultant destination image.

Noise Suppression

Of course, if the image itself exhibits high spatial frequencies then it will be somewhat degraded after filtering.

Noise Suppression

These low-pass filters can be implemented by convolving the image with some simple mask

- For example, each of the mask values might be equally weighted,
- in which case, the operation we are performing is simply the evaluation of the local mean of the image in the vicinity of the mask.

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

Local Average Mask

101 * 1/9	100 * 1/9	103 * 1/9	105	107	105	103	110
110 * 1/9	140 * 1/9	120 * 1/9	122	130	130	121	120
134 * 1/9	134 * 1/9	135 * 1/9	131	137	138	120	121
132	132	132	133	133	150	160	155
134	140	140	135	140	156	160	174
130	138	139	150	169	175	170	165
126	133	138	149	163	169	180	185
130	140	150	169	178	185	190	200

Image smoothing using local average mask.

$$\begin{aligned} & 101 * 1/9 + 100 * 1/9 + 103 * 1/9 + \\ & 110 * 1/9 + 140 * 1/9 + 120 * 1/9 + \\ & 134 * 1/9 + 134 * 1/9 + 135 * 1/9 \end{aligned}$$

which is equivalent to:

$$\begin{aligned} & 1/9 * [101 + 100 + 103 + \\ & 110 + 140 + 120 + \\ & 134 + 134 + 135] = 121. \end{aligned}$$

Noise Suppression

Thus, the central point becomes 121 instead of 140 and the image will appear much smoother.

Noise Suppression

Occasionally, it may be more useful to apply this smoothing subject to some condition, e.g. the centre pixel is only assigned if the difference between the average value and the original pixel value is greater than a previously set threshold.

Noise Suppression

This goes some way toward removing noise without smoothing out too much of the detail in the original image.

This averaging or smoothing is applied at all points of the image.

Noise Suppression

The algorithm can be expressed in pseudo-code as follows :

```
/* For a 512x512 image, indexed from 0 to 511 */  
  
FOR i := 1 TO 510 DO  
    FOR j := 1 TO 510 DO  
  
        /* for each feasible point in the image: compute average */
```

Noise Suppression

```
average := (source_image[i-1, j-1] +
            source_image[i-1, j]      +
            source_image[i-1, j+1]      +
            source_image[i, j-1]       +
            source_image[i, j]         +
            source_image[i, j+1]       +
            source_image[i+1, j-1]      +
            source_image[i+1, j]       +
            source_image[i+1, j+1]) / 9

/* destination image assumes the average value */
destination_image[i,j] := average
```

Noise Suppression

Other noise suppression techniques.

For example, a **median filtering**

- a pixel is assigned the value of the median of pixel values in some local neighbourhood;
- the size of the neighbourhood is arbitrary, but neighbourhoods in excess of 3*3 or 5*5 may be impractical from a computational point of view.

Noise Suppression

- the evaluation of the median requires that the image pixel values be first sorted.

In general, the median filter is superior to the mean filter in that image blurring is minimised.

Noise Suppression

Unfortunately, it is computationally complex and is not easily effected in hardware thus tends not to be used much in machine vision.

Gaussian Filtering

- The image is convolved with a Gaussian function.
- Perhaps one of the most commonly used smoothing techniques in advanced computer vision since it possesses several useful properties.

Gaussian Filtering

- Unfortunately, it is not yet widely used in industrial machine vision because the size of the masks are very large and the processing is consequently intensive.

Gaussian Filtering

It should be noted, however, that some vendors do offer dedicated image processing boards for Gaussian Filtering.

If such hardware is available, this type of smoothing should be considered.

Gaussian Filtering

The Gaussian function $G(x, y)$ is defined by

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

where σ defines the effective spread of the function :

Gaussian Filtering

- Gaussian functions with a small value for σ are narrow,
- those with with a large value for σ are broad.

Gaussian Filtering

Note that, since the Gaussian function is defined over an infinite support, i.e. it has non-zero (but very small) values at $x, y = \pm\infty$, we must truncate the function.

Gaussian Filtering

- One chooses the quantisation resolution of the function by deciding on the integer number which will represent the maximum amplitude at the centre point (*e.g.* 1000),
- one chooses the mask size which included all non-zero values.

Gaussian Filtering

The Gaussian function for three values of s (1.5, 3.0, and 6.0) together with their corresponding discrete 1-D masks; note that the result of convolution with these masks should be normalised by dividing by the sum of the mask weights.

1	1	1
2	3	2
3	11	3
6	18	6
11	28	11
18	43	18
28	65	28
43	95	43
65	135	65
95	186	95
135	249	135
186	324	186
249	411	249
324	506	324
411	606	411
506	706	506
606	800	606
706	882	706
800	945	882
882	986	945
945	1000	986
1000	945	1000
945	800	945
800	945	800
945	800	945
800	606	882
606	411	800
411	249	706
249	135	606
135	65	506
65	28	411
28	11	324
11	3	249
3	1	186
1	135	135
186	95	95
249	65	65
324	43	43
411	28	28
506	18	18
606	11	11
706	6	6
800	3	3
882	2	2
945	1	1

Gaussian Filtering

Typically, a mask of size of 23*23 pixels would be required to represent a 2D Gaussian function with a value of 3.0 pixels for σ .

Gaussian Filtering

Fortunately, there is no need to use 2-D Gaussian functions since the convolution of a 2-D Gaussian can be effected by two convolutions with 1-D Gaussian functions.

Gaussian Filtering

Specifically :

$$G(x,y) * I(x,y) = G(x) * \{G(y) * I(x,y)\}$$

- the image is first convolved with a ‘vertical’ Gaussian
- the resulting image is convolved with a ‘horizontal’ Gaussian.

Gaussian Filtering

Why is the Gaussian such a popular smoothing function ?

- While the primary purpose of a smoothing function is to reduce noise,
- it is often desirable to be able to choose the resolution at which intensity changes are manifested in the image, *i.e.* to choose the level of detail which is retained in the image.

Gaussian Filtering

For example,

- an image which has been smoothed just a little (small σ) will retain a significant amount of detail,
- while one which has been smoothed a great deal (large σ) will retain only the gross structure.

Gaussian Filtering

We wish to sharply delimit the spatial frequencies (see section 3.1.1) which are present in the image, *i.e.* to localise the spatial frequencies bandwidth of the image.

Gaussian Filtering

We also need to ensure that the smoothing function does not excessively distort the image by smearing the features.

To allow for this, we need to ensure that the function has a limited support in space.

Gaussian Filtering

The Gaussian function optimises the trade-off between these two conflicting requirements.

Thinning, Erosion and Dilation

- Thinning is an iterative neighbourhood operation which generates a skeletal representation of an object.
- It assumes, of course, that you know exactly what constitutes the object in the image and what constitutes the background.

Thinning, Erosion and Dilation

- The skeleton of an object may be thought of as a generalised axis of symmetry of the object.
- It is a suitable representation for objects which display obvious axial symmetry.
- The Medial Axis Transform (MAT) proposed by Blum is one of the studied techniques for generating the skeleton.

Thinning, Erosion and Dilation

- More recently, however, Brady has introduced the related concept of a Symmetric Axis Transform (SAT).

Thinning, Erosion and Dilation

The skeleton exhibits three topological properties :

- Connectedness (one object generates one skeleton)
- Invariance to scaling and rotation
- Information preservation in the sense that the object can be reconstructed from the medial axis.

Thinning, Erosion and Dilation

The concept of thinning a binary image of an object is related to such medial axis transformations in that

- it generates a representation of an *approximate* axis of symmetry of a shape
- by successive deletion of pixels from the boundary of the object.

Thinning, Erosion and Dilation

In general, this thinned representation is not formally related to the original object shape and it is not possible to reconstruct the original boundary from the object.

Thinning

- a logical neighbourhood operation
- where object pixels are removed from an image.

Thinning

Set of conditions for pixel removal :

The first restriction is that the pixel must lie on the border of the object.

This implies that it has at least one 4-connected neighbouring pixel which is a background pixel.

Thinning

The removal of pixels from all borders simultaneously would cause difficulties :

For example, an object two pixels thick will vanish if all border pixels are removed simultaneously.

Thinning

A solution to this is to remove pixels of one border orientation only on each pass of the image by the thinning operator.

Thinning

Opposite border orientations are used alternatively to ensure that the resultant skeleton is as close to the medial axis as possible.

Thinning

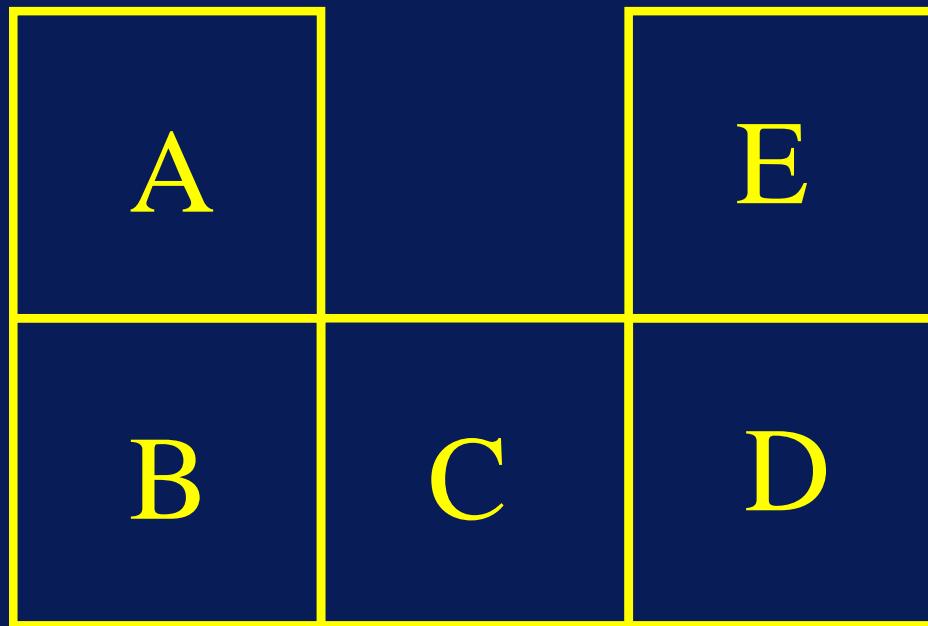
The second restriction is that the deletion of a pixel should not destroy the object's connectedness, i.e. the number of skeletons after thinning should be same as the number of objects in the image before thinning.

Thinning

This problem depends on the manner in which each pixel in the object is connected to every other pixel.

Consider now the five pixel object shown next

Thinning



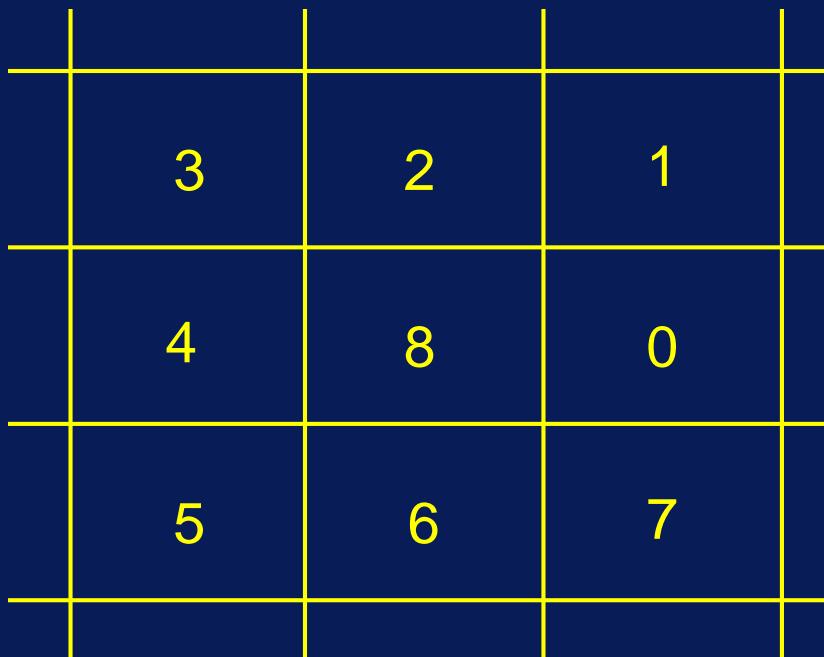
A Critically Connected Object

Thinning

The pixel C *connects* the two object segments AB and ED, that is, if C were removed then this would break the object in two; this pixel is *critically connected*.

Thinning

Given a pixel, labelled 8 and its eight adjacent neighbours, labelled 0-7



Thinning

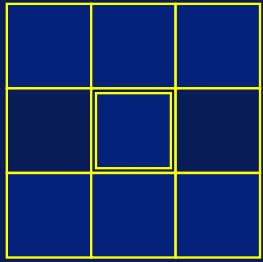
Assume that writing the pixel number (*e.g.* 7) indicates presence, *i.e.* it is an object pixel.

Thinning

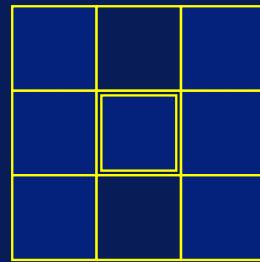
Whereas writing it with an over-bar (e.g. $\bar{7}$) indicates absence, it is a background pixel.

Then pixel 8 is critically connected if the following expression is true.

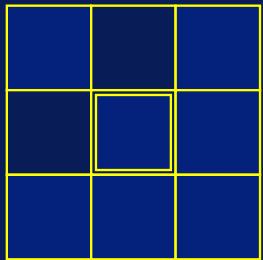
$$\begin{aligned} & 8 \cdot \left\{ \left[(1+2+3) \cdot (5+6+7) \cdot \bar{4} \cdot \bar{0} \right] \right. \\ & + \left[(1+0+7) \cdot (3+4+5) \cdot \bar{2} \cdot \bar{6} \right] \\ & + \left[3 \cdot (5+6+7+0+1) \cdot \bar{2} \cdot \bar{4} \right] \\ & + \left[1 \cdot (3+4+5+6+7) \cdot \bar{2} \cdot \bar{0} \right] \\ & + \left[7 \cdot (1+2+3+4+5) \cdot \bar{0} \cdot \bar{6} \right] \\ & \left. + \left[5 \cdot (7+0+1+2+3) \cdot \bar{4} \cdot \bar{6} \right] \right\} \end{aligned}$$



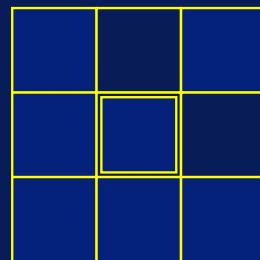
$[(1+2+3) . (5+6+7) . \bar{4} . \bar{0}]$



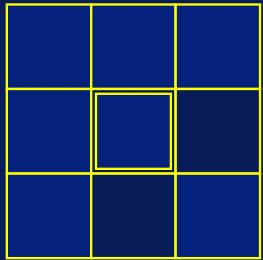
$[(1+0+7) . (3+4+5) . \bar{2} . \bar{6}]$



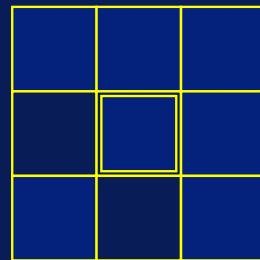
$[3 . (5+6+7+0+1) . \bar{2} . \bar{4}]$



$[1 . (3+4+5+6+7) . \bar{2} . \bar{0}]$



$[7 . (1+2+3+4+5) . \bar{0} . \bar{6}]$



$[5 . (7+0+1+2+3) . \bar{4} . \bar{6}]$

Neighbourhoods exhibiting Critical Connectivity

Thinning

A thinning algorithm should also preserve an object's length.

To facilitate this, a third restriction must be imposed such that arc-ends, *i.e.*, object pixels which are adjacent to just one other pixel, must not be deleted.

Thinning

Note that a thinned image should be invariant under the thinning operator, *i.e.* the application of a thinning algorithm to a fully thinned image should produce no changes.

This is also important as it provides us with a condition for stopping the thinning algorithm.

Thinning

Since the pixels of a fully thinned image are either critically-connected or are arc-ends, imposing restrictions 2 and 3 allows this property to be fulfilled.

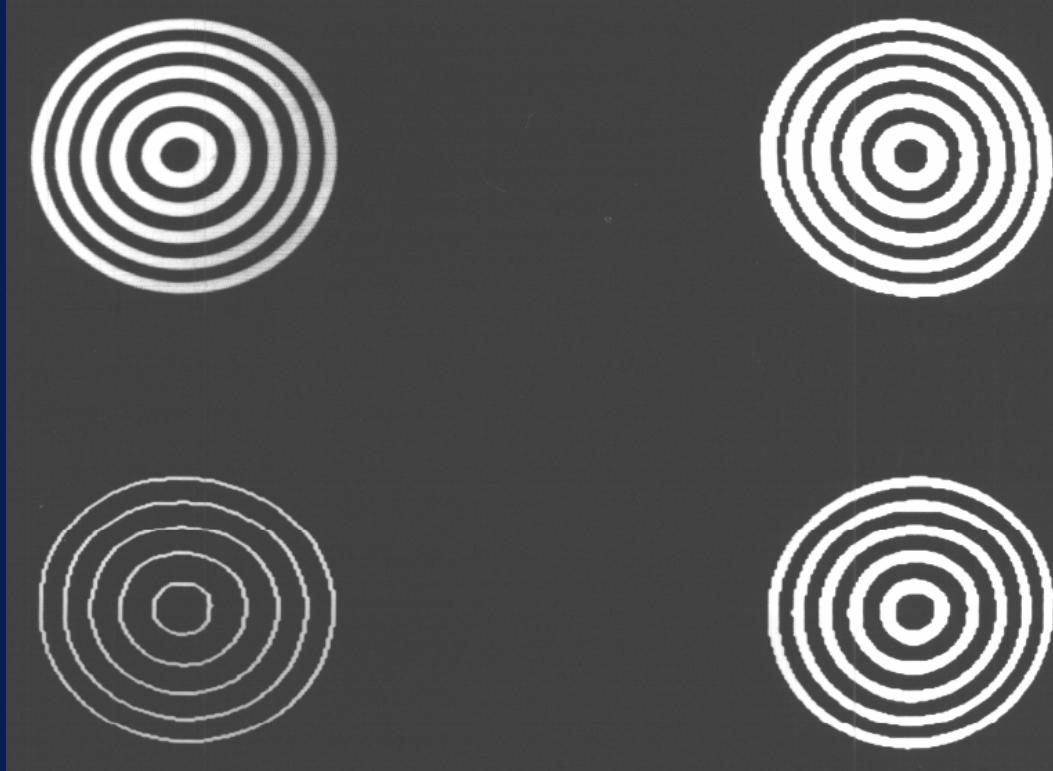
Thinning

The final thinning algorithm

- scan the image in a raster fashion,
- removing of all object pixels according to these three restrictions
- varying border from pass to pass.

Thinning

The image is thinned until four successive passes (corresponding to the four border orientations) producing no changes to the image are made, at which stage, thinning ceases.



A grey-scale image (top left) is thresholded to produce a binary image (top right) which is then thinned (bottom right). The image at the bottom left is an intermediate partially thinned version.

Erosion and Dilation

The concepts of erosion and dilation are related to thinning in the sense that

- erosion can be considered as a single pass of a thinning operator (stripping object pixels of all four border orientations).
- **Significant Difference:** one does not mind if the object breaks in two and, hence, the complex check for critical connectivity is no longer required.

Erosion and Dilation

- The dilation operation effects the reverse of erosion, *i.e.*, an expansion of the object into all those background pixel cells which border the object.

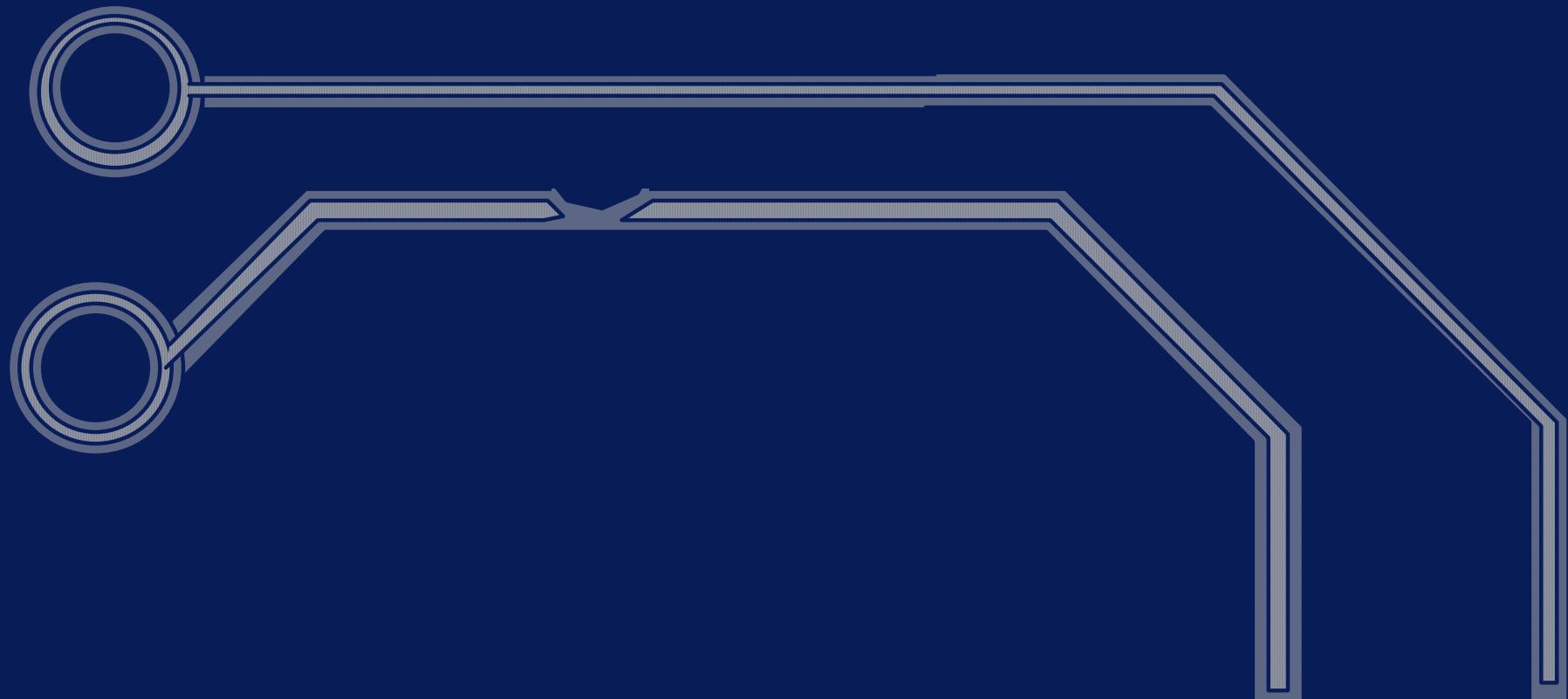
Thus, erosion *shrinks* an object while dilation *enlarges* it.

Erosion and Dilation



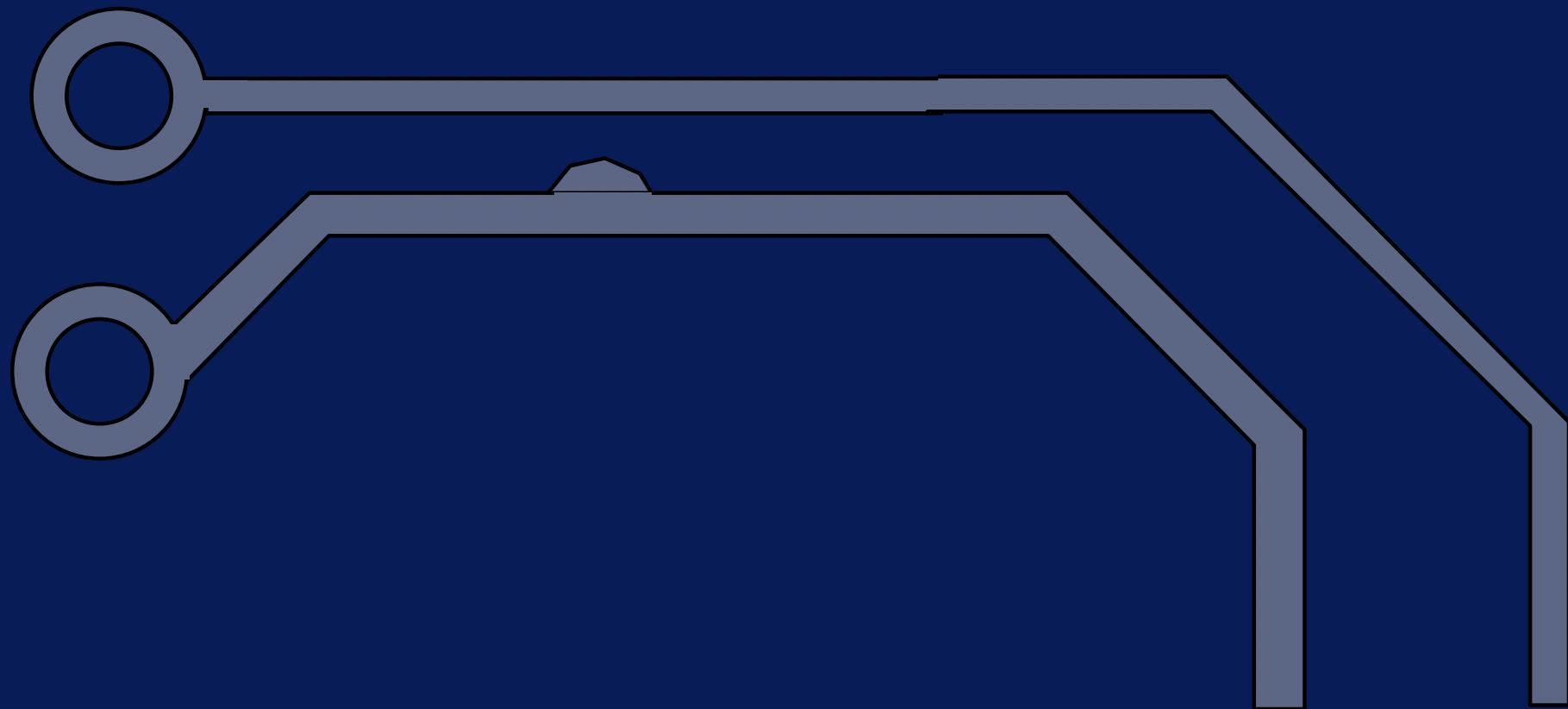
PCB track with a neck

Erosion and Dilation



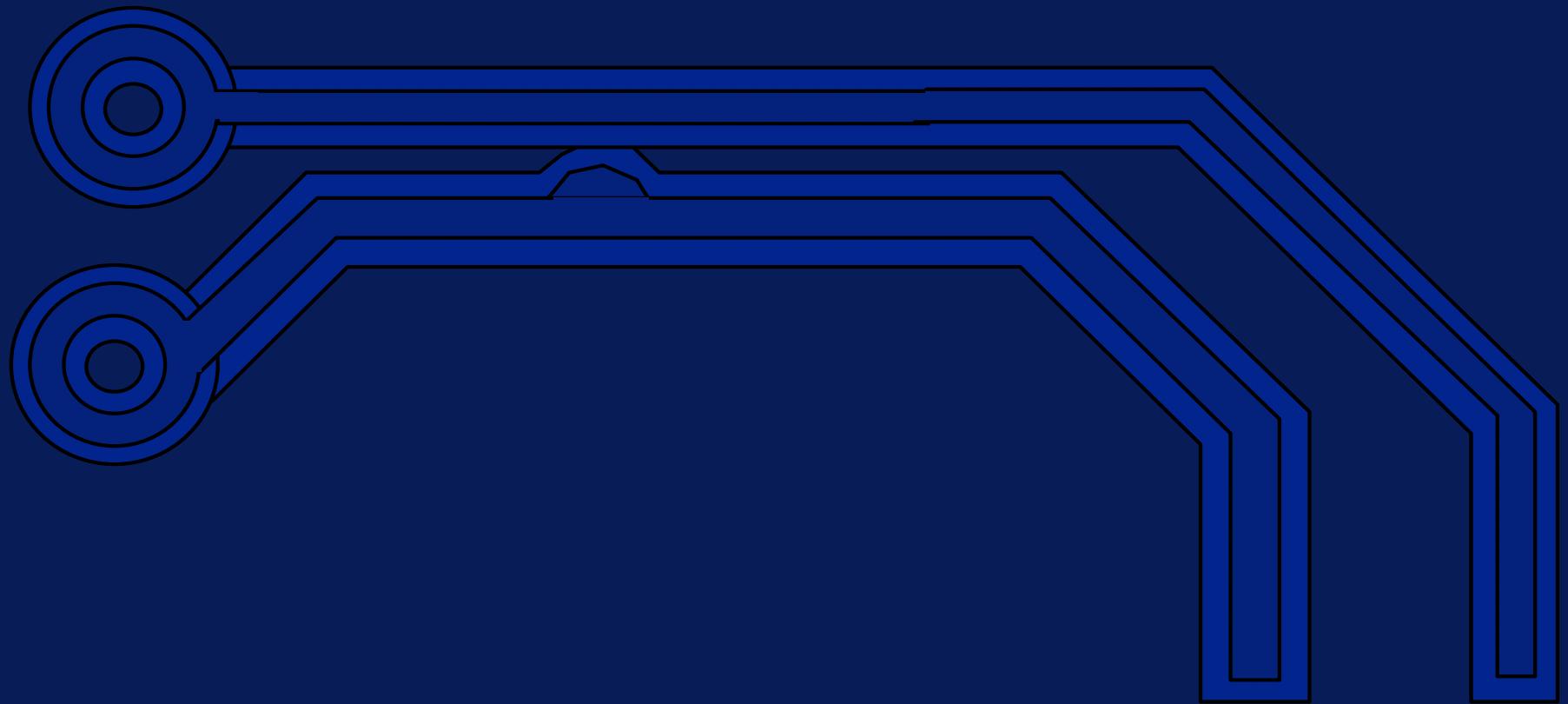
Eroded PCB Track

Erosion and Dilation



PCB Track with extraneous Copper

Erosion and Dilation



Dilated PCB Track

Erosion and Dilation

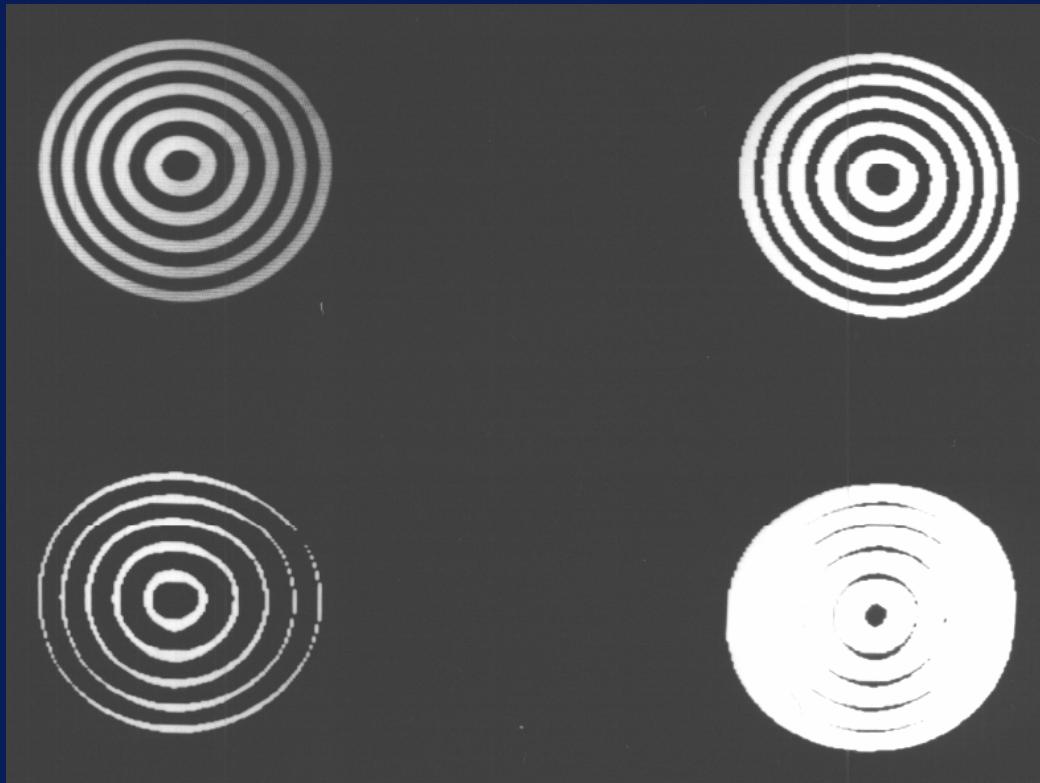
A pseudo-code algorithm for erosion can be formulated as follows :

```
/* erosion */  
FOR all pixels in the image  
    IF the pixel is an object pixel AND all  
    its neighbours are object pixels  
    {  
        copy it to the destination image  
    }
```

Erosion and Dilation

while a dilation algorithm can be formulated as :

```
/* dilation */  
FOR all pixels in the image  
    IF the pixel is an object pixel  
        make it and its eight  
        neighbours object pixels in  
        the destination image
```



A grey-scale image (top left) is thresholded to produce a binary image (top right) which is then eroded twice (bottom left). The application of two passes of the dilation algorithm is shown at the bottom right.

Geometric Operations

Geometric operations change the spatial relationship between objects in an image

- The relative distances between points a, b and c will typically be different after a geometric operation or “warping”.

The applications of such warping include

- Geometric Decalibration, *i.e.* the correction of geometric distortion introduced by the imaging system
- Image Registration, *i.e.* the intentional distortion of one image with respect to another so that the objects in each image superimpose on one another.

Spatial Warping

The approach to geometric image manipulation described here is called *Spatial Warping* and involves :

- the computation of a mathematical model for the required distortion
- its application to the image
- and the creation of a new corrected (decalibrated or registered) image.

The distortion may be specified by

- locating control points (also called fiducial points) in the input image (the image to be warped)
- identifying their corresponding control points in an ideal (undistorted or registered) image.

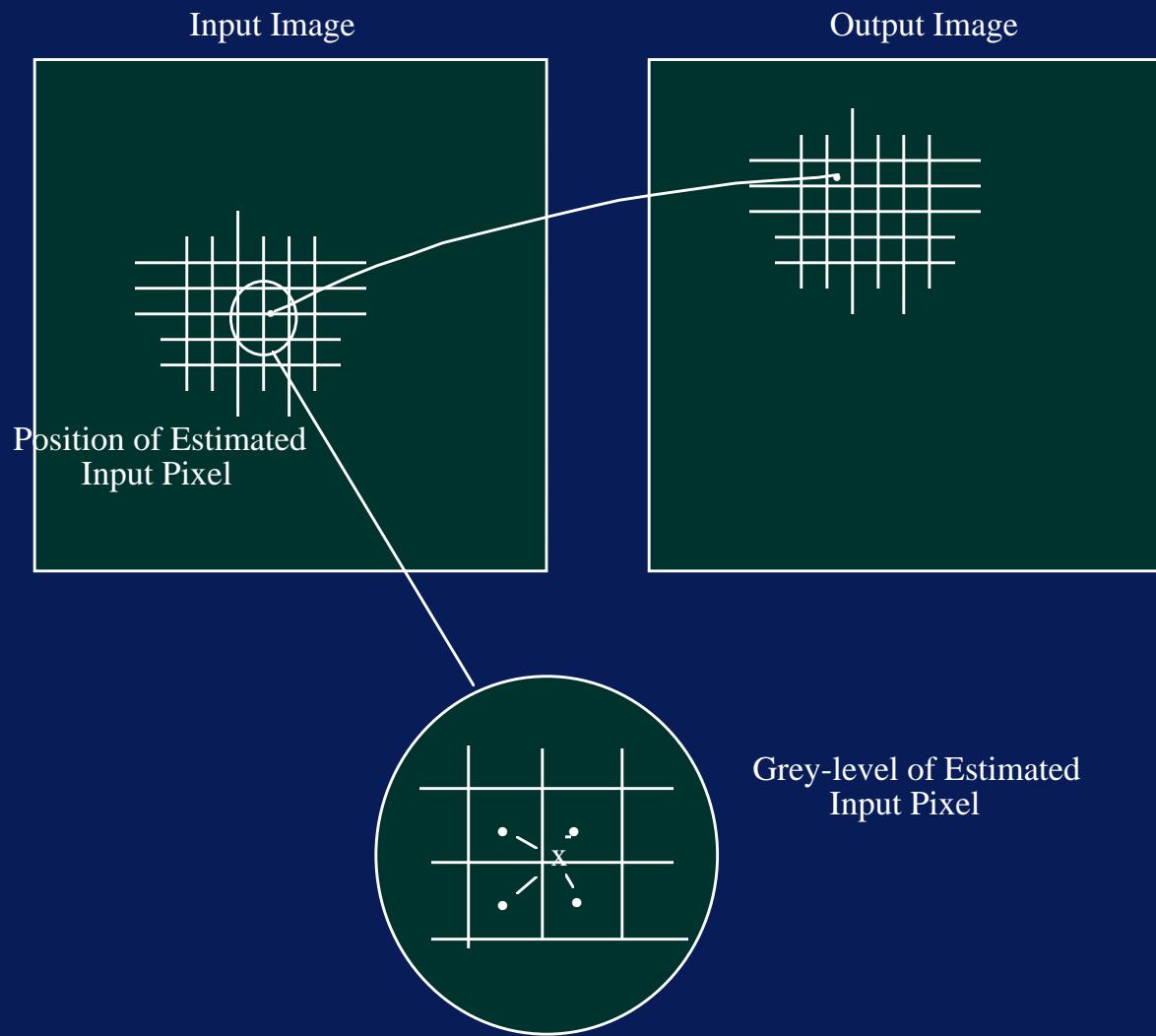
The distortion model is then computed in terms of the transformation between these control points

- generating a spatial warping function
- which will allow one to build the output image pixel by pixel
- by identifying the corresponding point in the input image.

The estimates of the co-ordinates of input pixels yielded by the warping function will not correspond to exact (integer) pixel locations.

We also require some method of estimating the grey-level of the output pixel when the “corresponding” pixel falls “between the integer co-ordinates”.

The question is : how do the four pixels surrounding the computed point contribute to our estimate of its grey-level, *i.e.*, how do we interpolate between the four?



Spatial Transformation and Grey-Level Interpolation

The two requirements of geometric operations

:

- a) A spatial transformation which allows one to derive the position of a pixel in the input which corresponds to the pixel being “filled” or generated in the output.
- b) An interpolation scheme to estimate the grey level of this input pixel.

The Spatial Transformation

The spatial transformation is expressed in general form as a mapping from a point (x, y) in the output image to its corresponding (warped) position (i, j) in the input image.

$$(i, j) = (W_x(x, y), W_y(x, y))$$

- the first co-ordinate, i , of the warped point is a function of the current position in the output
- likewise for the second co-ordinate, j .

Thus, given any point (x, y) in the output image, the co-ordinates of the corresponding point in the input image may be generated using the warping functions W_x and W_y respectively.

It would, of course, be ideal if we had some analytic expression for W_x and W_y but this is rarely the case.

Instead, we normally model each spatial warping function by a polynomial function.

So, we assume that, for example, the warping functions are given by the following equations :

$$W_x(x, y) = \sum_p^n \sum_q^n a_{pq} x_p y_q$$

$$W_y(x, y) = \sum_p^n \sum_q^n b_{pq} x_p y_q$$

For example, if $n=2$ (which is adequate to correct for most distortions) :

$$W_x(x, y) = a_{00}x^0y^0 + a_{10}x^1y^0 + a_{20}x^2y^0 + \\ a_{01}x^0y^1 + a_{11}x^1y^1 + a_{21}x^2y^1 + \\ a_{02}x^0y^2 + a_{10}x^1y^2 + a_{22}x^2y^2$$

$$\begin{aligned} W_y(x, y) = & b_{00}x^0 y^0 + b_{10}x^1 y^0 + b_{20}x^2 y^0 + \\ & b_{01}x^0 y^1 + b_{11}x^1 y^1 + b_{21}x^2 y^1 + \\ & b_{02}x^0 y^2 + b_{10}x^1 y^2 + b_{22}x^2 y^2 \end{aligned}$$

Now, the only thing that remains to complete the specification of the spatial warping function is to determine the values of these coefficients, *i.e.* to compute

$$a_{00} - a_{22} \text{ and } b_{00} - b_{22}.$$

To do this, we have to assume that we know the transformation exactly for a number of points (at least as many as the number of coefficients; nine in this case), that is, to assume that we know the values of x and y and their corresponding i and j values.

We then write the relationships explicitly in the form of the two equations above.

We then solve these equations simultaneously to determine the value of the coefficients.

Remember that we have two sets of simultaneous equations to set up

- one for the “ a ” coefficients
- one for the “ b ” coefficients.

However, the same values relating x, y to i, j
can be used in each case.

This is now where the control points come in
as we are going to use these to provide us
with the (known) relationships between (x, y)
and (i, j) .

If we have nine unknown coefficients, as in the example above, then in order to obtain a solution we require at least nine such observations,

$$\left\{ \left(x_1, y_1 \right), \left(i_1, j_1 \right) \right\} \dots \left\{ \left(x_9, y_9 \right), \left(i_9, j_9 \right) \right\}$$

say.

Such a system is said to be exactly determined.

However, the solution of these exact systems are often ill-conditioned (numerically unstable) and it is usually good practice to over-determine the system by specifying more control points than you need (and hence generate more simultaneous equations).

The first point to note is that an over-determined system does not have an exact solution : there are going to be some errors for some points.

The idea, then, is to minimise these errors.

We will use the common approach of minimising the sum of the square of each error (*i.e.* to generate the so-called least-square-error solution).

Consider, again, a single control point and assume we are attempting to compute the coefficients : a_{pq}

$$\begin{aligned} i_1 = & a_{00}x_1^0 y_1^0 + a_{10}x_1^1 y_1^0 + a_{20}x_1^2 y_1^0 + \\ & a_{01}x_1^0 y_1^1 + a_{11}x_1^1 y_1^1 + a_{21}x_1^2 y_1^1 + \\ & a_{02}x_1^0 y_1^2 + a_{12}x_1^1 y_1^2 + a_{22}x_1^2 y_1^2 \end{aligned}$$

If we use m control points in total we will have m such equations which (noting that x^0 and y^0 both equal to 1) we may write in matrix form as :

$$\begin{bmatrix} i_1 \\ i_2 \\ \dots \\ i_m \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & y_1^1 & x_1^1 & y_1^1 & x_1^2 & y_1^1 & y_1^1 & x_1^1 & y_1^2 & x_1^2 & y_1^2 \\ 1 & x_2^1 & x_2^2 & y_2^1 & x_2^1 & y_2^1 & x_2^2 & y_2^1 & y_2^1 & x_2^1 & y_2^2 & x_2^2 & y_2^2 \\ \dots & \dots \\ 1 & x_m^1 & x_m^2 & y_m^1 & x_m^1 & y_m^1 & x_m^2 & y_m^1 & y_m^2 & x_m^1 & y_m^2 & x_m^2 & y_m^2 \end{bmatrix} * \begin{bmatrix} a_{00} \\ a_{10} \\ \dots \\ a_{22} \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_m \end{bmatrix}$$

We have to include the errors since there won't be a set of $a_{00} - a_{22}$ which will simultaneously provide us with exact $i_1 - i_m$ in the over-determined case.

101	100	103	105	107	105	103	110
110	140	120	122	130	130	121	120
134	134	135	131	137	138	120	121
132	132	132	133	133	150	160	155
134	140	140	135	140	156	160	174
130	138	139	150	169	175	170	165
126	133	138	149	163	169	180	185
130	140	150	169	178	185	190	200

Nearest Neighbour

Computed Point

Nearest Neighbour Interpolation

Let us abbreviate this matrix equation to :

$$i = Xa + e$$

Similarly

$$j = Xb + e.$$

We require a so we might think of multiplying across by X^{-1} to obtain an appropriate expression.

Unfortunately, X is non-square (number of equations is greater than the number of coefficients) and one cannot invert a non-square matrix).

However, we can indulge in a little algebra and calculus to derive an expression for a in terms of X and i .

$$i = Xa + e$$

$$e = i - Xa$$

We form the sum of the square of each error by computing $e^T e$:

$$e^T e = (i - Xa)^T (i - Xa)$$

Differentiating $e^T e$ with respect to a , to find out how the errors change as the coefficients change :

$$\begin{aligned}\frac{d(e^T e)}{d(a)} &= (0 - XI)^T(i - Xa) + (i - Xa)^T(0 - XI) \\ &= (-XI)^T(i - Xa) + \left(i^T - (Xa)^T\right)(-XI) \\ &= -IX^T(i - Xa) + \left(i^T - a^T X^T\right)(-XI) \\ &= -IX^T i + IX^T Xa - i^T XI + a^T X^T XI\end{aligned}$$

But noting that $i^T XI$ and $a^T X^T XI$ are 1*1 matrices and that the transpose of a 1*1 matrix is equal to itself, we transpose these two sub-expressions :

$$\begin{aligned}\frac{d(e^T e)}{d(a)} &= -IX^T i + IX^T Xa - IX^T i + IX^T Xa \\ &= 2(I)(X^T Xa - X^T i)\end{aligned}$$

The sum of the square of each error is minimised when $\frac{d(e^T e)}{d(a)}$ is equal to zero, thus :

$$\begin{aligned}0 &= 2(I)(X^T Xa - X^T i) \\(X^T X)^{-1} X^T Xa &= (X^T X)^{-1} X^T i \\a &= (X^T X)^{-1} X^T i\end{aligned}$$

$(X^T X)^{-1} X^T$ is commonly referred to as the pseudo-inverse of X and is written as X^\dagger .

When this has been computed, the coefficient matrix a may be computed by simply multiplying X^\dagger by i ; b is obtained in a similar manner.

Grey-Level Interpolation

Once the spatial mapping function has been found, the output image can be built, pixel by pixel and line by line.

The co-ordinates given by the warping function, denoting the corresponding points in the input image, will not in general be integer values and the grey-level must be interpolated from the grey-level of surrounding pixels.

The simplest interpolation function is “nearest neighbour” interpolation (zero-order interpolation)

- the grey level of the output pixel (which is what we are trying to estimate)
- is given by the grey-level of the input pixel which is nearest to the calculated point in the input image.

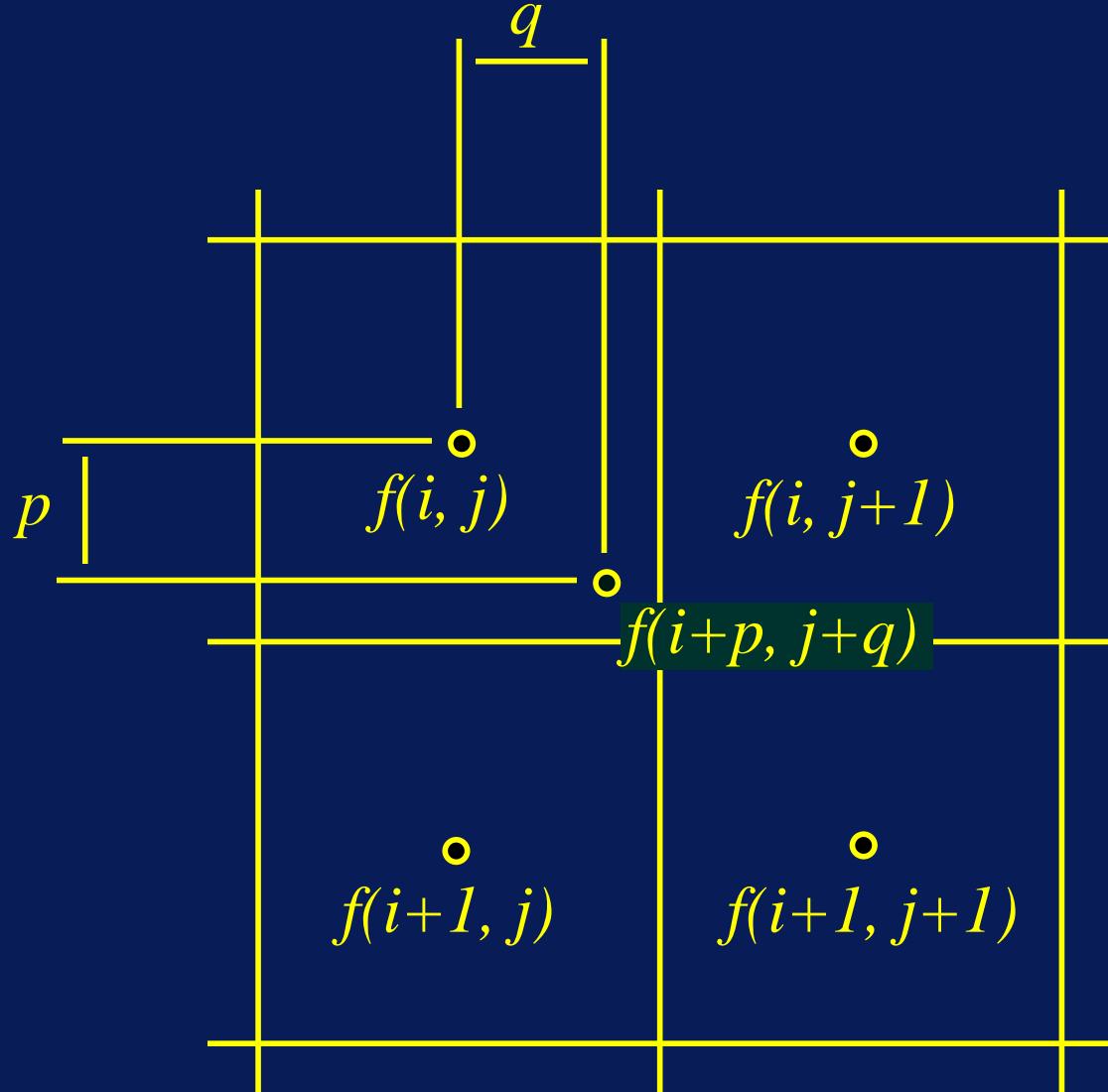
(see Slide 8 *OR* figure 4.22).

The computation involved in this interpolation function is quite trivial but the function generally yields quite adequate results.

If the image exhibits very fine detail in which adjacent pixel grey levels vary significantly, *i.e.* the image exhibits high spatial frequencies, some of this detail may be lost.

In such cases, bi-linear (*i.e.* first order) interpolation should be considered

- estimate is made on the basis of four neighbouring input pixels.



Bilinear Transformation

Consider the case shown in the previous diagram (*Bilinear Interpolation*) where we need to estimate the grey-level of a point somewhere between image pixels

(i, j) $(i, j+1)$, $(i+1, j)$ and $(i+1, j+1)$.

Let the position of this point relative to pixel (i, j) be given by co-ordinates

$$(p, q); 0 \leq p, q \leq 1$$

The grey level at point (p, q) is constrained by the grey-level at the four neighbouring pixels and is a function of its position between these neighbours.

To estimate the grey-level, $f(p, q)$, we fit a surface through the four neighbours' grey levels.

- the equation of which will identify in general the grey-level at any point between the neighbours.

The surface we fit is a hyperbolic paraboloid and is defined by the bilinear equation:

$$f(p, q) = ap + bq + cpq + d$$

There are four coefficients, a , b , c and d , which we must determine to identify this function for any given 2*2 neighbourhood in which we wish to interpolate.

Thus we require four simultaneous equations in a , b , c and d ; these are supplied from our knowledge of the grey-levels at the four neighbours given by relative co-ordinates $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$.

Specifically, we know that :

$$a x 0 + b x 0 + c x 0 x 0 + d = f(i, j) \quad (4.1)$$

$$a x 0 + b x 1 + c x 0 x 1 + d = f(i, j+1) \quad (4.2)$$

$$a x 1 + b x 0 + c x 1 x 0 + d = f(i+1, j) \quad (4.3)$$

$$a x 1 + b x 1 + c x 1 x 1 + d = f(i+1, j+1) \quad (4.4)$$

Directly from (4.1), we have :

$$d = f(i, j) \quad (4.5)$$

Rearranging (4.2) and substituting for d , we have :

$$b = f(i, j+1) - f(i, j) \quad (4.6)$$

Rearranging (4.3) and substituting for d , we have :

$$a = f(i+1, j) - f(i, j) \quad (4.7)$$

Rearranging (4.4) and substituting for a , b and d , we have :

$$c = f(I+1, j+1) + f(I, j) - f(I+1, j) - f(I, j+1) \quad (4.8)$$

Equations (4.5) - (4.8) allow us to compute the coefficients a , b , c and d ; which define the bilinear interpolation for a given 2*2 neighbourhood with known pixel grey-levels.

For example, if the co-ordinates of the point at which we wish to estimate the grey-level are $(60.4, 128.1)$ and the grey-levels at pixels $(60, 128)$, $(60, 129)$, $(61, 128)$ and $(61, 129)$ are 10 , 12 , 14 and 15 , respectively, then the grey level at this point, in relative co-ordinates, is given by:

$$\begin{aligned}f(0.4, 0.1) &= (14 - 10) * 0.4 + \\&\quad (12 - 10) * 0.1 + \\&\quad (15 + 10 - 14 - 12) * 0.4 * 0.1 + \\&\quad 10 \\&= 11.76\end{aligned}$$

Registration and Geometric Decalibration

This technique for spatial warping can be directly used to effect

- registration of two images
- correct the geometric distortion which may have been introduced by the imaging system.

In the former case, one merely needs to :

- identify several corresponding points in the two images and
- use these as the control points when generating the polynomial coefficients.

In the latter case :

- one might use the imaging system to generate an image of a test card (*e.g.* a square grid) and
- superimpose an undistorted copy of this pattern on the distorted image.

- The corresponding control points can then be explicitly identified in each version of the pattern,
- for example, in the case of the grid pattern, the control points might be the points of intersection of the grid lines.

Mathematical Morphology

Basic Set Theory

Mathematical Morphology is a methodology for image processing and image analysis which is based on set theory and topology.

- Points and Vectors will be denoted by Latin lowercase letters : $x, y, z;$
- Sets will be denoted by Latin uppercase letters : $X, Y, Z.$

The symbol ϕ denotes the empty set.

– *Set Inclusion :*

This is written : $Y \subset X$, i.e. Y is a subset of (is included in) the set X .

This is defined as $y \in Y \Rightarrow y \in X$: If y is an element of Y , then y is also an element of X .

- *Complement* :

For any set X , the complement of X is written X^c
This is the set of all elements which are not
elements of X .

- *Union* :

The union of two sets X and Y , written $X \cup Y$
is defined : $X \cup Y = \{x \mid x \in X \text{ or } x \in Y\}$

This should be read: the set X union Y is the set of all x such that x is an element of X or x is an element of Y .

– *Intersection* :

The intersection of two sets X and Y ,
written $X \cap Y$, is defined :

$$X \cap Y = (X^c \cup Y^c)^c$$

In effect, the intersection of two sets is the complement of the union of their respective complements.

Thus, the intersection of X and Y is the complement of the set of elements which are not in X or not in Y , i.e. *the set of elements which is common to both sets X and Y .*

Let us now consider two further concepts.
The first is translation.

The translation of a set X by h is denoted by

. X_h

This is a set where each element (point) is translated by a vector h .

Second, we need to introduce a general symbolism for set transformation.

A transformation of a set X is denoted by $\Psi(X)$. $\Psi(\)$ is the set transformation and in an expression such as $Y = \Psi(X)$, Y is the transformed set.

Finally, we required the concept of *Duality* of set transformations.

Given some transformation Ψ , we define the dual of the transformation Ψ^* :

$$\Psi^*(X) \rightarrow (\Psi(X^c))^c$$

For example, intersection is the dual of union since $X \cap Y = (X^c \cup Y^c)^c$.

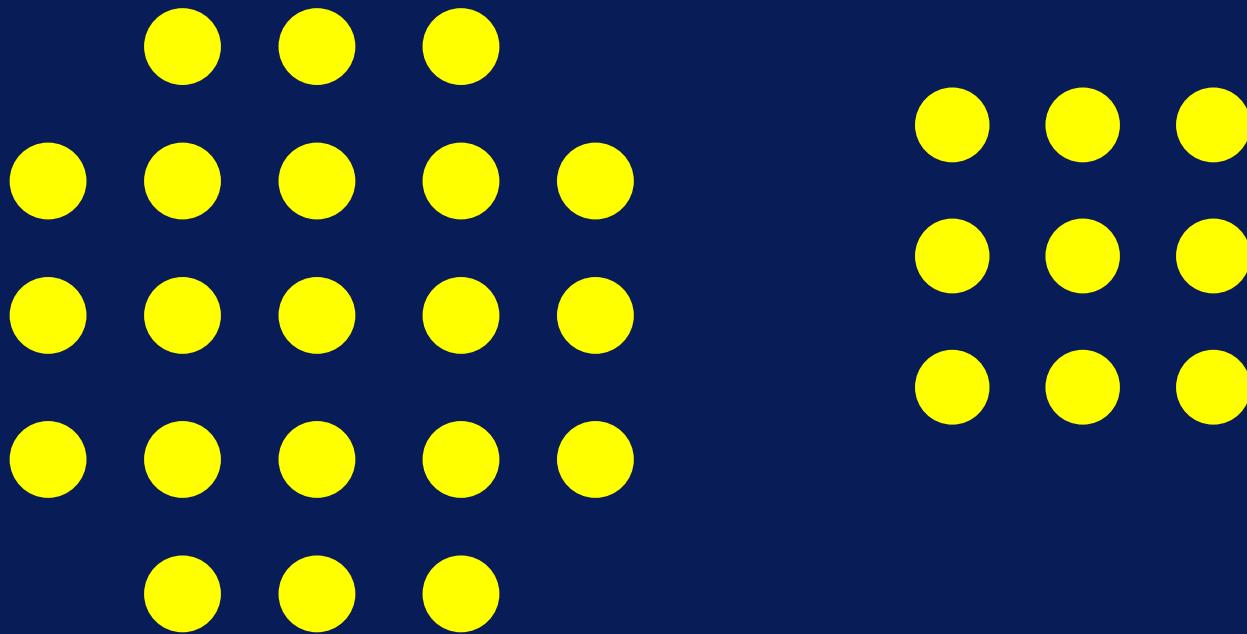
Structuring Elements and Hit or Miss Transformations

We are now in a position to continue with the discussion of mathematical morphology.

Let us begin with the concept of a *Structuring Element*.

A structuring element B_x , centred at x , is a set of points which is used to “extract” structure in a set, X , say.

For example, the structuring element might be a square or a disk, as shown in next diagram, or any other appropriate shape.



Square and Circular structuring elements

Now we define a *Hit or Miss Transformation* as the “point by point” transformation of a set X , working as follows.

We choose, and fix, a structuring element B .

Define

- B_x^1 to be that subset of B_x (recall that B_x is the translate of B to a position x) whose elements belong to the “foreground”.
- B_x^2 to be the subset of B_x whose elements belong to the “background”

i.e.

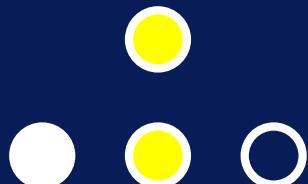
$$B^1 \cap B^2 = \emptyset$$

A point x belongs to the Hit or Miss transform, denoted $X \otimes B$, if and only if B_x^1 is included in X and B_x^2 is included in X^c , the complement of X :

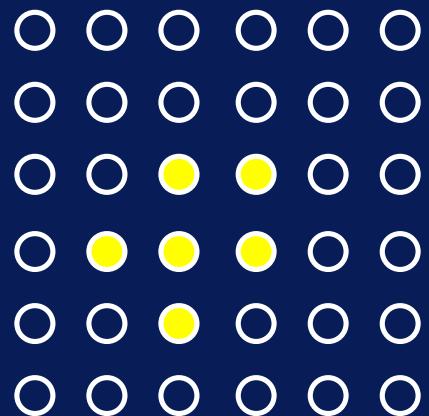
$$X \otimes B = \{x \mid B_x^1 \subset X; B_x^2 \subset X^c\}$$

Thus $X \otimes B$ defines the points where the structuring element B exactly matches (hits) the set X , *i.e.* the image.

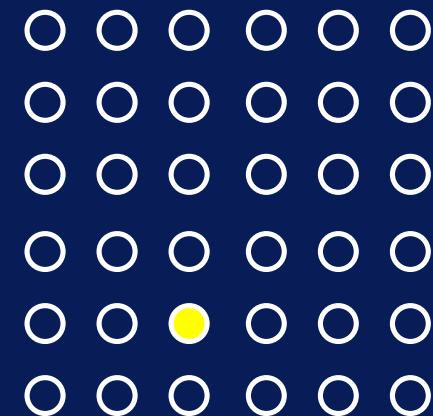
For example, let B be structuring element shown in figure 4.26 (a) and let X be the set shown in figure 4.26 (b). Then $X \otimes B$ is the set shown in figure 4.26 (c).



(a)



(b)



(c)

*(a) Structuring element B; (b) Image set X;
(c) B “hit or miss” X : $X \otimes B$*

Erosion and Dilation

In section 4.2.3, we informally introduced the concepts of erosion and dilation. We will now define them formally.

Let $\overset{\cup}{B}$ be the transposed set of B , *i.e.* the symmetrical set of B with respect to its origin.

Then, the erosion operation is denoted Θ and the erosion of a set X with B is defined :

$$X \Theta \overset{\circ}{B} = \{x \mid B_x \subset X\}$$

Note that this is equivalent to a Hit or Miss transformation of X with B , where

$B^2 = \Phi$, i.e. where there are no background points.

Note also that $X \Theta B$ is not an erosion - it is the *Minkowski Subtraction* of B from X .

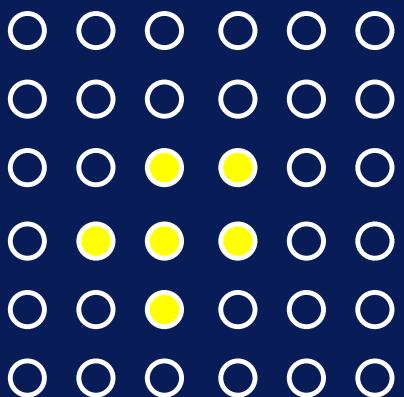
Intuitively, the erosion of a set by a structuring element amounts to the generation of a new (transformed/eroded) set where each element in the transformed set is a point where the structuring element is included in the original set X .

For example, let B and X be the structuring element and set shown in figure 4.27 (a) and (b) respectively; then $X \Theta B$ is depicted in figure 4.27 (c).

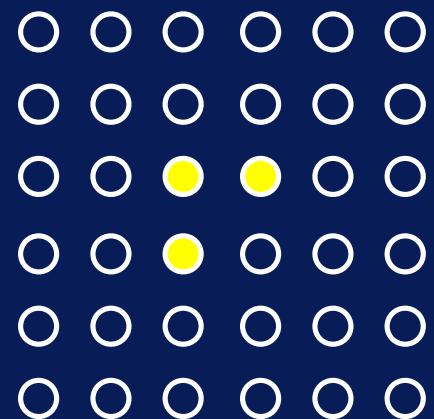
In a more complex case, if B is a circular structuring element, figure 4.28 schematically illustrates the effect of erosion.



(a)

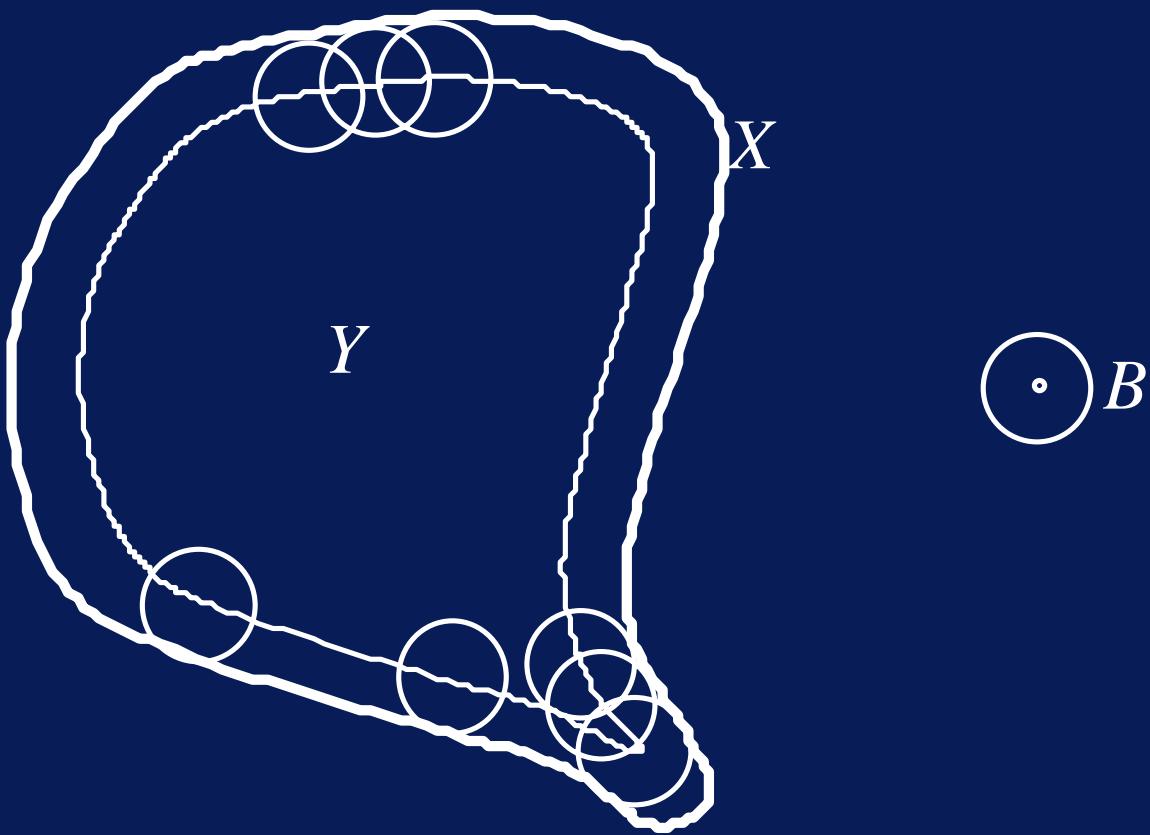


(b)



(c)

*(a) Structuring element B; (b) Image set X;
(c) the erosion of X with B : $X \ominus B$*



Erosion of X by a circular structuring element B .

Dilation is a closely related operation. In fact, dilation is dual of erosion. The dilation operation is denoted by the symbol \oplus . Thus,

$$X \oplus \overset{\circ}{B} = \left(X^c \Theta \overset{\circ}{B} \right)^c$$

That is, the dilation of X is the erosion of X^c

This amounts to saying that the erosion of a set (*e.g.* an object) with a given structuring element is equivalent to the dilation of its complement (*i.e.* its background) with the same structuring element, and *vice versa*.

Opening and Closing

After having eroded X by B , it is not possible in general to recover the initial set by dilating the eroded set $X \ominus B$ by the same B .

This dilate reconstitutes only a part of X ,

- which is simpler and has less details
- but may be considered as that part which is most essential to the structure of X .

This new set (*i.e.* the results of erosion followed by dilation) filters out (*i.e.* generates) a new subset of X which is extremely rich in morphological and size distribution properties.

This transformation is called an *Opening*.

The opening of a set X with a structuring element B , denoted X_B , is defined

$$X_B = \left(X \Theta \overset{\cup}{B} \right) \oplus B$$

The closing of X with respect to B , denoted X^B is defined

$$X_B = (X \oplus B) \Theta \overset{\cup}{B}$$

Opening is the dual of closing, *i.e.* :

$$(X^c)^B = (X_B)^c$$

and

$$(X_B)^c = (X^c)^B$$

The opening is the domain swept out by all the translates of B which are included in X . This effects

- a smoothing of the contours of X ,
- cuts narrow isthmuses
- suppresses small islands and sharp edges in X .

Thinning and the Extraction of End Points

From the perspective of mathematical morphology, the thinning of a set X by a sequence of structuring elements L , is denoted

that is

$$X \ominus \{L^i\}$$

$$\left(\left(\left(\dots \left((X \ominus L^1) \ominus L^2 \right) \ominus L^3 \right) \dots \ominus L^i \right) \right)$$

XOL is defined :

$$XOL = X / X \otimes L$$

That is, the set X less the set of points in X which hit L .

Thus, if $X \otimes L$ identified border points, and L is appropriately structured to maintain connectivity of a set, then repeated application of the thinning process successively removes border points from a set until the skeleton is achieved.

At this point, further application of the thinning transform yields no change in the skeletal set.

This, of course, should be reminiscent of the thinning algorithm discussed in section 4.2.3.

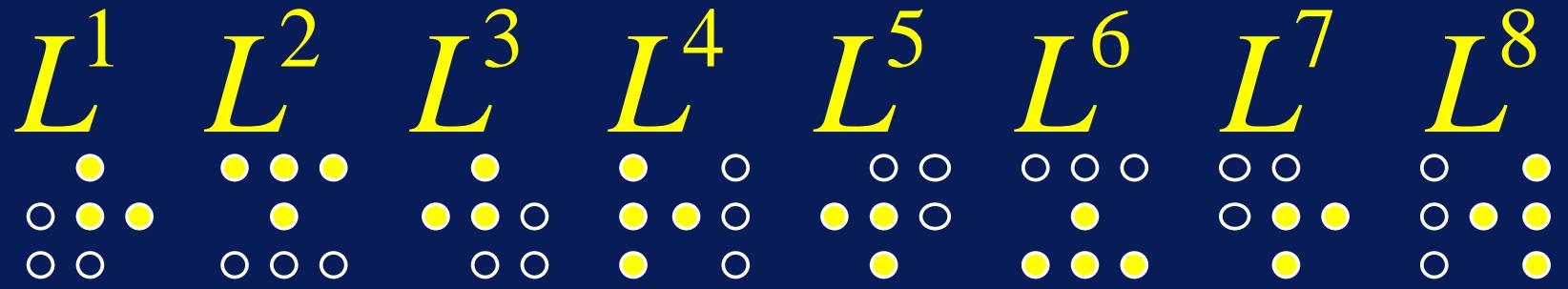
Recalling the definition of a hit and miss transformation :

$$X \otimes L = \{x \mid {}^1L_x \subset X; {}^2L_x \subset X^c\}$$

$${}^1L_x \cap {}^2L_x = \Phi$$

We can now proceed to develop a thinning algorithm by defining L .

The sequence $\{L\}$ which is used for thinning is based on a single structuring element and is generated by rotating the structuring element (through 360° in increments of 45° for a square lattice). This sequence $\{L\}$ is shown in figure 4.29



*Sequence of structuring elements used in the
thinning operation.*

The thinning algorithm then amounts to the repeated transformation of a set

$X_i \rightarrow X_{i+1}$ defined

$$X_{i+1} = \left(\left(\left(\dots \left(X_i \text{OL}_1 \right) \text{OL}_2 \right) \text{OL}_3 \right) \dots \text{OL}_8 \right)$$

The skeleton is achieved when $X_i = X_{i+1}$.

Initially $X_0 = X$, *i.e.*, the original (unthinned) image.

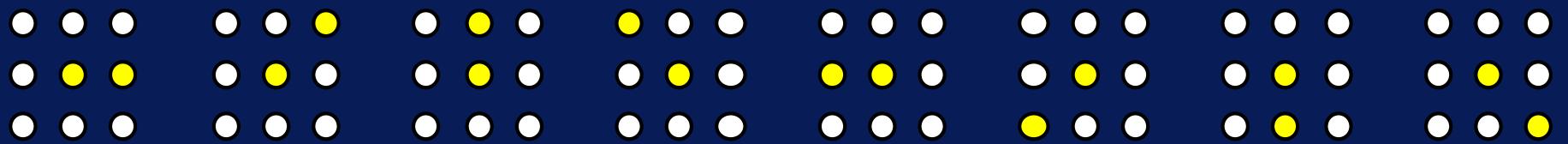
Given a skeleton set X , we can identify the endpoints, *i.e.* points which are connected to just one other point, using the Hit or Miss Transform and an appropriate set of structuring elements $\{E\}$, shown in figure 4.30.

Thus, the endpoints of the skeleton are given by :

$$Y = \bigcup_{i=0}^8 X \otimes E^i$$

That is, the union of all those points which hit with one of these end-point structuring elements.

E^1 E^2 E^3 E^4 E^5 E^6 E^7 E^8



Structuring elements used to identify end-points

*Application : Identification of Endpoints
of Electrical Wires*

Step 1 : Threshold the grey-level image
to obtain a binary image. Call
this set . X_1

Step 2 : Generate the skeleton X_2

$$X_2 = X_1 O \{L\}$$

Step 3 : Identify endpoints of skeleton X_3

$$X_3 = \bigcup_{i=0}^8 X_2 \otimes E^i$$

X_3 is the set of all end-points of these electrical wires.

A Brief Introduction to grey-scale Mathematical Morphology

So far, all of the mathematical morphology has assumed that we are dealing with a set X and its complement X^c

- binary images comprising a foreground (the set of object points X)
- background (the set of all other points X^c)

For the following, we will consider 1D slices
of a 2D image function.

This makes it easier to represent in diagrams
and it is somewhat easier to follow.

Thus, a slice along the X axis (*i.e.* a line of) a grey-scale image can be viewed as shown in figure 4.31 (a).

If we threshold this function choosing values
 $x_\lambda \geq T$, that is, generate a set

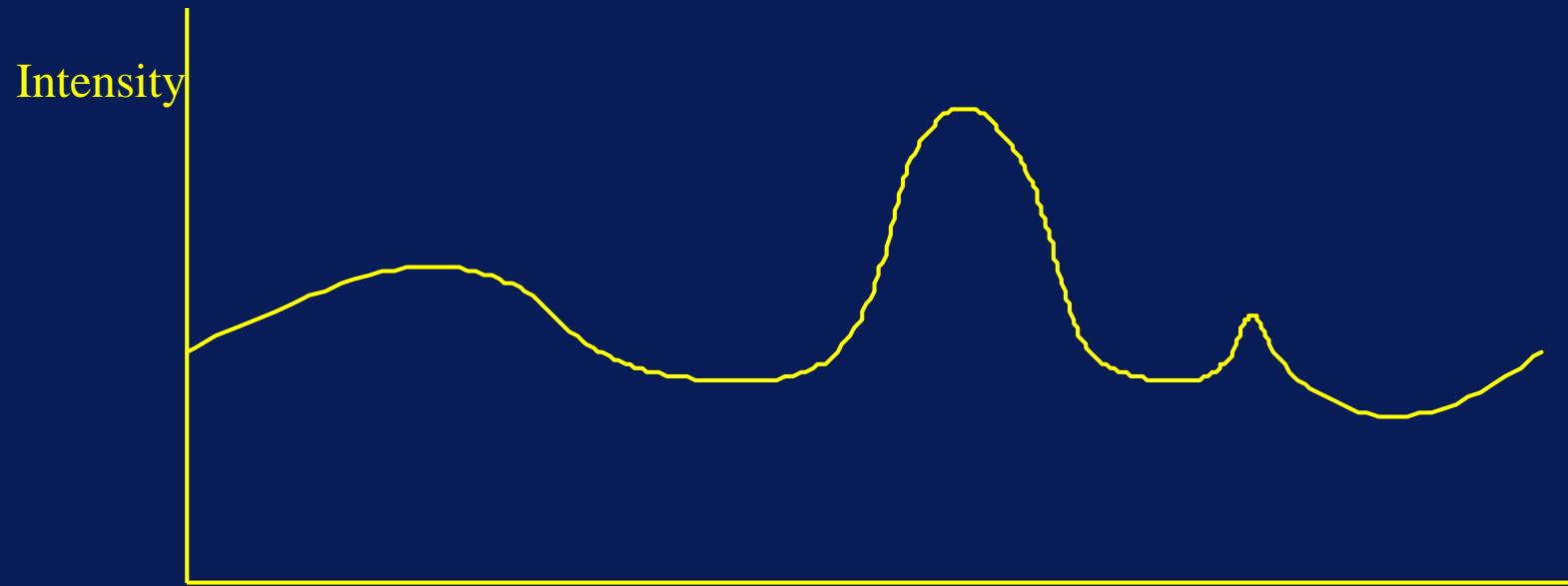
$$x_\lambda = \{x \mid x \geq T\}$$

where T is the threshold value, we generate
the set of points shown in bold horizontal
lines in figure 4.31 (b)

A grey-scale image is then considered to be a function f and is a sequence of sets :

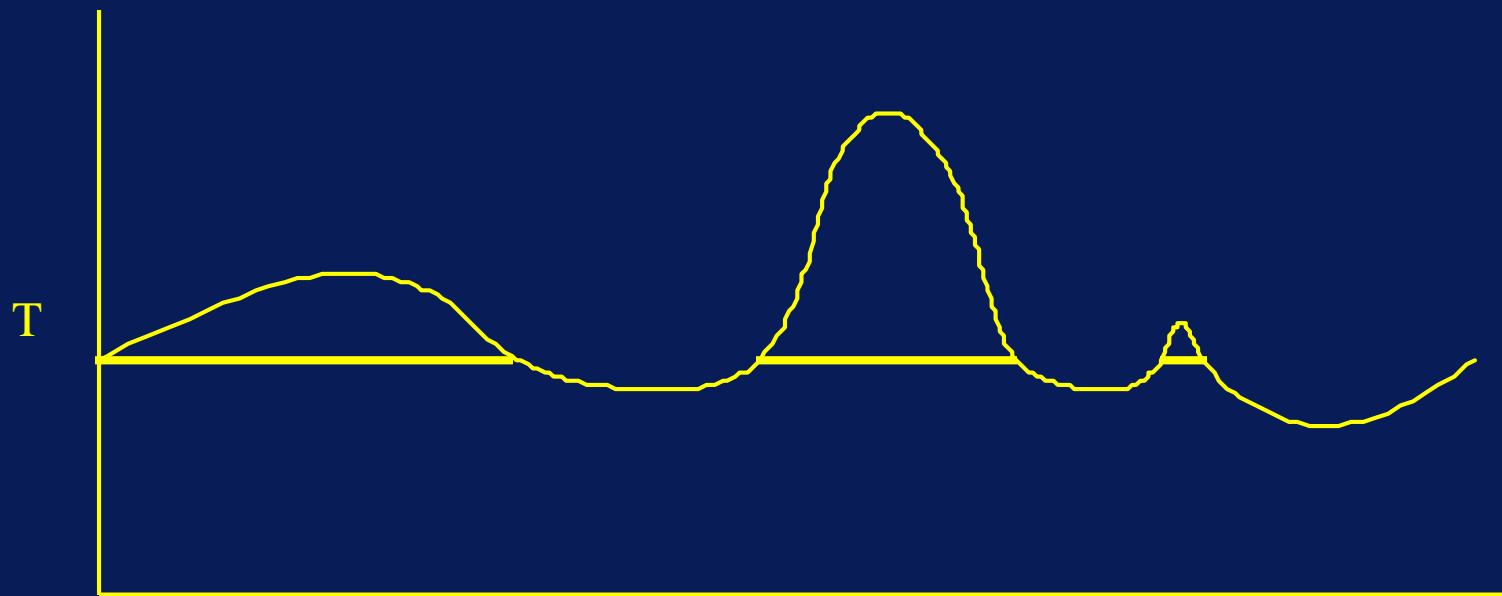
$$f \Leftrightarrow \{x_\lambda(f)\}$$

and the grey-scale image is effectively the entire sequence of sets generated by successively decreasing the threshold level, as shown in figure 4.31 (c)



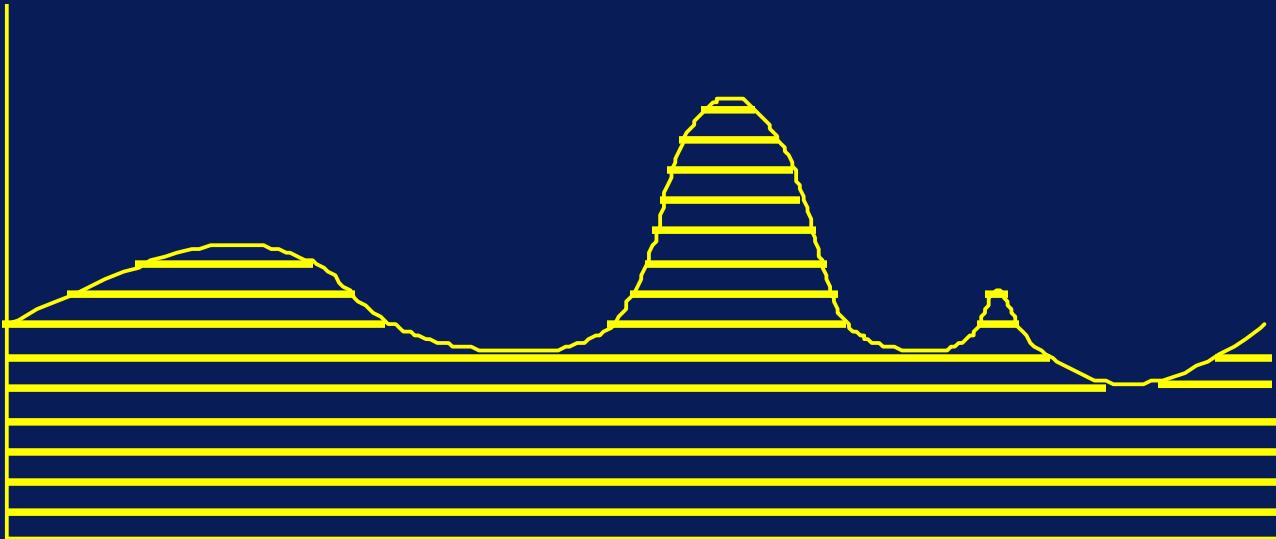
(a)

A 1D slice of a 2D image function



(b)

Thresholding this 1D function at a value T generates a set $x_\lambda = \{x \mid x \geq T\}$ depicted by bold the line.



*Representation of an image by a
sequence of such sets.*

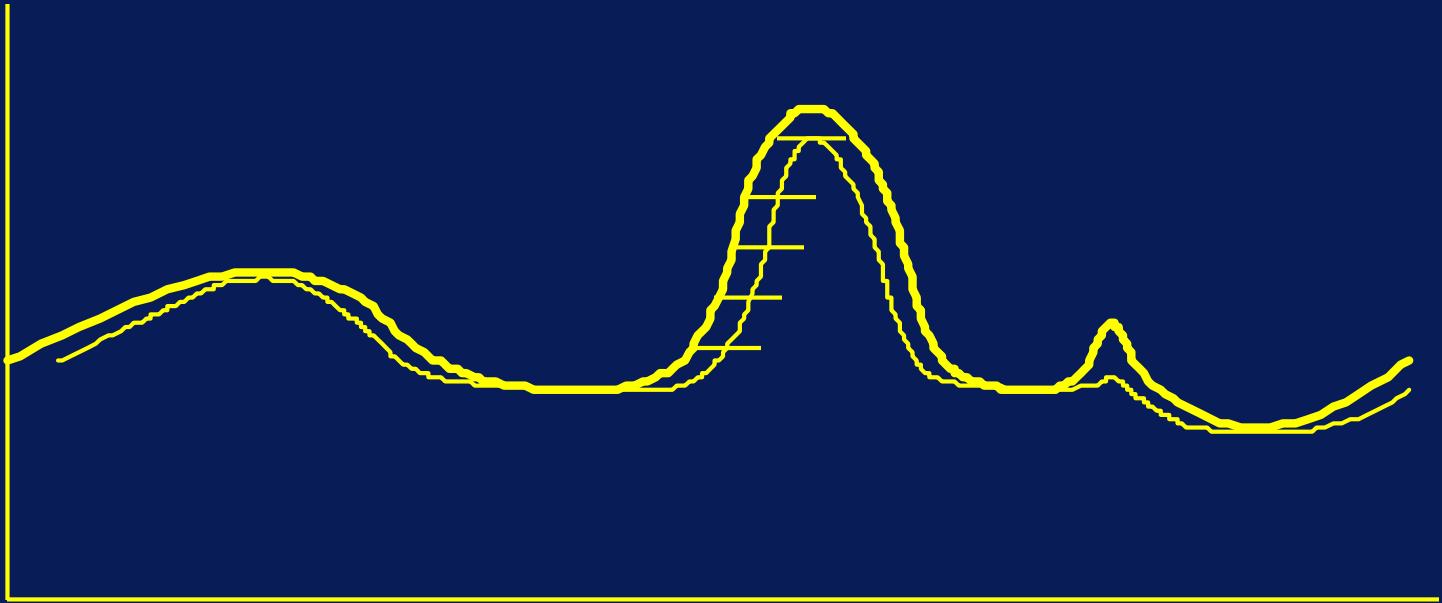
Grey-Level Erosion is defined :

$$f \rightarrow f \Theta \stackrel{\cup}{B}$$

and equivalently :

$$\{x_\lambda(f)\} \rightarrow \left\{x_\lambda(f) \Theta \stackrel{\cup}{B}\right\}$$

This grey-level erosion is the function, or sequence of sets, which are individually the erosion of sets generated at successive thresholds.

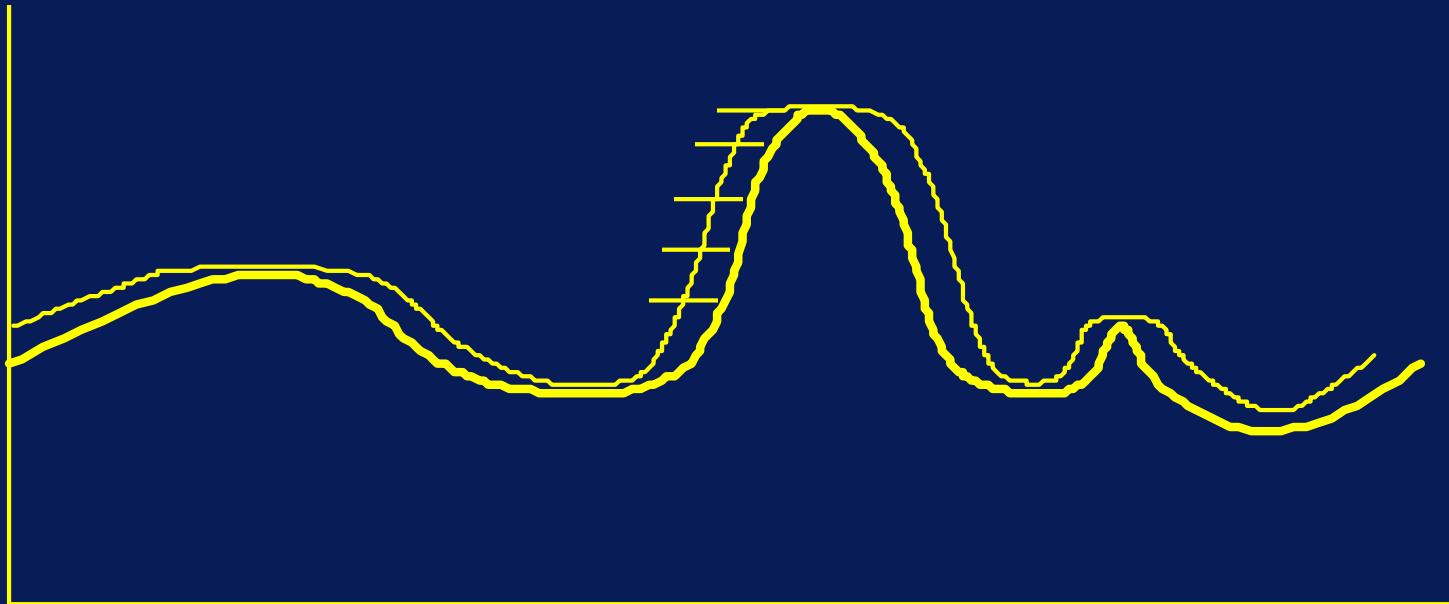


Erosion of a (1D) function with a flat structuring element; the eroded function is depicted by the thin curve.

Similarly, *Grey-Level Dilation* is defined :

$$f \rightarrow f \oplus \overset{\cup}{B}$$

$$\{x_\lambda(f)\} \rightarrow \left\{x_\lambda(f) \oplus \overset{\cup}{B}\right\}$$



Dilation of a (1D) function with a flat structuring element; the dilated function is depicted by the thin curve.

The Segmentation Problem

- *Introduction :*
- Region-based and Boundary-based approaches.

- ***Segmentation :***

- A grouping process in which the components of a group are similar with respect to some feature or set of features.

- The inference is that this grouping will identify regions in the image which correspond to unique and distinct objects in the visual environment.

- Two complementary approaches :
 1. Boundary Detection
 2. Region Growing

- Region-growing effects the segmentation process by :
 - grouping elemental areas (in simple cases, individual image pixels)
 - sharing a common feature
 - into connected two-dimensional areas called ***Regions.***

- Such features might be pixel grey-level or some elementary textural pattern, *e.g.* the short thin bars present in a herring-bone texture.

- Boundary-based segmentation is concerned with
 - detecting or enhancing the boundary pixels of objects within the image
 - and subsequently isolating them from the rest of the image.

- The boundary of the object, once extracted, may easily be used to define the location and shape of the object, effectively completing the isolation.

- An image comprising boundaries alone
 - is a much higher level representation of the scene than is the alone grey-scale image
 - in that it represents important information explicitly.

- In cases where the boundary shape is complicated, the gap between these two representations is wide
 - it may be necessary to introduce an intermediate representation which is independent of the shape.

- Boundaries of objects are often manifested as intensity discontinuities
 - a natural intermediate representation is composed of local object-independent discontinuities in image intensity, normally referred to as edges.

- an edge occurs in an image if some image attribute (normally image intensity) changes its value discontinuously.

- *Edges are seen as local intensity discontinuities while boundaries are global ones.*

- The usual approach to segmentation by boundary detection is :
 - to construct an edge image from the original grey-scale image
 - to use this edge to construct the boundary image without reference to the original grey-scale data by
 - edge linking to generate short curve segments.

- edge-thinning
 - gap-filling
 - curve segment linking
-
- Frequently with the use of domain-dependant knowledge.

- This association of local edges is normally referred to as ***Boundary Detection*** and ...
- The generation of the local edge image is referred to a ***Edge Detection***.

- Boundary detection algorithms
 - vary in the amount of domain-dependant information or knowledge which they incorporate in associating or linking the edges;
 - their effectiveness is obviously dependant on the quality of the edge image.

- The more reliable the edge elements in terms of their position, orientation and indeed, authenticity, the more effective the boundary detector will be.

- For relatively simple well-defined shapes, boundary detection may become redundant or at least trivial, as edge detection performance improves.

- Computational Complexity for the segmentation process depends on
 - the complexity of the edge detection,
 - the complexity of the boundary detection.

- Minimal segmentation complexity may be achieved by a trade-off between the sophistication of the edge detector and the boundary detector.

- Edge Detection is essentially a filtering process and can often be effected in hardware.
- Boundary Detection will require more sophisticated software.

- The current (and, probably, correct) trend is to deploy the most effective and sophisticated edge detector (*e.g.* the *Canny Operator* or the *Marr-Hildreth Operator*) and to simplify the boundary detection process.

Thresholding

- Grey-level thresholding is a simple region based technique.
- In situations where :
 - an object exhibits a uniform grey-level
 - and rests against a background of a different grey-level.

- Thresholding will assign
 - a value of 0 to all pixels with a grey-level less than the threshold level;
 - a value of 255 (say) to all pixels with a grey-level greater than the threshold level.

- Thus, the image is segmented into two disjoint regions, one corresponding to the background and the other to the object.

Global, Local and Dynamic Approaches

- A threshold operation may be viewed as a test involving :
 - some function of the grey-level at a point
 - some local property of the point, *e.g.* the average grey-level over some neighbourhood,
 - the position of the point in the image.

- Thus, a threshold operation may be viewed as a test involving a function T of the form :
 - $\mathbf{T}(x, y, N(x, y), g(x, y))$
- where $g(x, y)$ is the grey-level at the point (x, y) . $N(x, y)$ denotes some local property of the point (x, y) .

- If $g(x, y) > T(x, y, N(x, y), g(x, y))$ then
- (x, y) is labelled an object point, otherwise it is labelled as a background point, or conversely.

- Three classes of thresholding (global, local and dynamic) may be distinguished on the basis of restrictions placed on this function (see Weszka 1978).

- These are :
- $T = T(g(x, y))$ - *Global Thresholding* :
 - the test is dependant only on the grey-level of the point.

- $T = T(N(x, y), g(x, y))$ ***Local Thresholding:***
 - the test is dependant on a neighbourhood property of the point and on the grey-level of the point.

- $T = T(x, y, N(x, y), g(x, y))$
- *Dynamic Thresholding* :
 - the test is dependant on the point co-ordinates, a neighbourhood property of the point and on the grey-level of the point.

- Most systems utilise the simplest of these three approaches, global thresholding :
 - the threshold test is based exclusively on the global threshold value and on the grey-level of a test-point
- irrespective of its position in the image or of any local context.

- The approach is facilitated either :
 - by constraining the scene to ensure that there is no uneven illumination or
 - by photometrically decalibrating the image before thresholding.

- The advantage of this approach is that the thresholding can be accomplished by commonly-available hardware using a look-up table.

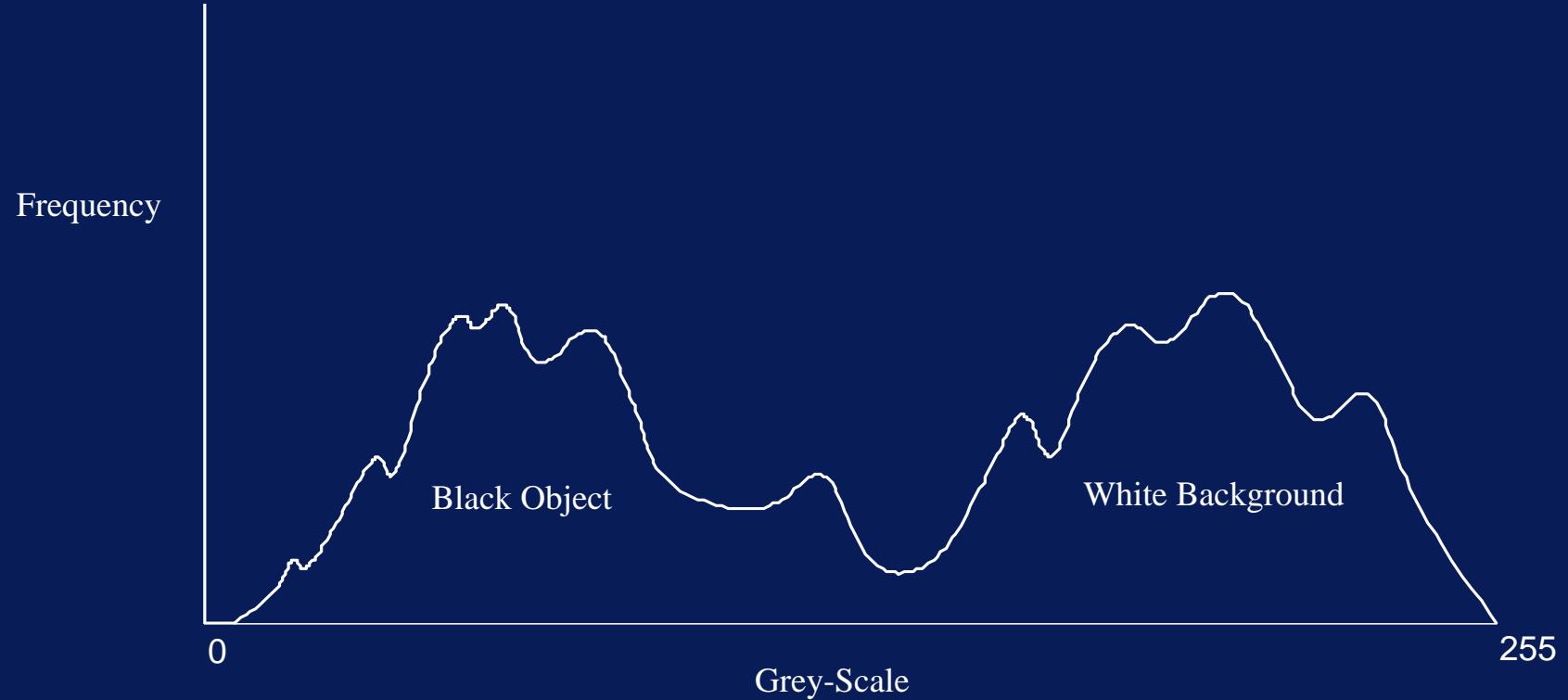
Threshold Selection

- The selection of an appropriate threshold is the single major problem for reliable segmentation.
- Most techniques are based on the analysis of the grey-level histogram, selecting thresholds which lie in the region between the two modes of the (bi-modal) histogram.

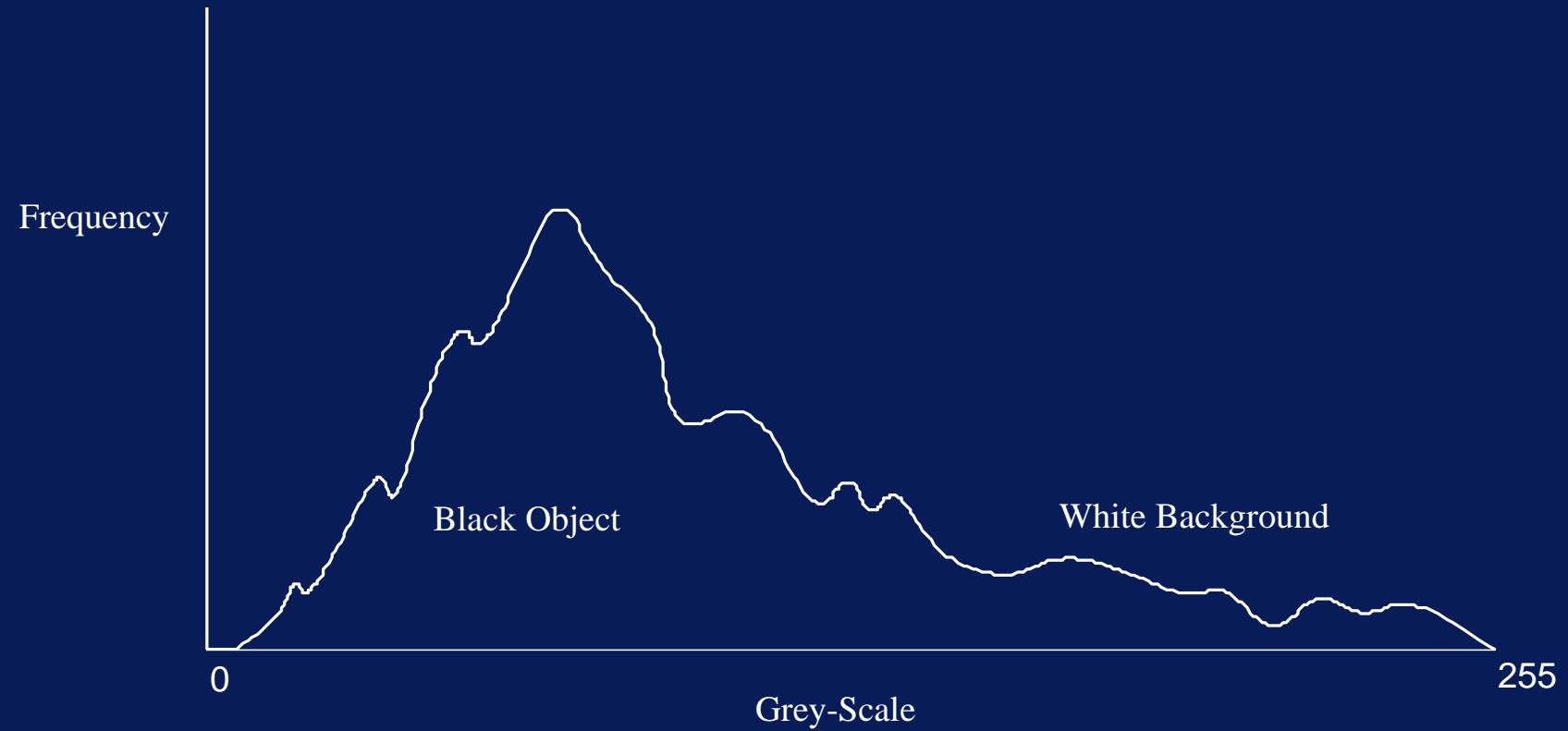
- The assumption that the histogram is indeed bi-modal,
 - with one mode corresponding to the grey-level representing the object
 - the other to the grey-level representing the background
- is not often valid.

- Histograms are frequently noisy and the two modes may be difficult to detect.
- Just as often, the object will generate a single mode while the background will comprise a wide range of grey-levels, giving rise to a uni-modal histogram.

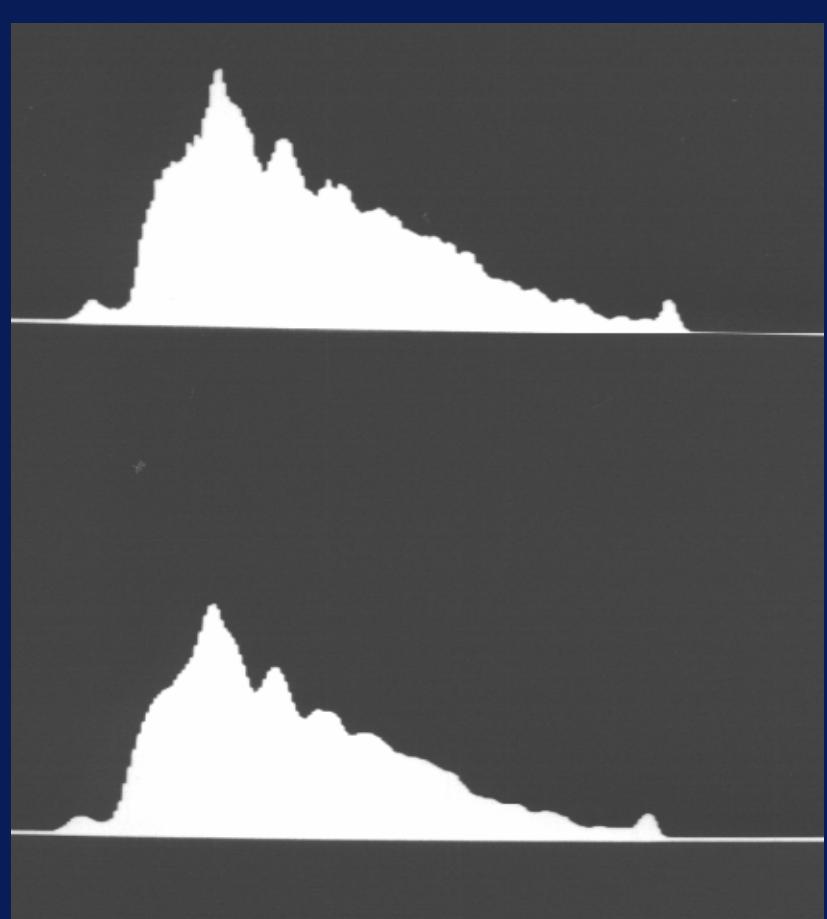
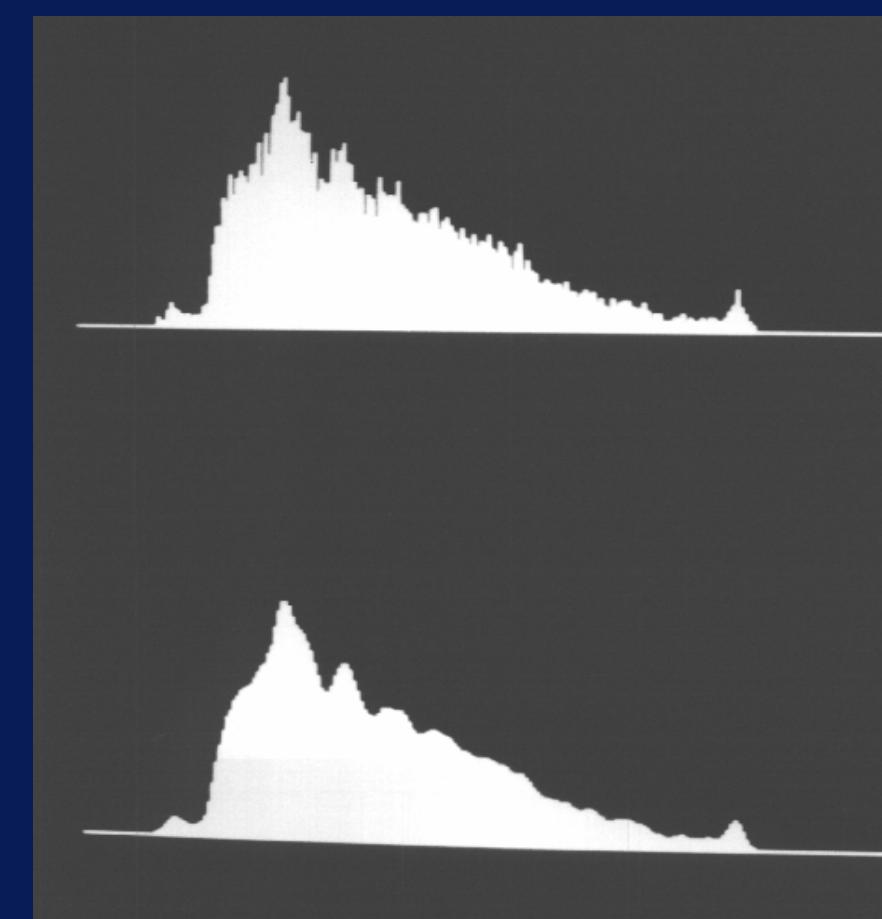
- While several applications of a simple smoothing (or local averaging) operator to a noisy histogram will help with noisy bi-modal histograms, it will be of little use with uni-modal histograms.



- *Noisy Bi-Modal Grey-Scale Histogram*



- *Uni-Modal Grey-Scale Histogram*



- *Grey-scale histogram smoothing*

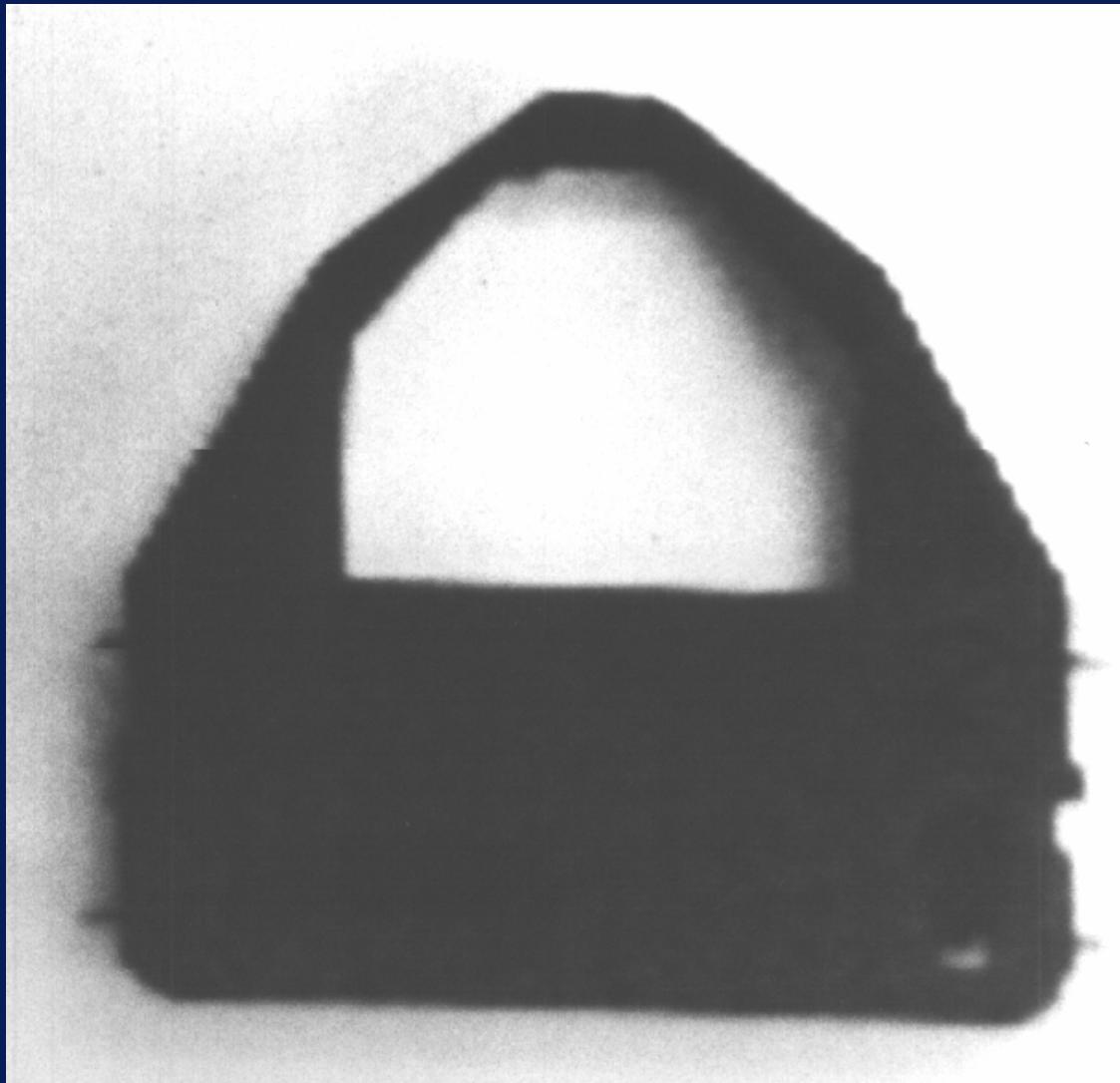
Top-left: no smoothing; top-right: one application of a 3x1 neighbourhood average operator; Bottom-left: two applications; bottom-right: three applications.

- Another useful approach to thresholding selection is to use the average grey-level of those pixels which are on the boundary between the object and the background as an estimate of the threshold value.

- As the grey-level of this boundary pixel will typically lie between those of the object and the background, they provide a good indication of the threshold value.
- The difficulty, of course, lies in deciding which pixels *are* on the boundary.

- A threshold selection procedure
 - uses a Marr-Hildreth operator locate edges in the image;
 - the mean grey-level of the image pixels at these edge locations is computed;
 - this mean represents the global threshold value.

- *Although this threshold selection technique is computationally complex and may take a significant amount of time to compute, it is only a calibration exercise and need not be performed before every threshold operation.*



- (a) *Original Grey-Scale Image*



- (b) *Automatically thresholded binary image*

An Overview of Edge Detection Techniques

- There are four distinct approaches to the problem of edge detection :
 1. Gradient and Difference based Operators
 2. Template Matching
 3. Edge Fitting
 4. Statistical Edge Detection.

Gradient and Difference Operators

- Define a local edge in an image to be a transition between two regions of significantly different intensities.

- The gradient function of the image
 - which measures the rate of change,
 - will have large values in these transitional boundary areas.

- Gradient-based, or first-derivative based, edge detectors
 - enhance the image by estimating its gradient function and then
 - signal that an edge is present if the gradient value is greater than some defined threshold.

- In more detail, if $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ represent the rates of change of a 2D function $f(x, y)$ in the x and y directions respectively, then the rate of change in a direction θ (measured in the positive sense from the X -axis) is given by:

$$\frac{\partial f}{\partial x} \cos \theta + \frac{\partial f}{\partial y} \sin \theta$$

- The direction θ at which the rate of change has the greatest magnitude is given by :

$$\arctan \left[\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right]$$

- The gradient of $f(x, y)$ is a vector at (x, y) with this magnitude and direction.
- Thus the gradient may be estimated if the directional derivatives of the function are known along (any) two orthogonal directions ...

- with magnitude

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

- The essential differences between all gradient edge detectors are
 - the directions which the operators use,
 - the manner in which they approximate the one-dimensional derivatives of the image function in these directions,
 - the manner in which they combine these approximations to form the gradient magnitude.

- These gradient functions are intuitively easy to understand when we confine ourselves to the discrete domain of digital images where partial derivatives become simple first differences.

- For example, the first difference of a 2D function in the x -direction is simply
 - $f(x+1, y) - f(x, y)$
- Similarly, the first difference of a 2D function in the y -direction is simply
 - $f(x, y+1) - f(x, y)$

- An operator due to Roberts estimate the derivatives diagonally over a 2*2 neighbourhood.

- The magnitude of the gradient, $g(x, y)$, at an image point, (x, y) , is approximated by taking the RMS of the directional derivatives :

$$g(x, y) \approx R(x, y) =$$

•

$$\sqrt{\{f(x, y) - f(x + 1, y + 1)\}^2 + \{f(x, y + 1) - f(x + 1, y)\}^2}$$

- $R(x, y)$ is usually referred to as the *Roberts Cross Operator*.

- The differences may be combined in a way other than the RMS to provide a computationally simpler version, the ***Roberts Absolute Value*** estimate of the gradient function, given by :

$$g(x, y) \approx R(x, y) = |f(x, y) - f(x + 1, y + 1)| + |f(x, y + 1) - f(x + 1, y)|$$

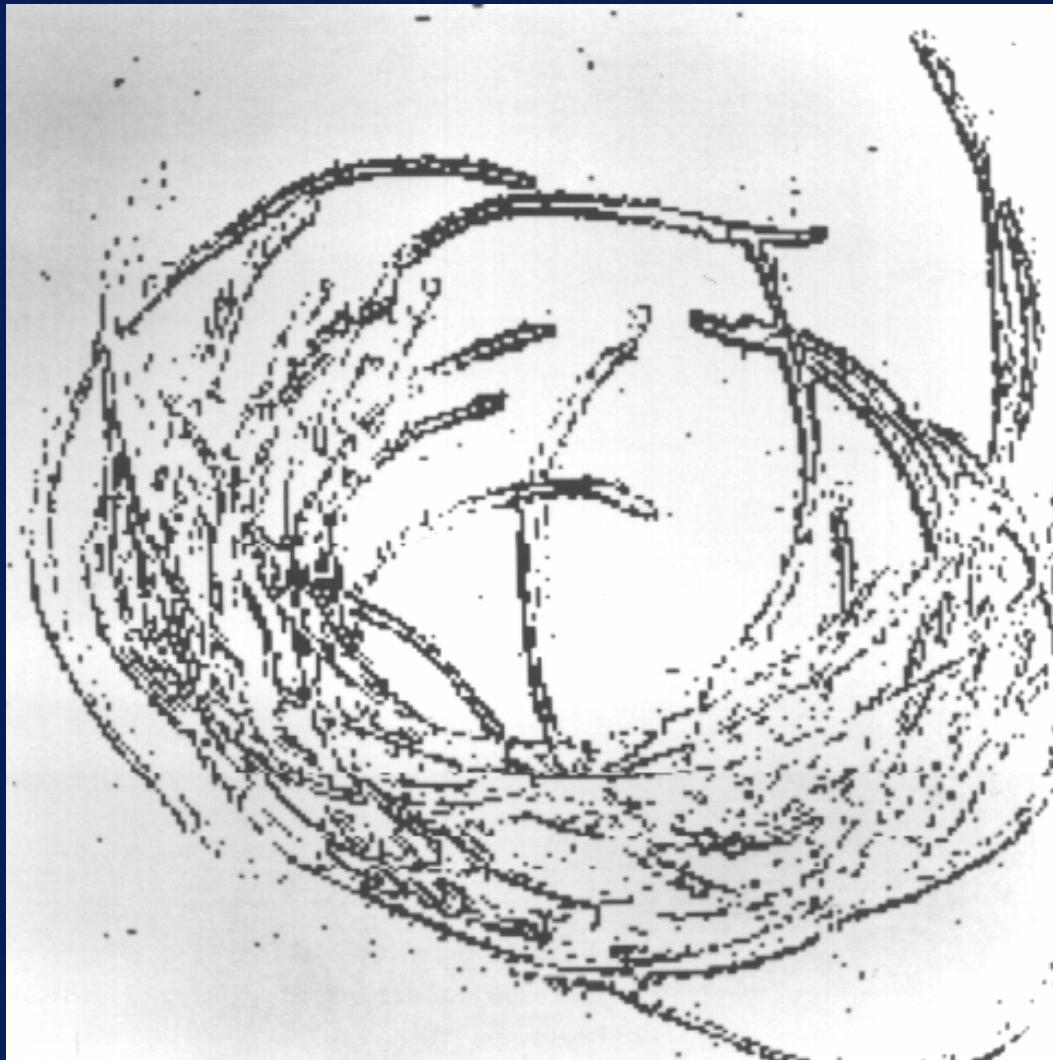
- Rosenfeld and Kak have argued that a third version, the *Max Roberts Operator*, given by :

$$g(x, y) \approx R(x, y) = \\ \text{Max}(|f(x, y) - f(x + 1, y + 1)| + |f(x, y + 1) - f(x + 1, y)|)$$

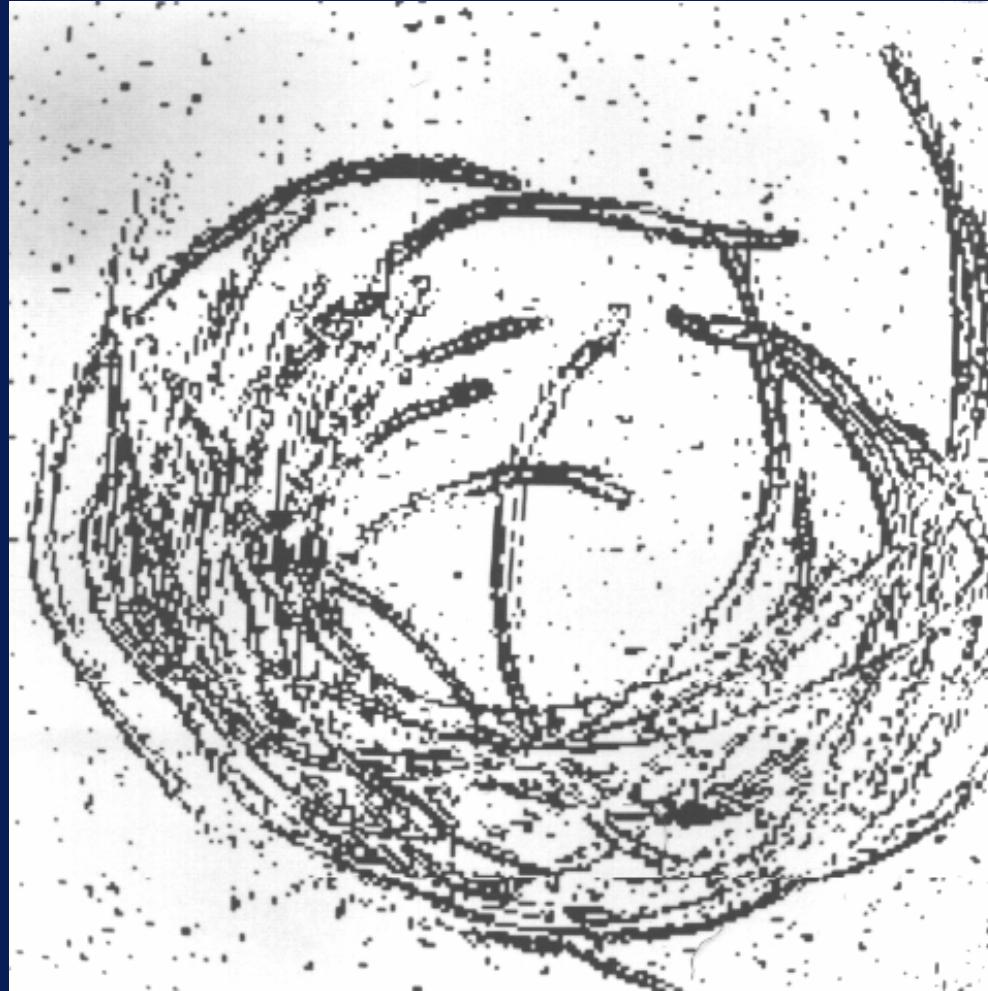
- ... affords better invariance to edge orientation. Applying the Max operator to edges of equal strength, but of different orientation, produces less variation in the resultant magnitude value than if the cross operator were used.



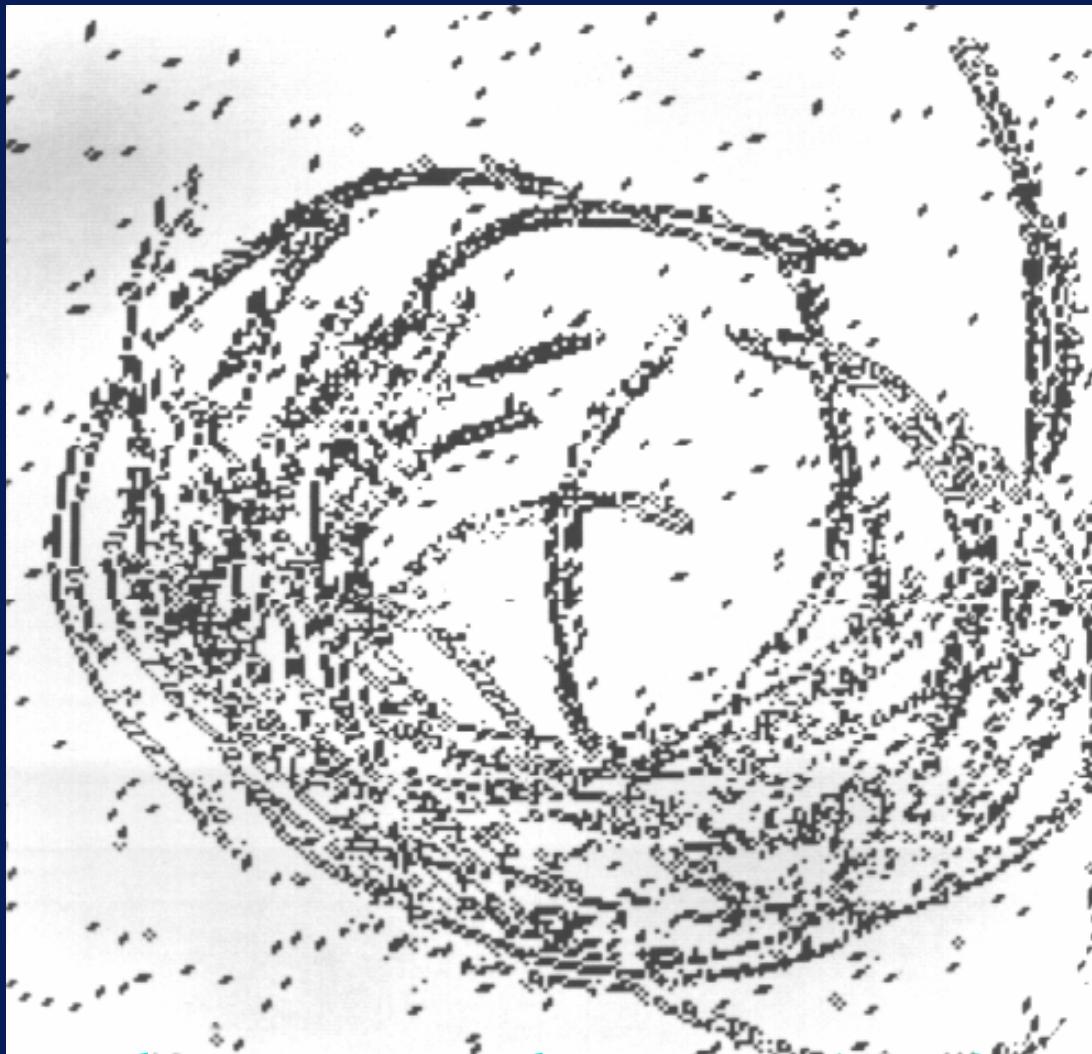
- *A Tray of Wires*



- *Roberts KMS Edge Detection Operator*



- *Roberts Absolute Value*
- *Edge Detection Operator*



- *Roberts Max Edge Detection Operator*

- One of the main problems with the Roberts Operator is its susceptibility to noise because of the manner in which it estimates the directional derivatives, *i.e.*, the first differences, of the image function $f(x, y)$.

- This has prompted the estimation of the gradient by combining the differencing process with local averaging.

- For example, the *Sobel Operator* estimates the partial derivative in the x direction over a 3*3 region centred at $f(x, y)$ by :

$$S_x = \left\{ f(x+1, y-1) + 2f(x+1, y) + f(x+1, y+1) \right\} \\ - \left\{ f(x-1, y-1) + 2f(x-1, y) + f(x-1, y+1) \right\}$$

- This essentially takes the difference of a weighted average of the image intensity on either side of $f(x, y)$. Similarly,

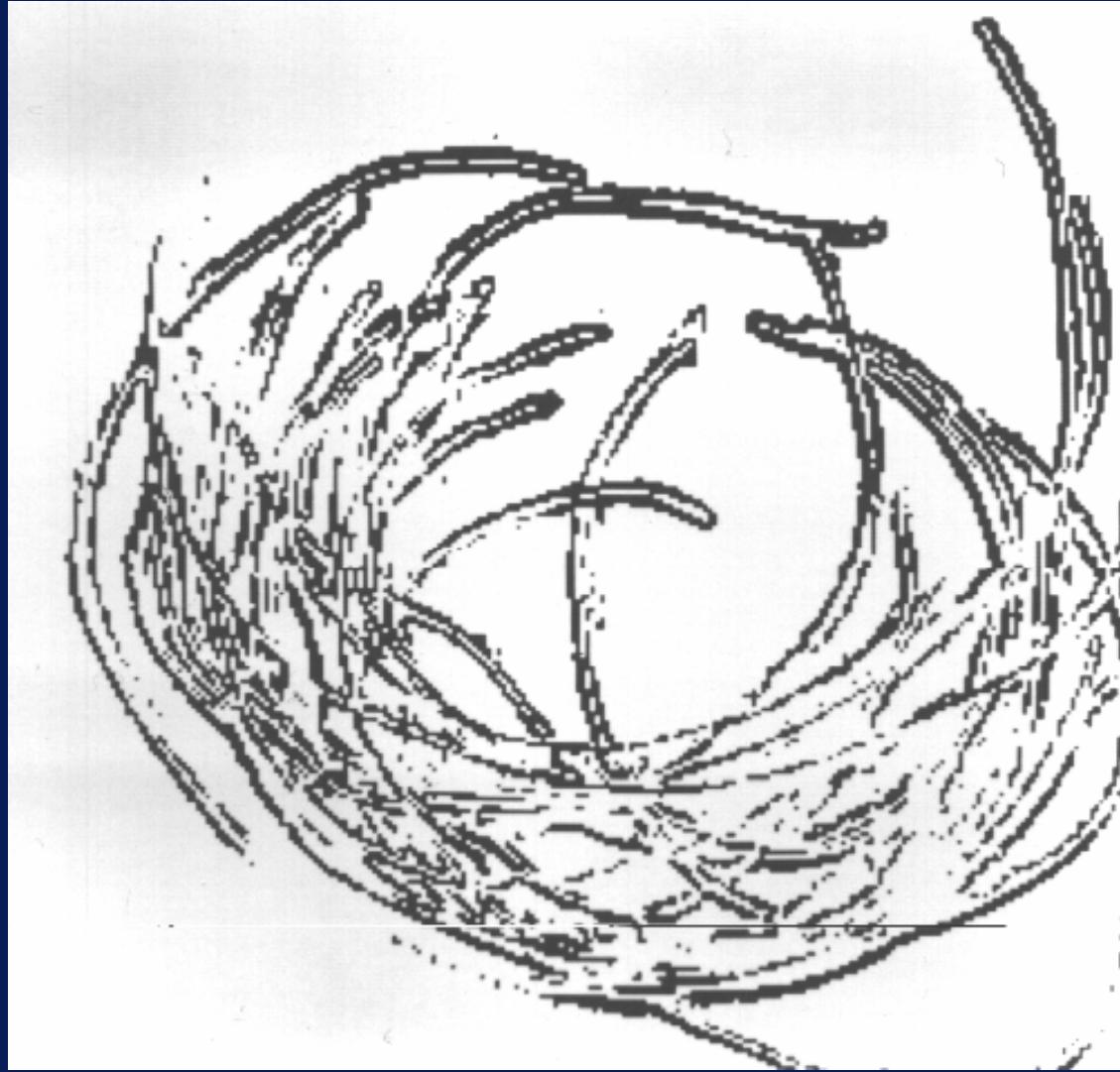
$$S_y = \left\{ f(x-1, y+1) + 2f(x, y+1) + f(x+1, y+1) \right\} \\ - \left\{ f(x-1, y-1) + 2f(x, y-1) + f(x+1, y-1) \right\}$$

- The gradient may then be estimated as before by either calculating the RMS (see figure 5.10) (*Slide 28*)

$$g(x, y) \approx S = \sqrt{S_x^2 + S_y^2}$$

- ... or by taking the absolute values (see Figure 5.11) (*Slide 29*)

$$g(x, y) \approx S = |S_x| + |S_y|$$



- *Sobel RMS Edge Detection Operator*

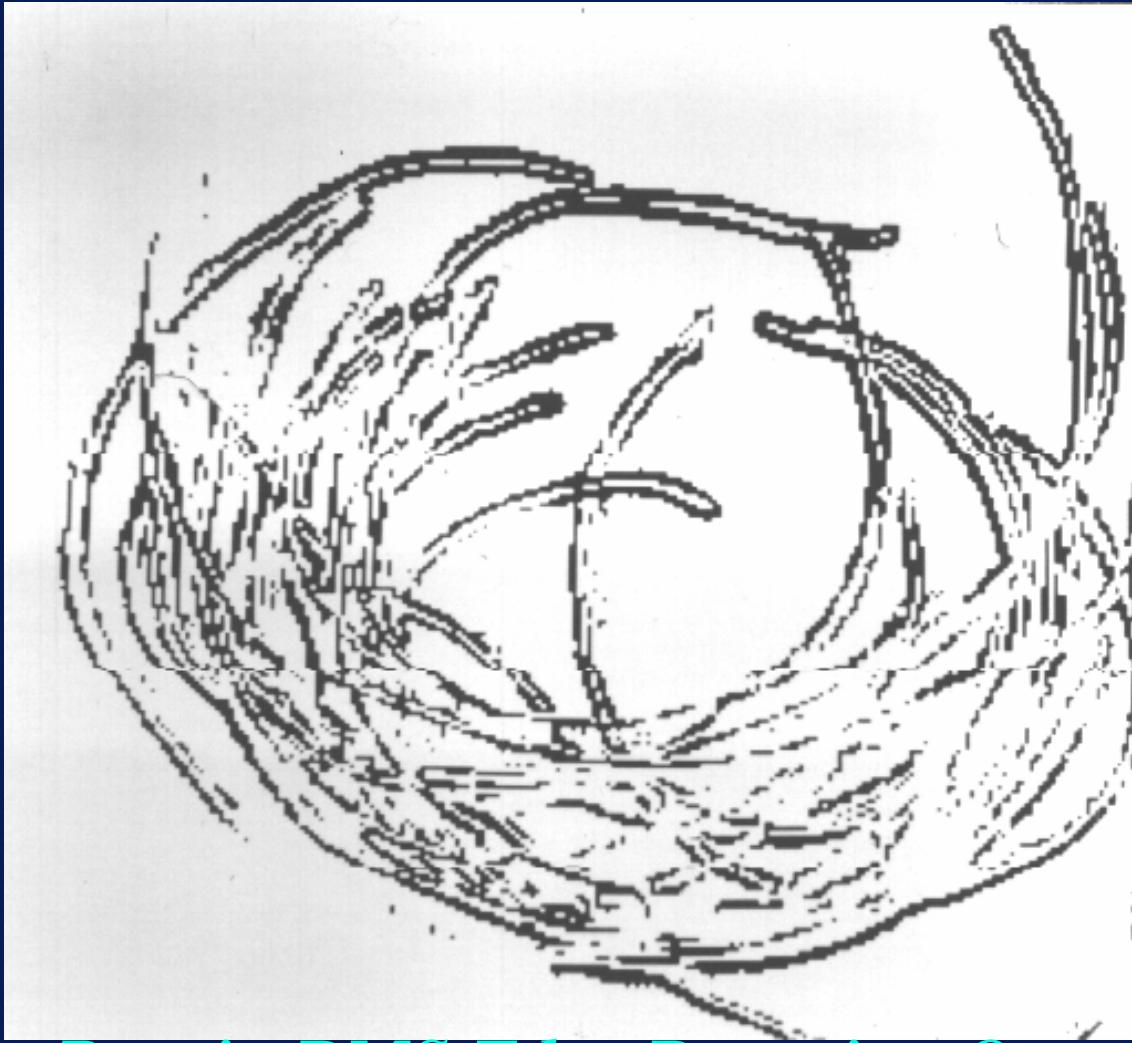


- *Sobel Absolute Value Edge Detection Operator*

- In an analogous manner Prewitt suggests an approximation of the partial derivatives by :

$$P_x = \left\{ f(x+1, y-1) + f(x+1, y) + f(x+1, y+1) \right\} \\ - \left\{ f(x-1, y-1) + f(x, y-1) + f(x+1, y-1) \right\}$$

- and the gradient may be estimated as before.



- *Prewitt RMS Edge Detection Operator*



- *Prewitt Absolute Value*
- *Edge Detection Operator*

- Quite often, the directional differences are estimated using simple convolution kernels, one kernel for each difference operator.

- These yield two partial derivative images:
 - which are then combined on a point by point basis.
 - either as the RMS or as the sum of absolute values
- to produce the final gradient estimate.

- Strictly speaking, the kernel should first be rotated by 180° before the convolution is performed (see section 4.2.1).
- However, this is normally omitted since the resultant error of 180° in the gradient direction can be ignored.

- (a) *Roberts*
- | | |
|---|----|
| 1 | 0 |
| 0 | -1 |
- (a)
-
- (b) *Sobel*
- | | | |
|----|----|----|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |
- | | | |
|----|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |
- (b)
-
- (c) *Prewitt*
- | | | |
|----|----|----|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
- | | | |
|----|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |
- (c)

- Once the gradient magnitude has been estimated, a decision as to whether or not an edge exists is made by comparing it to some pre-defined value;
 - an edge is deemed to be present if the magnitude is greater than this threshold.

- In noisy images threshold selection involves a trade-off between missing valid edges and including noise-induced false edges.

- So far, edge detection has been discussed on the basis of first derivative directional operators.
- However, an alternative method uses an approximation to the Laplacian :

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

- *i.e.* the sum of second-order, unmixed, partial derivatives.
- The standard approximation is given by:
- $L(x, y) =$
$$f(x, y) - 1/4 \{ f(x, y+1) + f(x, y-1) + f(x+1, y) + f(x-1, y) \}$$

- The digital Laplacian has :
 - zero-response to linear ramps (and thus gradual changes in intensity)
 - but it does respond on either side of the edge, once with a positive sign and once with a negative sign.

- To detect edges
 - the image is enhanced by evaluating the digital Laplacian
 - isolating the points at which the resultant image goes from positive to negative, *i.e.*, at which it crosses zero.

- The Laplacian has one significant disadvantage : it responds very strongly to noise.
- A different and much more successful application of the Laplacian to edge detection was proposed by Marr and Hildreth in 1980.

- This approach first smoothens the image by convolving it with a two-dimensional Gaussian function,
- and subsequently isolating the zero-crossings of the Laplacian of this image :

$$\nabla^2 \left\{ I(x, y) * G(x, y) \right\}$$

- where $I(x, y)$ represents the image intensity at a point (x, y) and $G(x, y)$ is the 2D Gaussian function, of a given standard deviation, defined :

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

- Despite some criticism of this technique, it is very widely used.
- The operator possesses a number of useful properties.

- For example : the evaluation of the Laplacian and the convolution commute so that, for a Gaussian with a given standard deviation, we can derive a single filter :
- *The Laplacian of Gaussian :*

$$\nabla^2 \{ I(x, y) * G(x, y) \} = \nabla^2 G(x, y) * I(x, y)$$

- Furthermore, this 2D convolution is separable into four 1D convolutions :

$$\nabla^2 \left\{ I(x, y) * G(x, y) \right\} = \\ G(x) * \left\{ I(x, y) * \frac{\partial^2}{\partial y^2} G(y) \right\} + G(y) * \left\{ I(x, y) * \frac{\partial^2}{\partial x^2} G(x) \right\}$$

- Bearing in mind that an implementation of the operator requires an extensive support, *e.g.* 63*63 pixels for a Gaussian with standard deviation of 9.0.

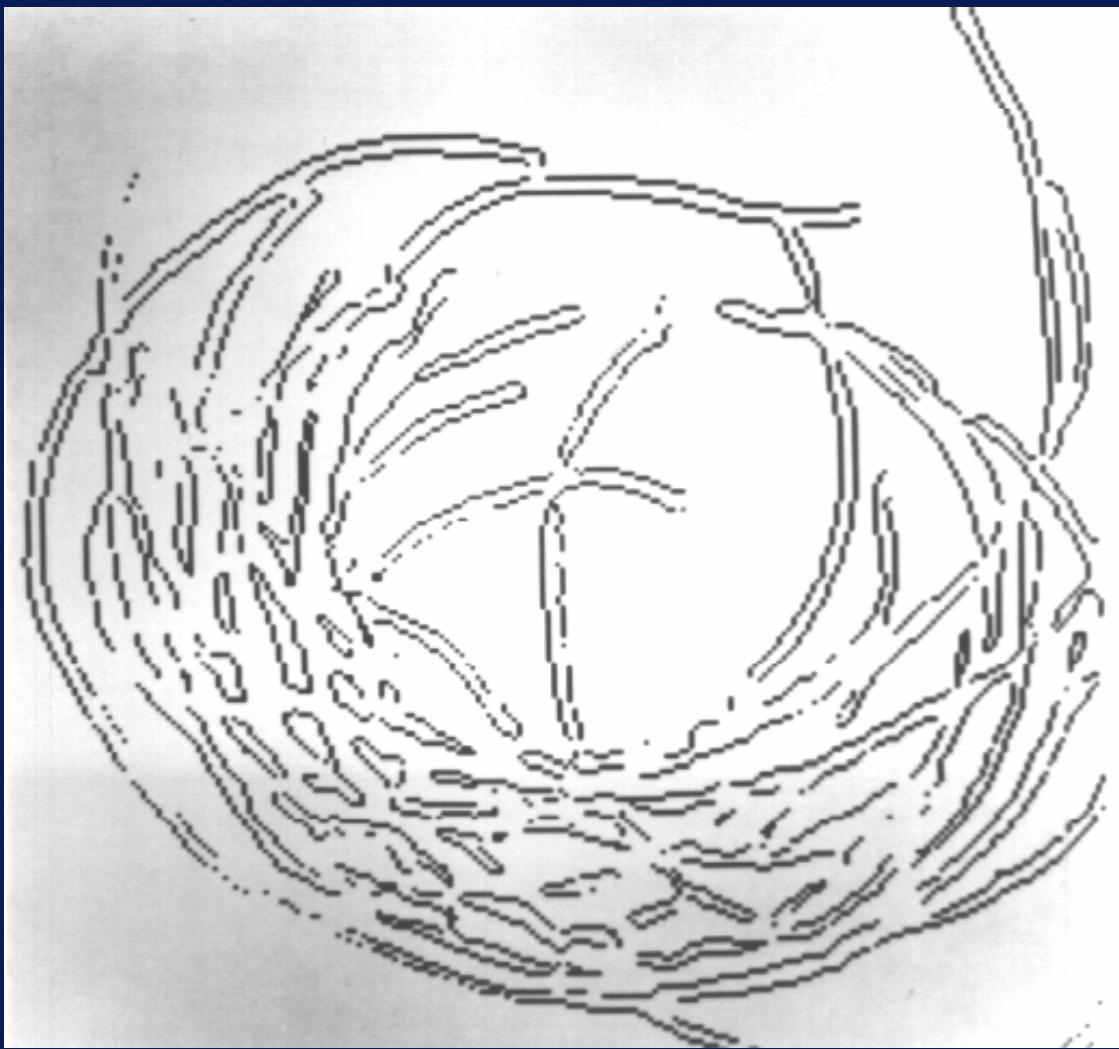
- This separability facilitates significant computational savings, reducing the required number of multiplications from n^2
- to $4n$ for a filter kernel size of n pixels.

- The Laplacian of Gaussian operator also yields thin continuous closed contours of zero-crossing points.
- This property is most useful in subsequent processing, such as when characterising the intensity discontinuities or edges as object boundaries.

- The slope of a zero-crossing is the rate at which the convolution output changes as it crosses zero and is related to the contrast and width of the intensity change.
 - Since the Gaussian is used to smooth the image

- *and*
 - Since different standard deviations yield edges detected at different scales within the image (successively smoothing out image detail),

- ... Marr's theory also requires the correlation of edge segments derived using Gaussians of different standard deviation.
- However, the edges detected by one operator alone are often sufficiently reliable for many industrial applications.



- *Marr-Hildreth Edge Detection Operator*

Template Matching

- Since an ideal edge is essentially a step-like pattern :
 - one straight-forward approach to edge detection is to try to match templates of these ideal step edges with regions of the same size at every point in the image.

** Cross correlation is discussed in Chapter 6 on Image Analysis*

- Several edge templates are used, each template representing an ideal step at a different orientation.
- The degree of match can, for example, be determined by evaluating the cross-correlation* between the template and the image.

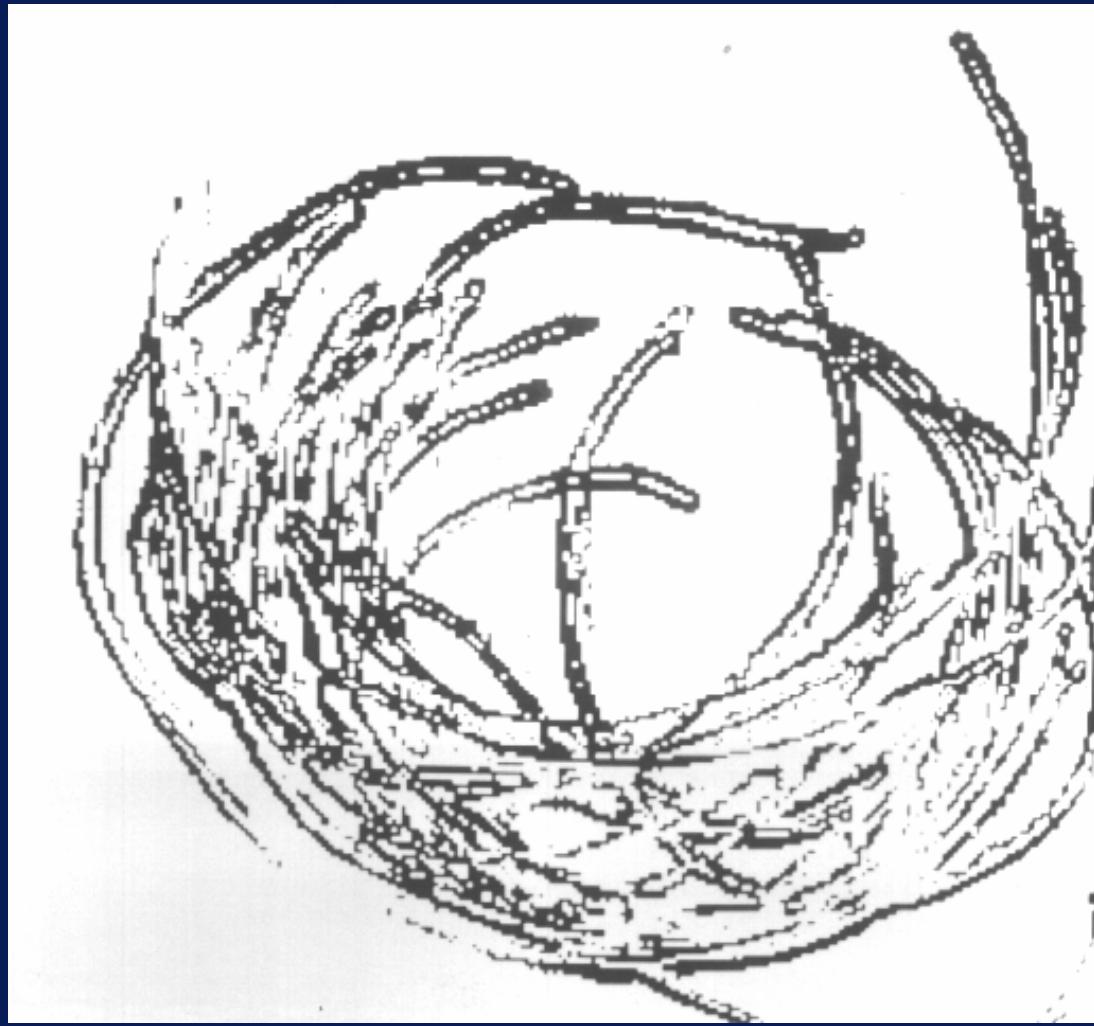
- The template producing the highest correlation determines the edge magnitude at that point and the edge orientation is assumed to be that of the corresponding template.
- Detection is accomplished by thresholding in the same manner as discussed for gradient approaches.

- These templates are often referred to as edge masks.

- Such a set of masks, due to Kirsch, is :

$$\begin{array}{cccc}
 \bullet & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 \\
 \bullet & 1 & -2 & 1 & -1 & -2 & 1 & -1 & -2 & 1 & -1 & -2 & 1 \\
 \bullet & -1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 \\
 \\
 \bullet & -1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\
 \bullet & 1 & -2 & 1 & 1 & -2 & -1 & 1 & -2 & -1 & 1 & -2 & -1 \\
 \bullet & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 & -1
 \end{array}$$

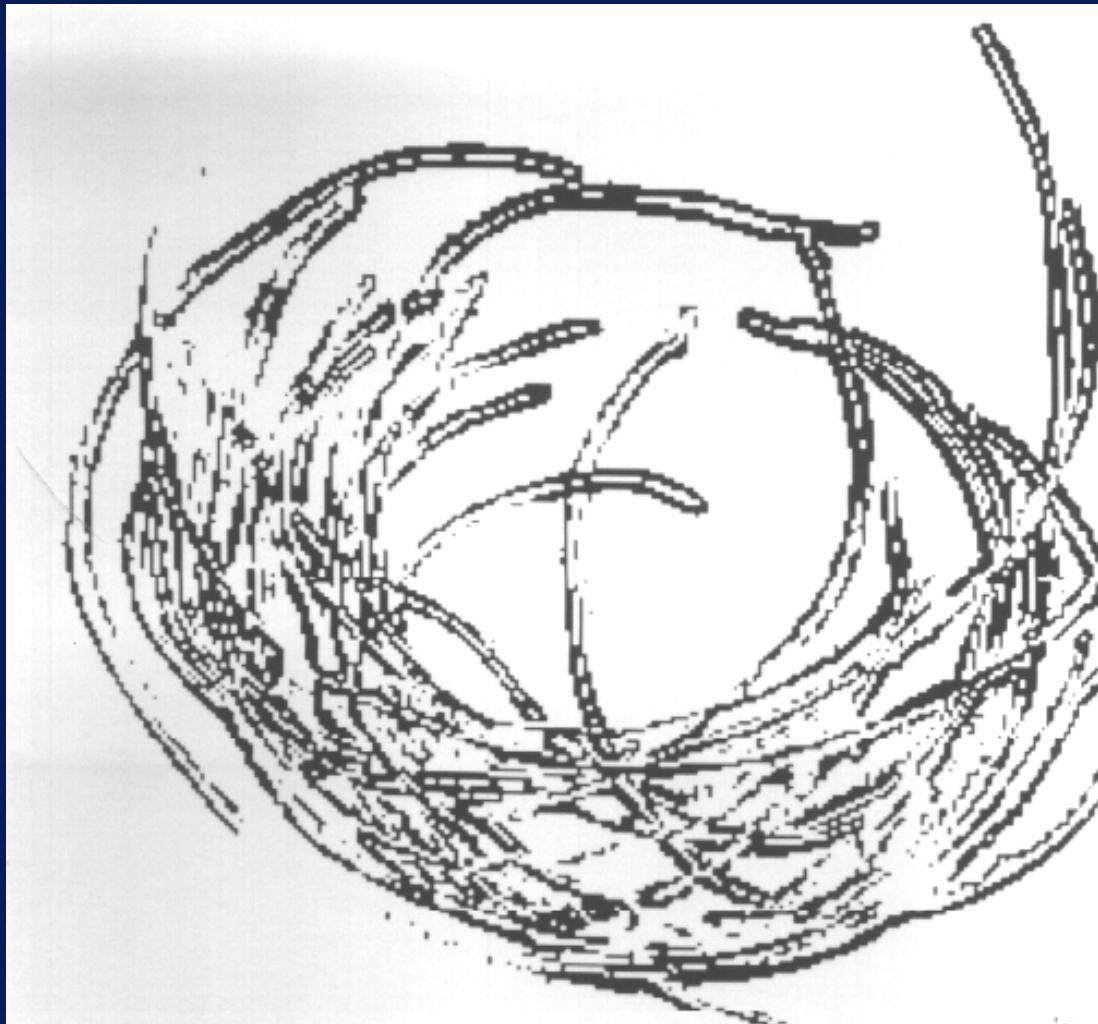
- Figure 5.16 (*Slide 61*) illustrates the effect of the application of this set of masks.



- *Kirsch Template Edge Detection Operator*

• Another set, due to Prewitt, is :

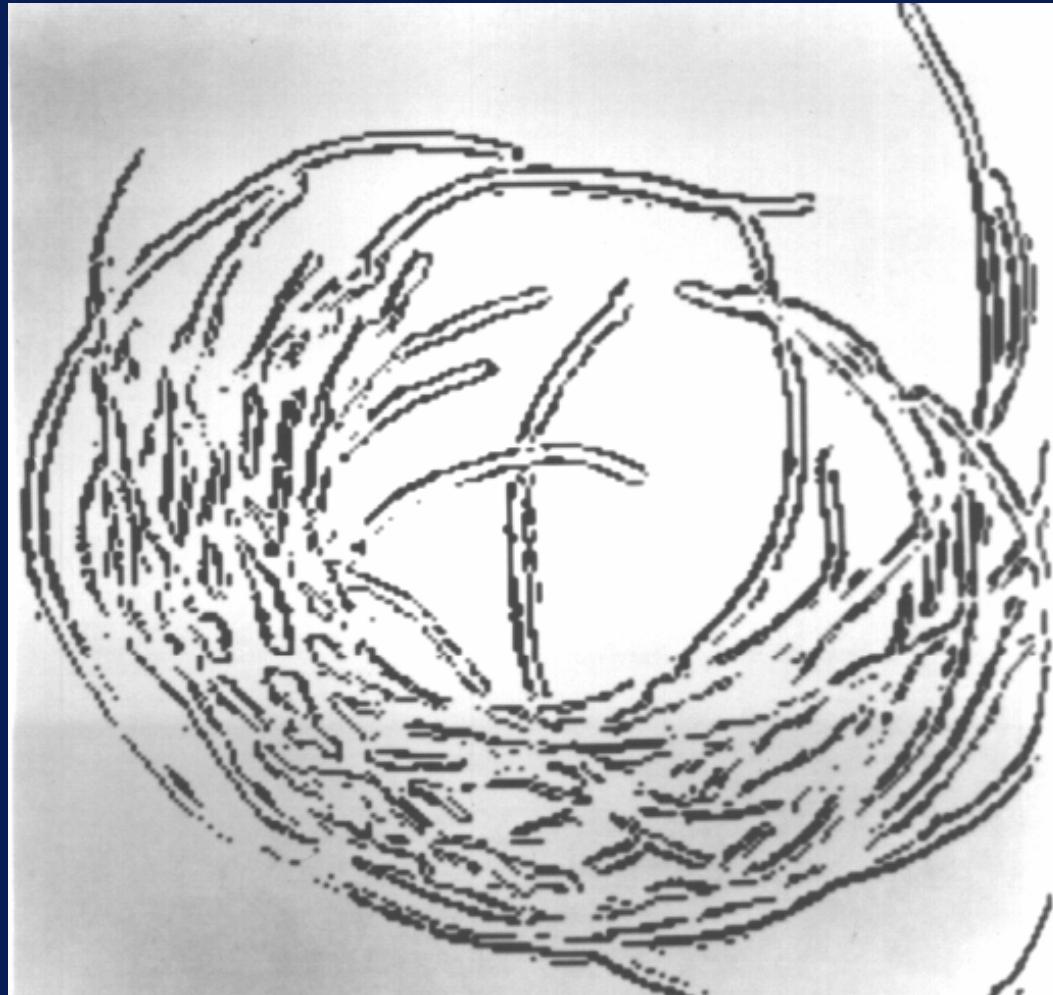
•	5	5	5	-3	5	5	-3	-3	5	-3	-3	-
3	-3	0	-3	-3	0	5	-3	0	5	-3	0	
5	-3	-3	-3	-3	-3	-3	-3	-3	5	-3	5	
•	-3	-3	-3	-3	-3	-3	-3	-3	5	-3	5	
5	-3	-3	-3	-3	-3	-3	-3	-3	5	-3	5	
•	-3	-3	-3	-3	-3	-3	5	-3	-3	5	5	-3
•	-3	0	-3	5	0	-3	5	0	-3	5	0	-3
•	5	5	5	5	5	-3	5	-3	-3	-3	-3	-
3												



- *Prewitt Template Edge Detection Operator*

- All the masks discussed so far, with the exception of Laplacian of Gaussian, when convolved with the image, will produce an enhanced image with large values, not only at the centre of the edge but also at points close to that edge.

- Subsequent thresholding of such an enhanced image will generate an edge map with thick edges.
- Nevatia and Babu suggested a template-matching algorithm (see Figure 5.17) (*Slide 66*) which produces thin edges.



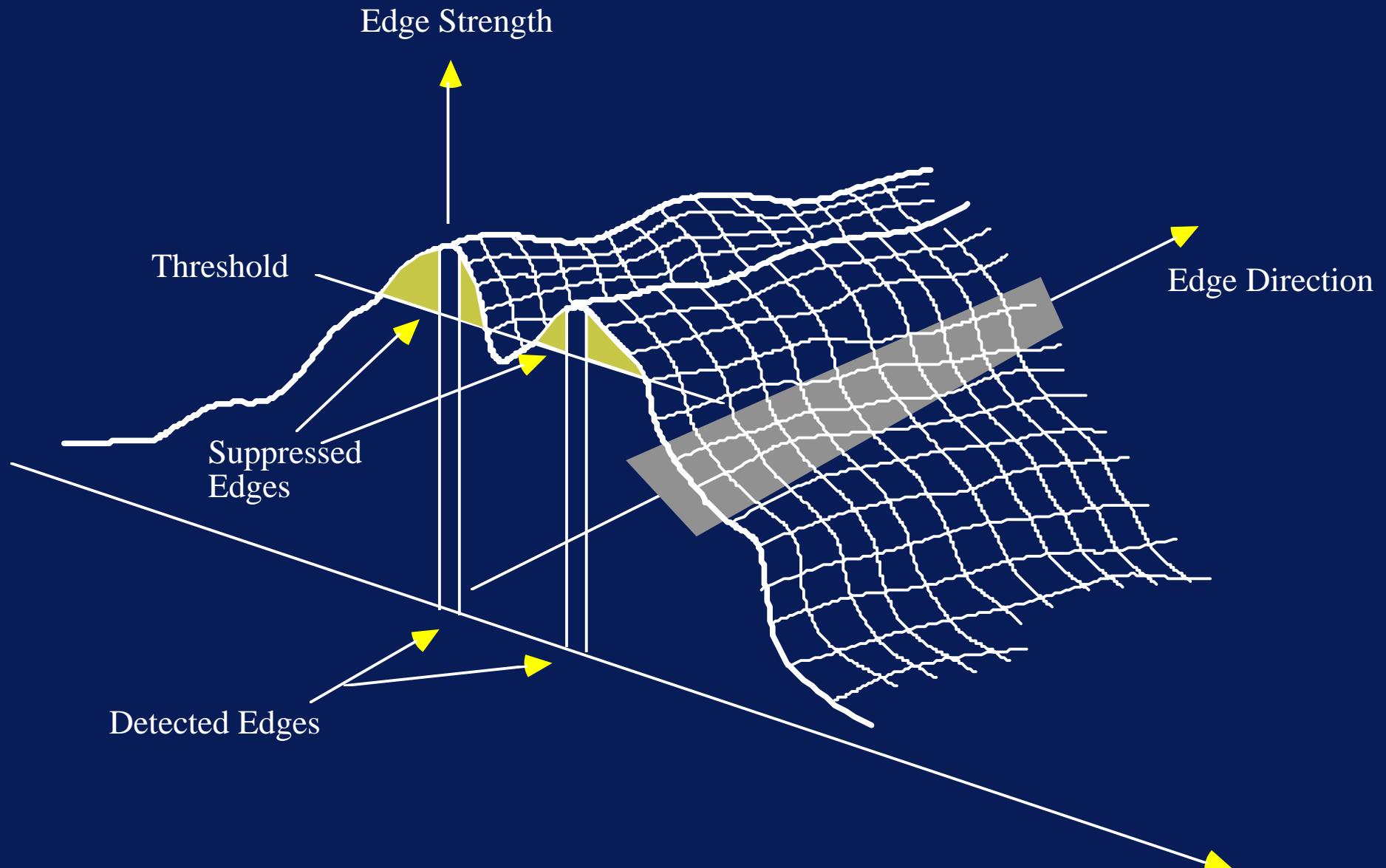
- *Nevatia-Babu Template*
- *Edge Detection Operator*

- In this case, six 5x5 masks (corresponding to edges at 0°, 30°, 60°, 90°, 120° and 150° orientations) are correlated with the image.

- An edge is deemed present at a particular orientation if
 - 1. The response at that orientation exceeds a set threshold and
 - 2. it is not dominated by responses at neighbouring points in a direction that is normal to the candidate edge.

- the edge magnitude must be thicker than the edge magnitude of the pixels on either side of it
- in a direction normal to the edge orientation;
- the neighbouring pixels are also required to have edge orientations similar to (within thirty degrees) that of the central point.

- This general technique is referred to as non-maxima suppression and was originally proposed by Rosenfeld and Thurston in 1971.

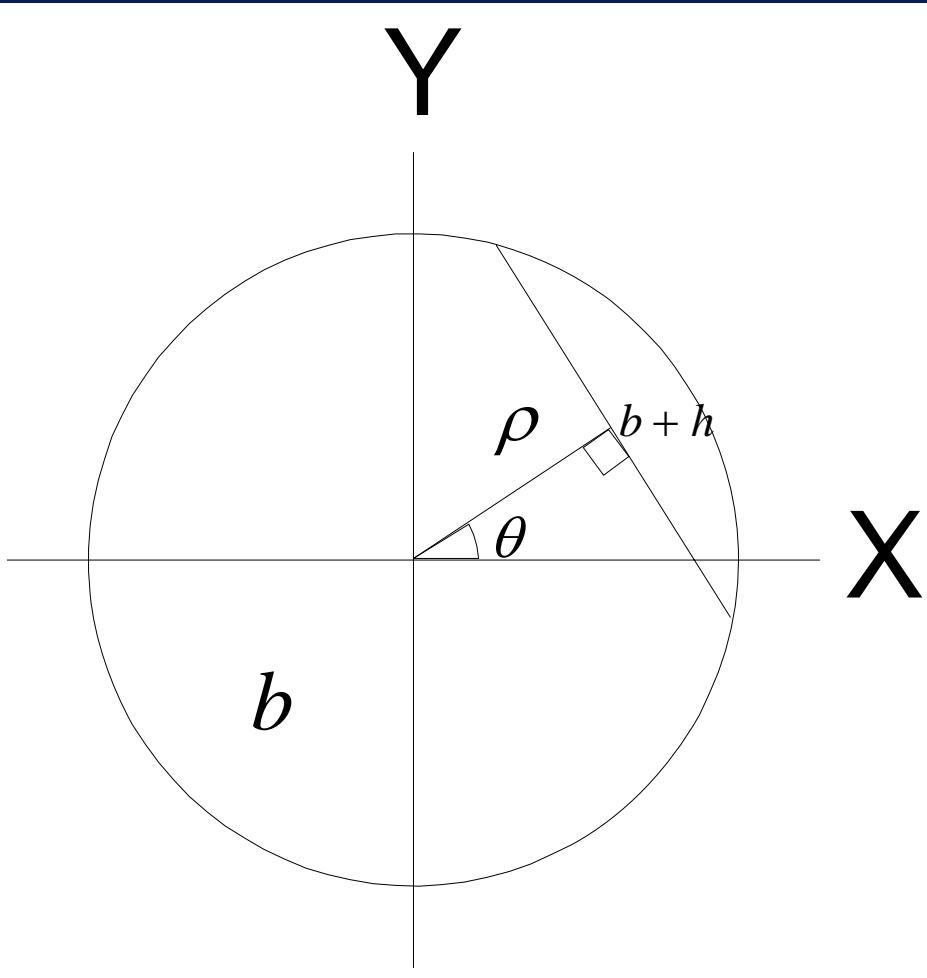


Edge Fitting

- As mentioned earlier, an ideal edge can be modelled as a step discontinuity in intensity at a particular location in the image.
- A step function can be defined in a circular region by a function $S(x, y)$ defined as follows (refer to Figure 5.20) (*Slide 3*) :

$$S(x, y) = \begin{cases} b & (x \cos \theta + y \cos \theta) < \rho \\ b + h & (x \cos \theta + y \sin \theta) \geq \rho \end{cases}$$

- where h is the step height (intensity difference), b is the base intensity, ρ and θ define the position and orientation of the edge line with respect to the origin of this circular region.



- *Parameters defining a step function*
 - *in a circular region*

- The approach to edge detection in this case is to determine how closely this model fits a given image neighbourhood
 - identify the values of b , h , ρ and θ that minimises some measure of the distance between this circular step function and a corresponding area in the image.
- This is the basis of *Hueckel's Operator*.

- The error, ε , between the ideal step $S(x, y, b, h, \rho, \theta)$, defined over a circular region C and (some) circular sub-image $f(x, y)$ of the same size can be given by :

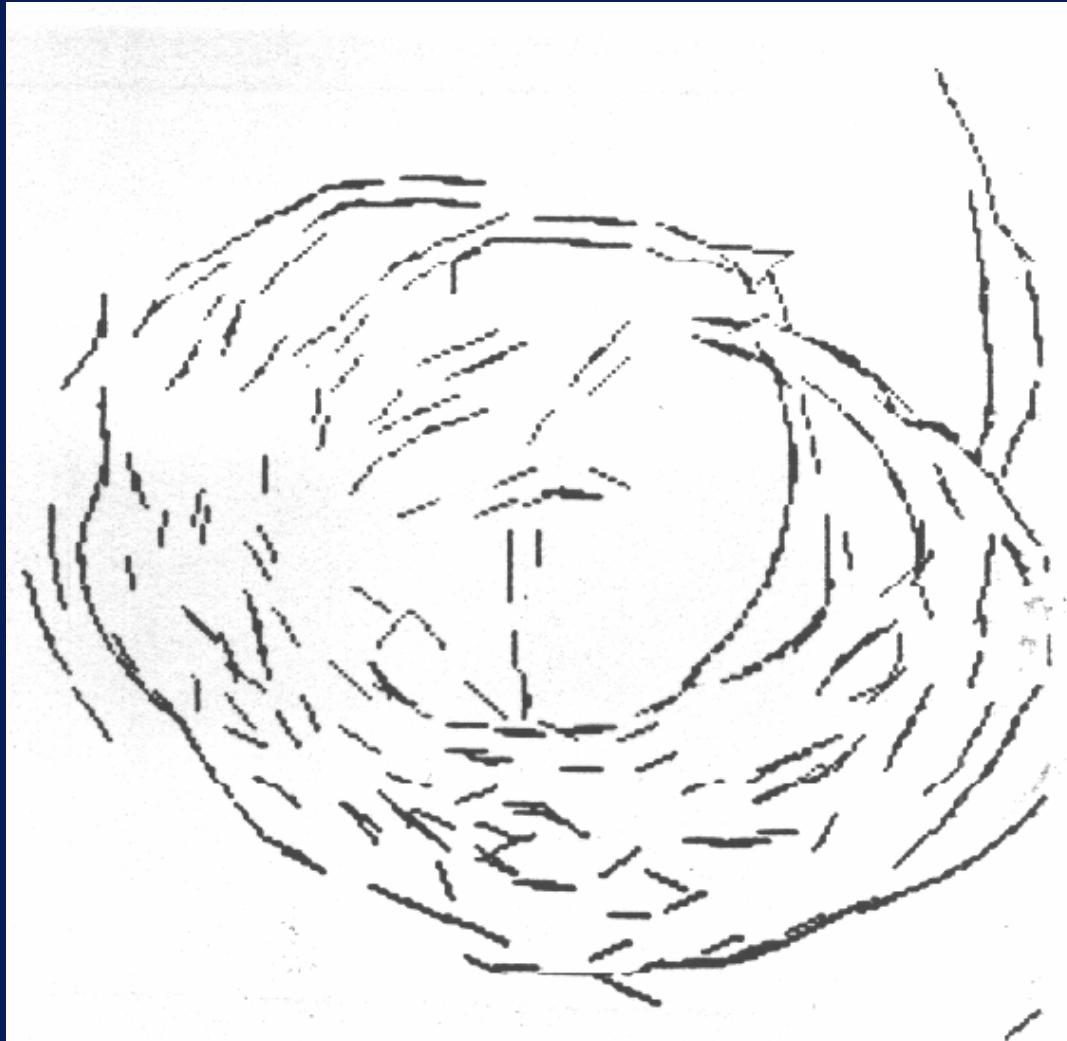
$$\varepsilon^2 = \sum_{x, y \in C} (f(x, y) - S(x, y, b, h, \rho, \theta))^2$$

- $f(x, y)$ is the image intensity at a point (x, y)
 - $S(x, y, b, h, \rho, \theta)$ is the ideal step edge for chosen values of b, h, r and θ at a point (x, y) .
- The summation is to be performed over the entire region C and b, h, ρ and q are to be chosen so that \hat{e}^2 minimised.

- The minimisation process is simplified by expanding both $f(x, y)$ and $S(x, y, b, h, \rho, \theta)$ in terms of a set of orthogonal (Fourier) basis functions
 - using the corresponding coefficients (f_i and S_i , say) in the error measure given above instead of $f(x, y)$ and $S(x, y, b, h, \rho, \theta)$

- In practice, only the first eight terms of the expansion are used.
- The operator was designed to detect the presence of two edge elements in the circular neighbourhood simultaneously, reporting only the more dominant edge.

- Each edge requires four parameters to describe it and it was for this reason that the eighth term was chosen as the cut-off.



- *Hueckel Edge Detection Operator*

- Thus the problem is not to minimise

$$\varepsilon^2 = \sum_{i=0}^7 (f_i - s_i)$$

- Hueckel then presents and proves a theorem which, in effect, reduces this problem to one of extremisation of a function in θ and he derives subsequent expressions for keywords and h .

- The decision as to whether an edge is present or not is based on the amplitude of the computed step as well as the degree of fit.

- Bearing in mind the computational complexity of the Hueckel edge fitting technique, the results are, in general, quite disappointing (see Figure 5.21) .

Statistical Edge Detectors

- If you consider a small region or window in an image (*e.g.* a 7*7 pixel area), you can view an edge as a boundary between two inhomogenous subregions.
- On the other hand, if the entire region is homogenous, then no edge exists.

- This is the basis of an edge detection technique due to Yakimovsky (see Figure 5.22) who treated edge detection as an exercise in hypothesis testing, choosing between the two hypotheses that :

- H0 : The image values on two sides of a line through the window are taken from the same region
- H1 : The image values on one side are from one region and on the other side from a different region.

- Assuming
 - each region comprises pixels having normally distributed grey-levels
 - that each pixel in the region is mutually independent.



- *Yakimovsky Edge Detection Operator*

- Yakimovsky argued that the choice can be made by considering the ratio of
 - (a) the grey level standard deviation of the combined region (raised to the power of the number of pixels in the region) to

- (b) the product of the grey-level standard deviations of both individual regions each with standard deviation raised to the power of the number of pixels in the respective region.

- To decide whether an edge of a given orientation exists in a window,
 - one can bisect the window at that orientation, thus creating two sub-regions,
 - calculate the appropriate standard deviations,
 - decide as to the presence of an edge based on some selected threshold.

- This may be done for several orientations, for example, 0° , 30° , 60° , 90° , 120° and 150° .
- There are several other statistical edge detectors but this one serves to illustrate the approach.

Assessment of Edge Detection

- It is difficult to quantitatively evaluate and compare edge detector performance.
 - There are a large number of detectors.
 - Each detector incorporates its own inherent edge model.

- Thus, certain edge detectors may be more appropriate or more successful in certain circumstances.
- Also, edge detectors will display differing degrees of noise, other will cater for (certain types) of noise but possibly at the cost of missing valid edges.

- A generic mathematical analysis of detectors is difficult, even for simple images, due to the frequent complexity and non-linearity of the detectors (but see [Hildreth 1986] and [Torre and Poggio 1986] for discussions of the topic).

- It is also worth noting that many of the detectors which can claim some degree optimality (*e.g.* Canny, Marr-Hildreth, Fleck) are optimal only with respect to the criteria they choose as relevant.
- To coin a phrase : “*You pick your optimality criteria and you take your choice*”

- In spite of these difficulties, a quantitative evaluation is clearly desirable even if only to provide a very rough guide to relative performance of detectors in limited circumstances.

- If a clear-cut objective analytic measure is difficult to establish (but, again, see [Torre and Poggio 1986]), an empirical assessment may often suffice.

- *It cannot be stressed too much, however, that empirical tests are only relevant in the application domain in which they are carried out.*

- One figure of measure that can be used in such tests was suggested by Pratt [Pratt 1978; Abdou and Pratt 1979] and provides a way of assessing the performance of edge detectors in localising the position of edges.

- Specifically, a figure of merit R , yielding a value in the interval 0.0 - 1.0, is defined :

$$R = \frac{1}{MAX(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha d^2}$$

- ... where I and A represent the number of ideal and actual edge map points, d is the separation distance of an actual edge point normal to a line of ideal edge points.

- The value α is a scaling constant and is adjusted to penalise edges offset from their true location. It is typically set at 0.111.

- Normalising the figure of merit by the maximum of the number of actual and ideal edge points ensures that fragmented and smeared edges are penalised.

Region Growing

- Region-growing effects the segmentation process
 - by grouping elemental areas which share a common feature into large connected two-dimensional areas called regions
 - with the implicit assumption that these resultant regions correspond to some real-world surface or object.

- Thus, the central idea underlying region-growing techniques is to merge initially small regions (*e.g.* individual pixels) into large ones.

- This merging process must then address two questions :
 - upon what criterion will regions be merged
 - at what point will merging cease ?

- To answer the first question, we must define more rigorously what we understand by the term *region*.

- A region is an aggregate (collection) of pixels all of which satisfy some *uniformity predicate* $U(p)$, such that the value of $U(p)$ is true if some local property of the neighbourhood of pixel p satisfies the uniformity predicate and false otherwise.

- Most region-growing techniques base the test for uniformity on some feature of the grey-level.
- The answer to the question : “what is the criteria for merging two regions?” is thus, quite simply, the uniformity predicate.

- Two adjacent regions are merged if the merged region satisfies the uniformity predicate.
- The answer to the second question: “at what point does merging cease?” is now clear.

- Merging (or growing) ceases when no two adjacent regions satisfy the uniformity predicate.

- Thus, the region growing procedure is an iterative one:
 - small regions are merged to form larger ones,
 - these are then merged (if appropriate) to form still larger ones,
 - and so on until no more regions can be merged.

- A natural consequence of this approach is that sophisticated data structures are required to keep track of the intermediate regions and their features.

- In the trivial global thresholding technique, the uniformity predicate is simply that the grey-level of the pixel (region) in question should lie within a pre-set range of grey-levels;

- The final regions are grown in one step as only one test is needed for each pixel and there are no intermediate regions.

The Split and Merge Technique using Quad-Trees

- This region-growing algorithm makes use of an image representation called the *Quad-tree*.
 - A quad-tree is a tree (a collection of nodes organised in a hierarchical manner) in which each node has either four sons or no sons.

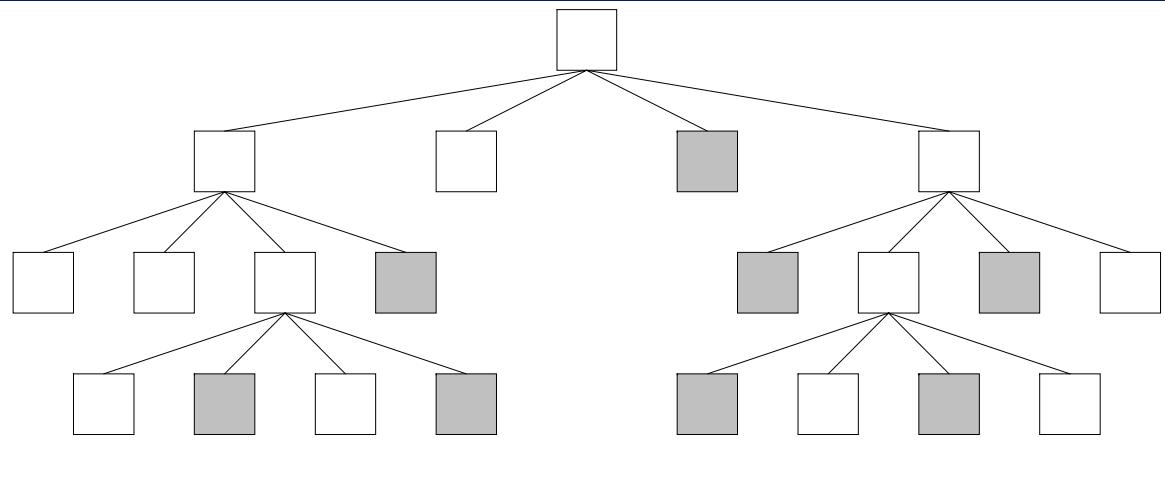
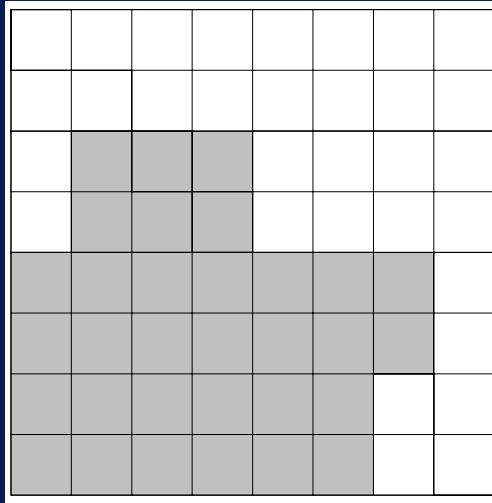
- The root of the quad-tree is a node which represents the average grey-level of the entire image.
- The leaves of a full quad-tree, *i.e.* the nodes which do not have any offspring, correspond to the individual pixels of the image.

- The parent of a group of four nodes corresponds to the average grey-level of a 2*2 neighbourhood of the image.
- If each of these four pixels/nodes have the same grey-level then there is no need to represent them explicitly as the parent node can do the job just as well and they can be deleted.

- Similarly, the parent of each of these nodes corresponds to a further $2*2$ grouping of these $2*2$ neighbourhoods;
 - again, the off-spring are present in the quad-tree if and only if they have different grey-levels.

- This hierarchical aggregation continues until one reaches the root of the tree which clearly represents the average grey-level of its sons, and hence the average grey-level of the entire image.

- Thus, one can see that the quad-tree is useful a method of image compression if there are large regions of uniform grey-level since much of these regions can be represented by nodes close to the root, without the need to have nodes for the individual pixels.



- *A Quad-Tree Representation*
 - *of an 8*8 Binary Image*

- The split and merge procedure begins with a quad-tree of a given depth.
- Typically where the depth is such that the leaves correspond not to pixels but to larger blocks.

- Leaves are then merged: if four leaves from a parent satisfy the uniformity predicate then the leaves are deleted and the parent inherits their average grey-level (assuming that we are going to use grey-level as a way of labelling a region).

- This merging continues at the next highest level, and then at subsequent levels, until no more merging is possible.

- At this stage a splitting is initiated. Here, the quad-tree is traversed and at each leaf the uniformity predicate of the corresponding group is evaluated;
 - if the value is false then the block is split and the four corresponding nodes are added to the node.

- This process continues until all leaves satisfy the uniformity predicate.

- At this stage, the quad-tree is traversed horizontally.
- The neighbours of each leaf are examined and adjacent leaves from different parents which satisfy the uniformity predicate are linked together to form a region.

- It is worth noting that, for industrial applications, region-growing techniques are considered too computationally complex (and hence too slow) to be of much use and should only be considered when thresholding or edge detection techniques are incapable of yielding satisfactory results.

Boundary Detection

- Edge Detection is only the first stage of the boundary-based segmentation process.

- We need also to aggregate these local edge elements :
 - which are a relatively featureless representation
 - into structures better suited to the process of interpretation.

- This is normally achieved using processes such as :
 - edge thinning (recall that the gradient-based and template matching edge detectors produce thick edges)
 - edge linking
 - gap filling
 - curve-segment linking

- In order to generate a distinct, explicit and unambiguous representation of the boundary.
- There are several techniques for boundary detection and they vary in the amount of knowledge or domain-dependant information that is used in the grouping process.

- These approaches include, in order of decreasing use of domain-dependant information :
 - Boundary Refining
 - The Hough Transform
 - Graph Searching
 - Dynamic Programming
 - Contour Following

Boundary Refining

- Boundary refining methods
 - use an initial *a priori* estimate of the position of the boundary to guide a search for the actual, or real, boundary
 - and subsequently to refine this initial estimate.

- The estimate may have been generated by the analysis of lower resolution images or, alternatively, explicit high-level knowledge that the boundary lies on, or is close to, a given curve.

- For example, Bolles incorporated this technique in an algorithm which
 - builds boundaries by searching in a direction at right angles to the a priori boundary for local edges and ...

- selects the element with the highest gradient value, providing its direction is (almost) parallel to the boundary direction [Bolles 1977].

- This is repeated at regular intervals along the approximate a priori boundary and if sufficient edge elements are extracted, then these locations are fitted with some analytic function, *e.g.* a low-degree polynomial.

- This parameterised curve then represents the actual boundary.

- Another approach, referred to as *Divide and Conquer* or *Iterative End-Point Fit* boundary detection
 - is often applicable where a low curvature boundary is known to exist between two edge elements.

- The general idea here is to fit an initial line between the two points ;
 - if the normal distance from the line to the point of maximum gradient magnitude is less than some pre-set tolerance,
 - the approximation is complete,

- otherwise the point of greatest normal distance from the line becomes a break-point on the boundary, forming two new line segments.
- All new segments are then subjected, in an iterative manner, to this same process.

Graph-Theoretic Techniques

- *A second approach to boundary detection, which need not depend significantly on the existence of a priori knowledge, treats the aggregation or association of edge points into boundaries as a (low cost) graph traversal.*

- In particular, edge points are viewed as nodes in a graph and there is a cost function or weight associated with connecting two neighbouring points or nodes
 - the desired boundary may be interpreted as the minimal cost (or any low-cost) path through the graph.

- The cost function or weight associated with connecting two edge points is normally defined to be a function of
 - the distance between them,
 - the difference in their directions,
 - and edge strength.

Dynamic Programming

- This approach formulates the boundary-following procedure as a dynamic programming problem by defining a cost function which embodies a notion of the “best boundary”.

- This is not dissimilar to the idea of the graph-theoretic technique in that the path specified by the boundary minimises the cost function.

- For example, suppose that a local edge detection operator is applied to a grey-level image to produce edge magnitude and direction information at points

$\vec{x}_1, \dots, \vec{x}_n \quad \cdot$

- One possible criterion for a good boundary is a weighted sum of high cumulative edge strength and low cumulative curvature; that is for a curve with n segments :

$$H(x_1, \dots, x_n) = \sum_{k=1}^n s(x_k) + c \sum_{k=1}^{n-1} q(x_k, x_{k+1})$$

- ... with the implicit constraint that consecutive points must be grid neighbours
- :

$$\|x_{k+1} - x_k\| < 2$$

- The function $q(x_k, x_{k+1})$ embodies the difference in direction between edge element x_k and the element x_{k+1} .
- Note that c is a negative constant. The function $s(x_k)$ is the edge strength (or gradient magnitude).

- The evaluation function $H(x_1, \dots, x_n)$ is in the form of a serial optimisation problem where not all variables in the evaluation function are simultaneously inter-related and can be solved by a multi-stage optimisation process referred to as *Serial Dynamic Programming* [Bellman and Dreyfus 1962].

Contour Following

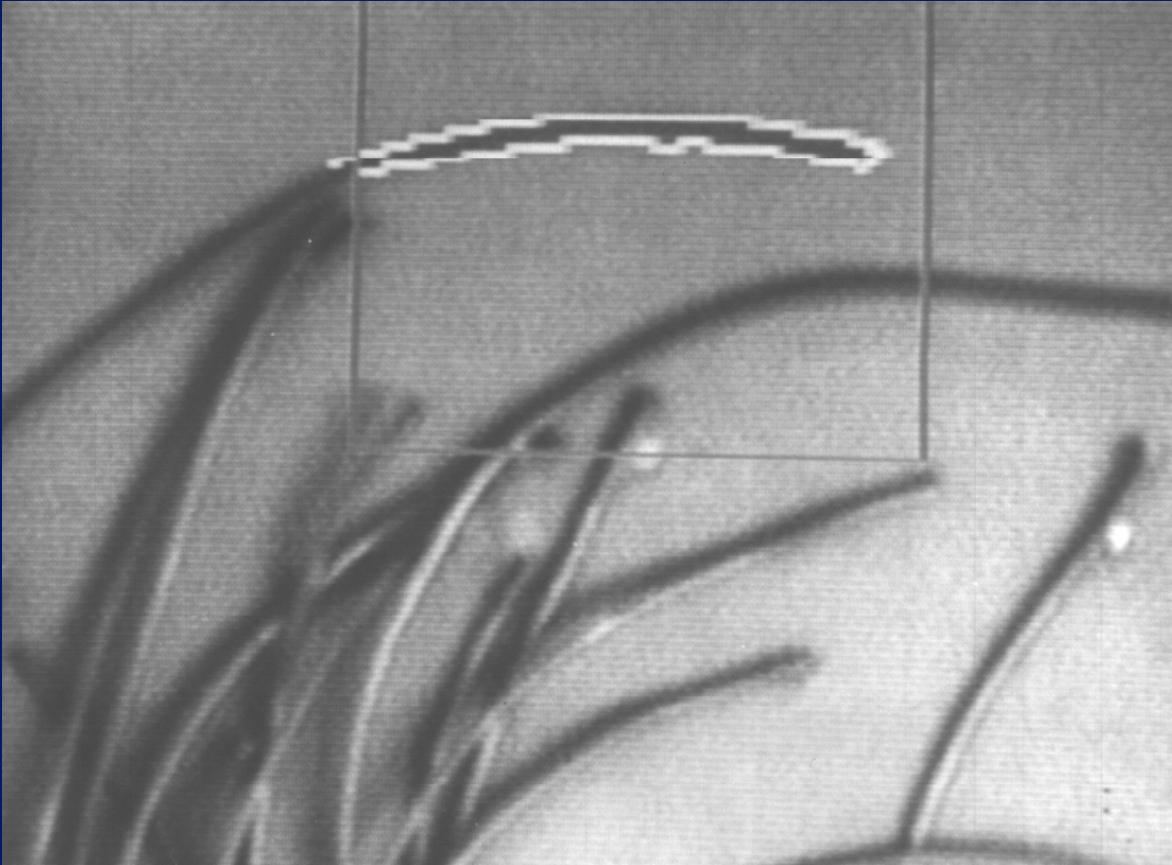
- *Contour following is a simple approach which uses no domain-dependant information. It “follows” the boundary or contour based exclusively on locally-derived data.*

- The basis of the technique is, essentially,
 - to start with a point that is believed to be on the boundary (some local edge point, say)
 - to extend the boundary by adding a neighbouring point in the contour direction (*i.e.* the direction which is normal to the gradient direction).

- This process of extension is reiterated starting at this new boundary pixel.
- *In this way a contour is followed around the boundary.*

- The basis for selecting a candidate varies from task to task and from algorithm to algorithm but normally is dependant at least on the gradient direction and on the gradient magnitude.

- Since contour following techniques are often based on gradient edge detectors, these techniques are normally most successful with images in which there is little noise.



- *Contour Following*

- To begin with, contour following algorithms cannot assume that the boundary constitutes a closed curve.

- Thus, the boundary is followed in two directions :
 - first in the forward boundary direction
 - and then in the reverse directions.

- The forward boundary direction is arbitrarily designated the direction equal to the edge normal (or gradient) direction + 90°
- and the reverse direction corresponds to the edge normal direction - 90°.

- The boundary following algorithm proceeds on a pixel to pixel basis, tracing the local maximum gradient given by the gradient direction at that point on the boundary to the next pixel.

- Tracing continues as long as the difference between the current and candidate pixel gradient directions is not too large ;
 - this helps avoid following boundaries into noisy areas which are characterised by frequent changes in edge direction.

- If no acceptable edge is encountered when tracing, a search is made in two zones ahead of the boundary :
 - the first zone separated by a gap of one pixel from the current boundary point,
 - the second by two pixels.

- If a suitable edge is found, the intervening gap pixels are filled in and the trace is restarted from this point.
- If none are found then the boundary is traced in the reverse direction from the original start point, providing this has not already been done.

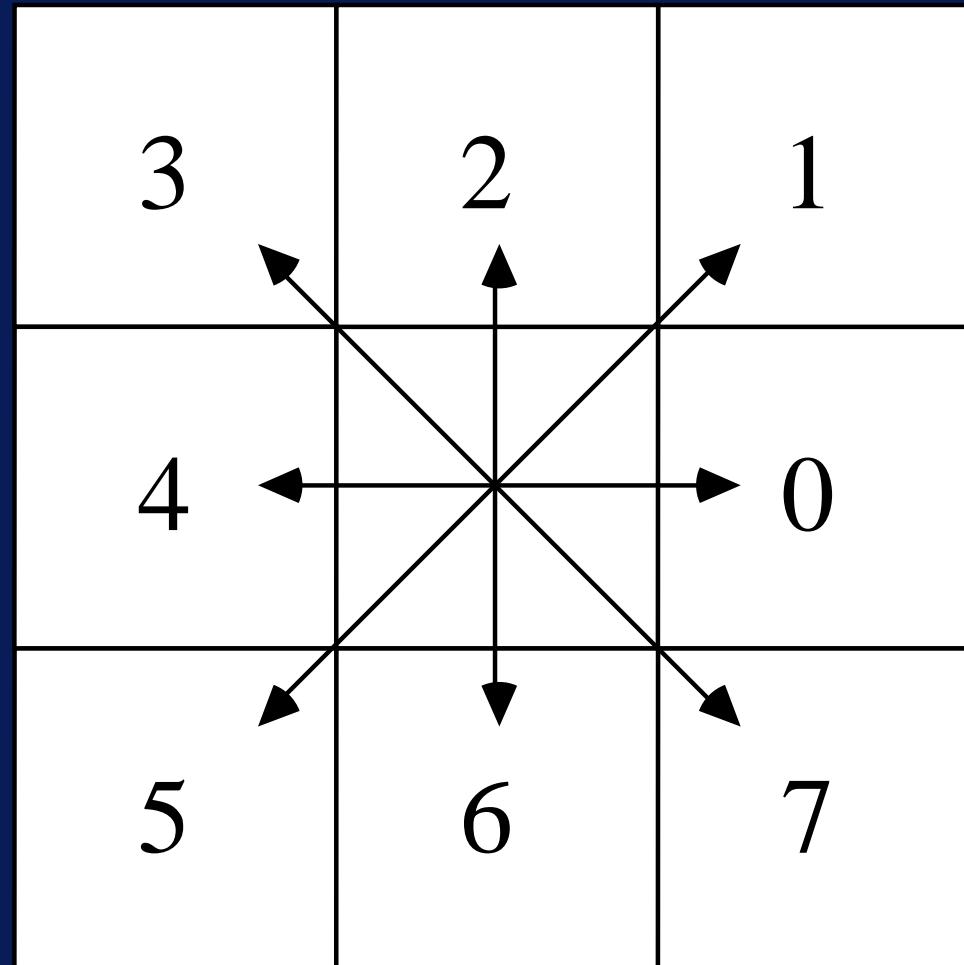
- The boundary following algorithm terminates when the boundary can be followed no further: that is, if neither the tracing nor the searching algorithms yield a valid boundary point.

- Boundary following also terminates if the original start point is re-encountered during tracing, signifying that a boundary is represented by a closed curve.
- *Thus, the extraction of both open and closed contours is facilitated.*

- As the algorithm traces around the boundary, it builds a Boundary Chain Code (BCC) representation of the contour.

- A Boundary Chain Code comprises
 - an integer pair, denoting the co-ordinates of an origin point on the contour
 - and a sequence (or chain) of integer values representing the direction of the next pixel on the contour.

- The range of possible directions is coarsely quantised and there are just eight possible directions corresponding to each of the eight pixels which are adjacent to the centre pixel in a 3*3 neighbourhood.



- *Freeman Direction Codes*

- *To avoid following multiple close parallel boundaries caused by the presence of "thick" edge responses, pixels in directions normal to the boundaries extracted by the following algorithm are suppressed, i.e. they are labelled, so as to exclude them from later consideration by either the trace or search modules.*

The Trace Algorithm

- The purpose of the trace algorithm is to follow a sequence of boundary pixels and to generate a model of the boundary, *i.e.* the BCC, by recording all the boundary pixels visited.

- It is assumed that the boundary pixels correspond to the pixels exhibiting the local maximum gradient magnitude and that the gradient direction indicates, approximately at least, the neighbouring pixel.

- The boundary direction, D , of an (edge) point is given by the edge gradient direction G as $D = G + 2$ for the forward direction and as $D = G - 2$ for the reverse direction (+90 and -90 degrees, respectively).
- All directional additions and subtractions are modulo 8 operations.

- The tracing algorithm chooses as candidates for inclusion in the boundary those pixels given by directions D , $D+1$ and $D-1$, that is, the pixel directly ahead of the current pixel and one pixel on either side of it.

- For example, if $D=1$ (boundary direction is 45°) then the pixels 0, 1 and 2 are chosen as candidates.

- The potential of each of these candidates to be a boundary point is evaluated as the difference between the points gradient magnitude and the pre-defined gradient magnitude threshold.

- If the pixel has been visited previously, either by the tracing or searching algorithms, or if the pixel overlaps the image boundary then it is assumed to have a negative potential.

- The candidate with the highest positive potential is selected as the next boundary point from which to continue the trace and is implicitly included in the list of boundary points by updating the BCC.

- If no candidate satisfies this condition, tracing is suspended and the search algorithm is invoked.

The Search Algorithm

- If all three neighbouring candidate boundary pixels selected by the trace algorithm fail to satisfy the criteria for inclusion in the boundary
 - then the boundary follower searches for other potential boundary pixels
 - which are not immediate neighbours of the current boundary pixel.

- The position and direction of this search is dependant on the direction of the current boundary point.
 - The associated region for searching is termed the “zone of influence” of the current boundary pixel.

- Two zones of influence, labelled A and B , of a boundary pixel are defined, based on the boundary direction D .
- Zone A comprises those pixels ahead of the boundary direction, separated from the current point by a one pixel gap. Pixels in zone B are separated by a two pixel gap.

- A pixel in a particular zone of influence is selected for inclusion in the boundary if it satisfies the following two criteria :
 1. The gradient magnitude is greater than the given threshold.
 2. The gradient direction is within a certain directional range.

- This directional range is based on the orientation of the current boundary point and the position of this candidate pixel relative to the current boundary point.

- The order in which the individual pixels are tested within a zone is important. In general, they are searched in the following order : 3
- 1
- Direction of Boundary ==> 0
- 2
- 4

- Since searching terminates as soon as an acceptable edge point is encountered, this ordering favours the extraction of a point that coincides with the boundary direction.

- In addition, zone *A* is searched before zone *B* is; thus, the bridging of short boundary gaps is favoured over larger gaps.
- If a search is successful the BCC is updated to include the selected pixel and the immediate gap pixel(s).

- If the search is not successful, boundary following in the current bias (forward or reverse) is terminated.

- *Zone of Influence A*

		X		A	B
				A	B
				A	B
				A	B

Zone of Influence B

					D+1 D	D+1 D
					D+1 D	D+1 D
				X	D D±1	D D±1
					D D-1	D D-1
					D D-1	D D-1

- *Allowable Directional Variation*
 - *of Zones A and B*

Image Analysis

- *Automatically extracting useful information from an image of a scene.*

- Techniques vary across a broad spectrum, depending on the complexity of the image and, indeed, on the complexity of the information to be extracted from it.
 - Template Matching
 - Statistical Pattern Recognition and
 - The Hough Transform.

- Unfortunately, this classification is not particularly useful when one is trying to identify a technique for a potential application.

- We can also classify the types of analysis we wish to perform according to function.
 - *Inspection*
Ascertain whether or not the visual appearance of objects is as it should,

- *Location*

Note that the location of an object requires the specification of both position and orientation (in either 2D or 3D).

Also, the co-ordinates might be specified in terms of the image frame of reference (where distance is specified in terms of pixels) or in the real world where distances correspond to millimetres, say.

The latter obviously necessitates some form of calibration, since initial measurements will be made in the image frame of reference.

– *Identification* of object type.

- *Inspection Applications* utilise the template matching paradigm.
- *Location* problems utilise the template matching paradigm and the Hough Transform.

- *Identification* can be addressed using all three techniques, depending on the complexity of the image and the objects.

Template Matching

- Many of the applications of computer vision simply need to know whether an image contains some previously defined object.
- In particular, whether a pre-defined sub-image is contained within a test image.

- This sub-image is called a *Template*
 - an ideal representation of the pattern or object which is being sought in the image.

- The template matching technique :
 - Translation of the template to every possible position in the image;
 - Evaluation of a measure of the match between the template and the image at that position;
 - If the similarity measure is large enough then the object can be assumed to be present.

- If the template does represent the complete object, then the technique is sometimes referred to as *Global Template Matching*.

- On the other hand, local template matching utilises several templates of local features of the object, *e.g.* corners in the boundary or characteristic marks, to represent the object.

Measures of Similarity

- Several similarity measures are possible,
 - some based on the summation of differences between the image and template
 - other based on cross-correlation techniques.

- A common measure employed when comparing the similarity of two images (*e.g.* the template $t(i, j)$ and the test image $g(i, j)$) is the metric based on the standard Euclidean distance between two vectors, defined by :

$$E(m, n) = \sqrt{\sum_i \sum_j [g(i, j) - t(i - m, j - n)]^2}$$

- The summation is evaluated for all i , such that $(i-m)$ is a valid co-ordinate of the template sub-image.
- This definition amounts to translating the template $t(i, j)$ to a position (m, n) along the test image and evaluating the similarity measure at that point.

- The position (m, n) at which the smallest value of $E(m, n)$ is obtained corresponds to the best match for the template.
- Consider a simple one dimensional entity, e.g. size (represented by, say, length).

- To compare the difference in size of two objects we just :
 - subtract the values
 - square the difference
 - take the square root of the result,
- leaving us with the absolute difference in size :

$$d = \sqrt{(s_1 - s_2)^2}$$

- Extending this to the 2D case, we might wish to see how far apart two objects are on a table, *i.e.* to compute the distance between them.

- The difference in position is simply :

$$d = \sqrt{\left((x_1 - x_2)^2 + (y_1 - y_2)^2 \right)}$$

- Similarly, in 3D :

$$d = \sqrt{\left((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 \right)}$$

- We can easily extend this to n dimensions (although we lose the intuitive concept of “distance”) by just making each co-ordinate an independent variable which characterises the entities we are comparing.

- For example, a 10×10 image template comprises 100 independent pixels, each of which specifies the template sub-image.

- Thus, we are now dealing with a
- $10 * 10 = 100$ dimensional comparison and the difference between the two sub-images is :

$$d = \sqrt{\left((image(1,1) - template(1,1))^2 + \dots + (image(10,10) - template(10,10))^2 \right)}$$

- which is identical to our definition of the Euclidean metric.

- A frequently-used and simpler template-matching metric is based on the absolute difference of $g(i, j)$ and $t(i-m, j-n)$ rather than the square of the difference.
- It is defined by :

$$S(m, n) = \sum_i \sum_j |g(i, j) - t(i - m, j - n)|$$

- As before, the summation is evaluated for all i and j , such that $(i-m, j-n)$ is a valid coordinate of the template sub-image.
- Note that the summation of the last term is constant since it is a function of the template only and is evaluated over the complete domain of the template.

- If it is assumed that the first term is also constant, or that the variation is small enough to be ignored, then $E^2(m, n)$ is small when the summation of the middle term is large.

- Thus, a new similarity measure might be $R(m, n)$, given by :

$$R(m, n) = \sum_i \sum_j g(i, j)t(i - m, j - n)$$

- again summing over the usual range of i and j .

- $R(m, n)$ is the familiar cross-correlation function.
- The template $t(i-m, j-n)$ and the section of $g(i, j)$ in the vicinity of (m, n) are similar when the cross-correlation is large.

- If the assumption that the summation of $g(i, j)$ is independent of m and n is not valid, an alternative to computing R is to compute the normalised cross-correlation $N(m, n)$, given by :

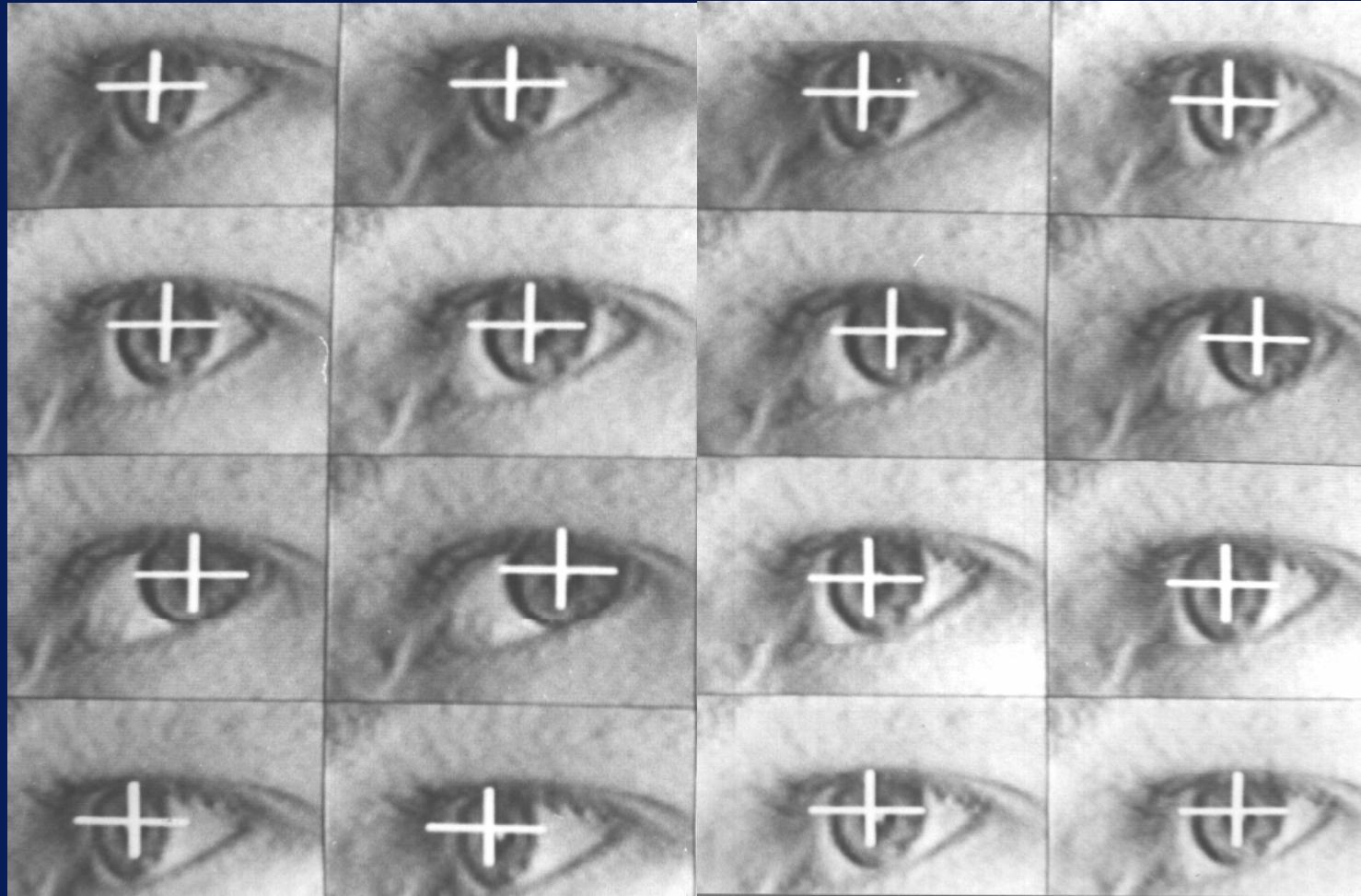
$$N(m, n) = \frac{R(m, n)}{\sqrt{\sum_i \sum_j g(i, j)^2}}$$

- ... summing over the usual range i and j .
Note that, by the Cauchy-Schwarz inequality,

$$N(m, n) \leq \sqrt{\sum_i \sum_j t(i - m, j - n)^2}$$

- Hence, the normalised cross-correlation may be scaled so that it lies in the range 0 to 1 by dividing it by the above expression. Thus, the normalised cross-correlation may be redefined :

$$N(m, n) = \frac{R(m, n)}{\left(\sqrt{\sum_i \sum_j g(i, j)^2} \sqrt{\sum_i \sum_j t(i - m, j - n)^2} \right)}$$



*Eye-tracking using
Normalised Cross-correlation*

Local Template Matching

- One of the problems of template matching is that each template represents the object or part of it as we expect to find it in the image.
 - No cognisance is taken of variations in scale or in orientation.

- If the expected orientation can vary, then we will require a separate template for each orientation and each one must be matched with the image.
- Thus template matching can become computationally expensive, especially if the templates are large.

- Use much smaller local templates to detect salient features in the image which characterise the object we are looking for.
- The spatial relationship between occurrences of these features are then analysed.

- We can infer the presence of the object if valid distances between these features occur.

- In summary, template matching techniques are useful in applications that can be severely restricted and where the number of objects and the variability of these objects, is small.
- *It is not an approach that is applicable in general situations.*

Decision-Theoretic Approaches

- Find objects within the image and identify or classify those objects.
- The central assumption is that the image depicts one or more objects and that each object belongs to one of several distinct and exclusive predetermined classes,

– *i.e.*, we know what objects exist and an object can only have one particular type or label.

Components of Statistical Pattern Recognition Process

- Three components of this type of pattern recognition process :
 - an object isolation module
 - a feature extraction module
 - a classification module.

- Each of these modules is invoked in turn and in the order given; the output of one module forming the input of the next.
- Thus, the object isolation module operates on a digital image and produces a representation of the object.

- The feature extraction module then abstracts one or more characteristic features and produces a (so-called) feature vector.
- The feature vector is then used by the classification module to identify and label each object.

- Object isolation, often referred to as segmentation, is in effect the grouping process which we discussed in the previous chapter.
- The similarity measure upon which the grouping process is based in this instance is the grey-level of the region.

- The next phase of the recognition scheme is the extraction of features which are characteristic of the object.
- The selection of the features to be used is an extremely important task, since all subsequent decisions will be based on them.

- Frequently it is intuition and experience which guides the selection process.

- Normally, we will
 - identify a number of reasonable feasible potential features,
 - test these to check their performance,
 - then select the final set of features to be used in the actual application.

- When selecting features , you should bear in mind the desirability of each feature being
 - *Independent*
a change in one feature should not change significantly the value of another feature.

- *Discriminatory*

Each feature should have a significantly different value for each different object.

- *Reliable*

Features should have the same value for all objects in the same class/group.

- The computational complexity of the pattern recognition exercise increases very rapidly as the number of features increases.
- Hence it is desirable to use the fewest number of features possible, while ensuring a minimal number of errors.

Simple Feature Extraction

- Most features are either based on the size of the object or on its shape.
- The most obvious feature which is based on size is the area of the object
 - simply the number of pixels comprising the object multitude by the area of a single pixel (frequently assumed to be a single unit).

- If we are dealing with grey-scale images, then the Integrated Optical Density (IOD) is sometimes used :
 - the area multiplied by the average grey level of the object and essentially
 - provides a measure of the ‘weight’ of the object, where the pixel grey-level encodes the weight per unit area.

- The length and the width of an object describe, too; its size.
- However, since we will not know its orientation, in general, we may have to first compute its orientation before evaluating the minimum and maximum extent of its boundary.

- Thus, these measures should always be made with respect to some rotation-invariant datum line in the object, *e.g.*, its major or minor axis.

- The *Minimum Bounding Rectangle* is a feature which is related to this idea of length and width.
- This is the smallest rectangle which can completely enclose the object.

- The main axis of this rectangle is in fact the principle axis of the object itself and, hence, the dimensions of the minimum bounding rectangle correspond to the features of length and width.

- Quite often, the distance around the perimeter of the object can be used for discriminating between two objects.
- Depending on how the object is represented, it can be quite trivial to compute the length of the perimeter and this makes it an attractive feature for industrial vision applications.

- Features which encode the shape of an object are usually very useful for the purposes of classification and because of this, Chapter 7 has been entirely given over to them.

- For the present, we will content ourselves by mentioning two very simple shape measures :
 - Rectangularity
 - Circularity

- There are two popular measures of rectangularity, both of which are easy to compute.

- The first is the ratio of the area of the object to the area of the minimum bounding rectangle :

$$R = \frac{A_{\text{object}}}{A_{\text{min. bound. rectangle}}}$$

- This feature takes on a minimum value of 1 for a perfect rectangular shape and tends toward zero for thin curvy objects.

- The second measure is the aspect ratio and is simply the ratio of the width of the minimum bounding rectangle to its length:

$$\text{Aspect Ratio} = \frac{W_{\text{min. bounding rectangle}}}{L_{\text{min. bounding rectangle}}}$$

- The most commonly used circularity measure is the ratio of the square of the perimeter length to the area:

$$C = \frac{A_{object}}{P_{object}^2}$$

- This assumes a maximum value for discs and tends toward zero for irregular shapes with ragged boundaries.

Classification

- The final stage of the statistical pattern recognition exercise;
 - classification of the objects on the basis of the set of features we have just computed, *i.e.* on the basis of the feature vector.
 - Feature values are viewed as ‘co-ordinates’ of a point in n -dimensional space.

- Objective of classification as being the determination of the sub-space to which the feature vector belongs.

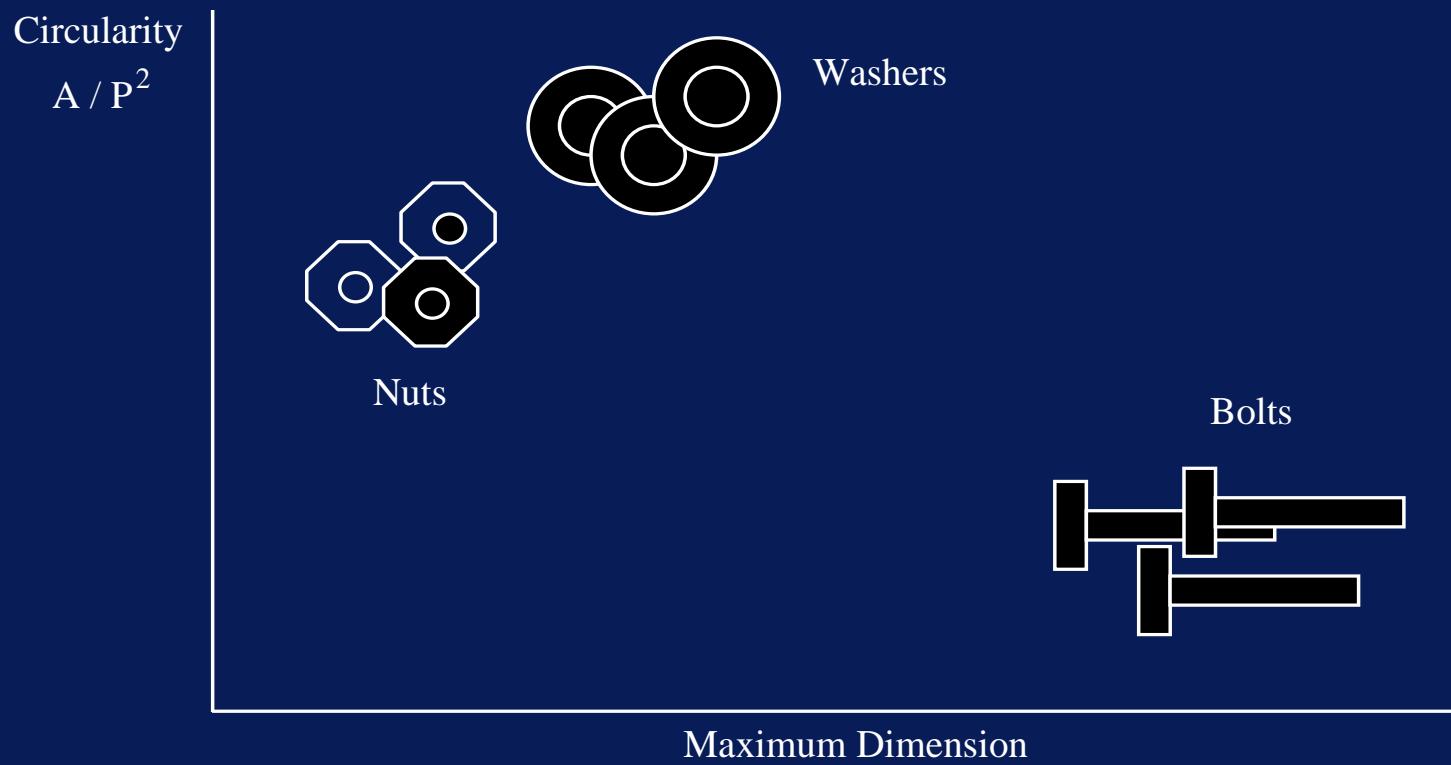
- Since each sub-space corresponds to a distinct object, the classification essentially accomplishes the object identification;
 - discriminate between nuts, bolts and washers on a conveyor belt.

- Assuming that we can adequately segment these objects, we might :
 - choose to use two features on which to base the classification;
 - washers and nuts are almost circular in shape, while bolts are quite long in comparison; use a circularity measure as one feature.

- washers have a larger diameter than nuts and bolts have an even longer maximum dimension: use the maximum length of the objects (its diameter in the case of the nuts and washers) as the second feature.

- Proceed to measure these feature values for a fairly large set of these objects, called the *Training Set*.
- Plot the results on a piece of graph paper (representing the two dimensional feature space, since there are two features).

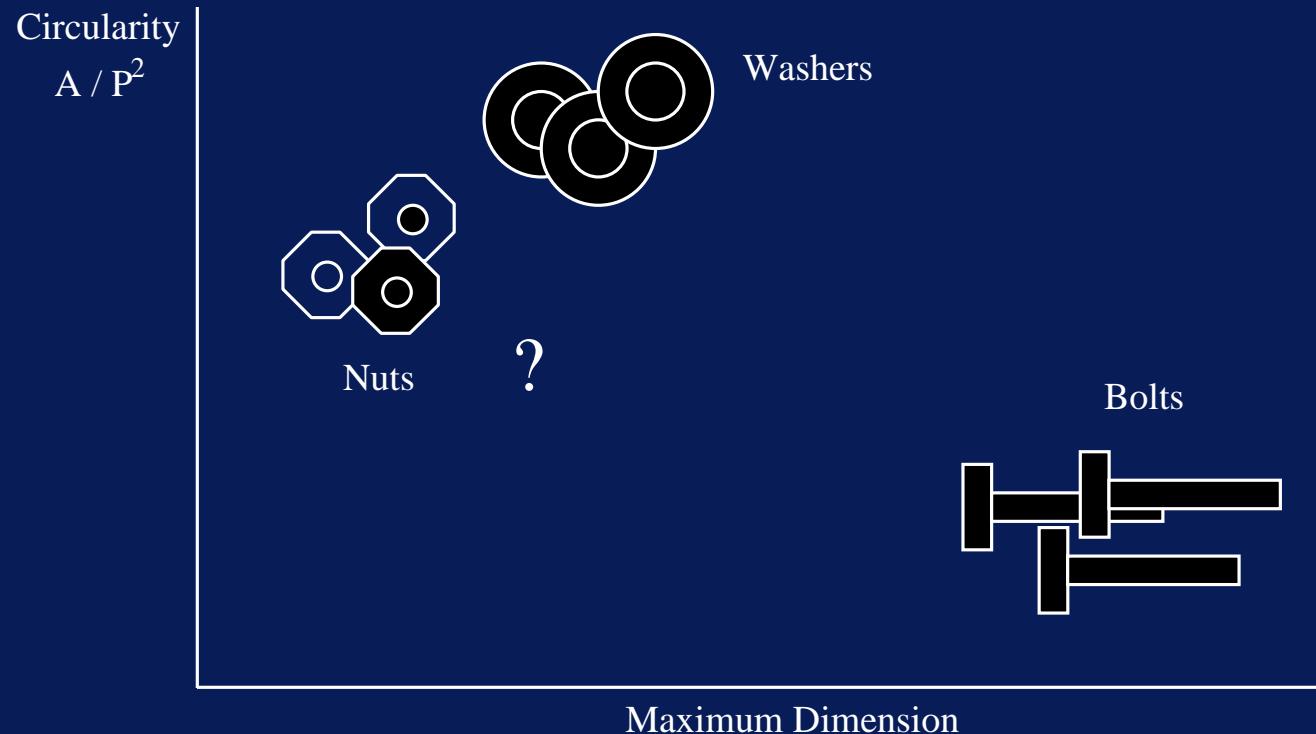
- We will probably observe the clustering pattern shown in Figure 6.2, where nuts, bolts and washers are all grouped in distinct sub-spaces.



Feature Space

- *Classify an unknown object*
 - generate the feature vector for this unknown object (*i.e.* compute the maximum dimension and its circularity measure A/P^2);
 - see where this takes us in the feature space.

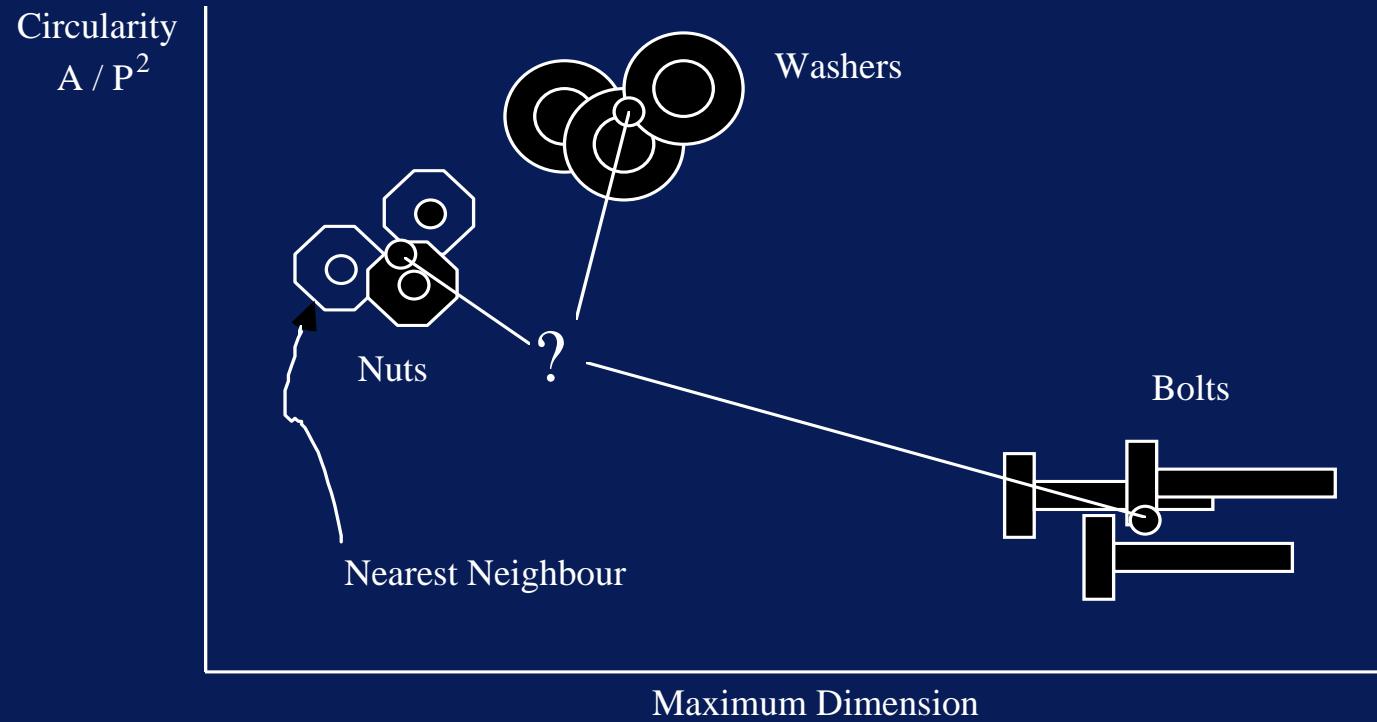
- The question is now: to which subspace does the vector belong, *i.e.*, to which class does the object belong?



*Co-ordinates of an unknown object
in the Feature Space*

- The nearest neighbour classification technique
 - classifies the object on the basis of the distance of the unknown object vector position from the centre of the three clusters
 - choosing the closest cluster as the one to which it belongs.

- The position of the centre of each cluster is simply the average of each of the individual training vector positions.



Nearest Neighbour Classification

A Synopsis of Classification using Bayes' Rule

- Maximum-likelihood classifier;
- Use an example which requires only one feature to discriminate between two objects; we do this because it is easier to visualise (and draw!!) the concepts being discussed.

- Suppose we wish to distinguish between nuts and bolts (no washers this time).
- Circularity measure will suffice and we have now just one feature and a one-dimensional feature space with two classes of object; nuts and bolts.
- Let us refer to these classes as C_n and C_b .

- Let us also refer to the circularity feature value as x .
- The first thing that is required is the probability density function (PDFs) for each of these two classes, *i.e.*, a measure of the probabilities that an object from a particular class will have a given feature value.

- Since it is not likely that we will know these *a priori*, we will probably have to estimate them.

- The PDF for nuts can be estimated in a relatively simple manner
 - measuring the value of x for a large number of nuts,
 - plotting the histogram of these values,
 - smoothing the histogram,
 - normalising the values so that the total area under the histogram equals one.

- The normalisation step is necessary since probability values have between zero and one and the sum of all the probabilities (for all the possible circularity measures) must necessarily be equal to a certainty of having that object, *i.e.*, a probability value of one.

- The PDF for the bolts can be estimated in a similar manner.
- *Probability of each class occurring.*
- *Probability of objects in each class have a particular value of x in a little more detail.*

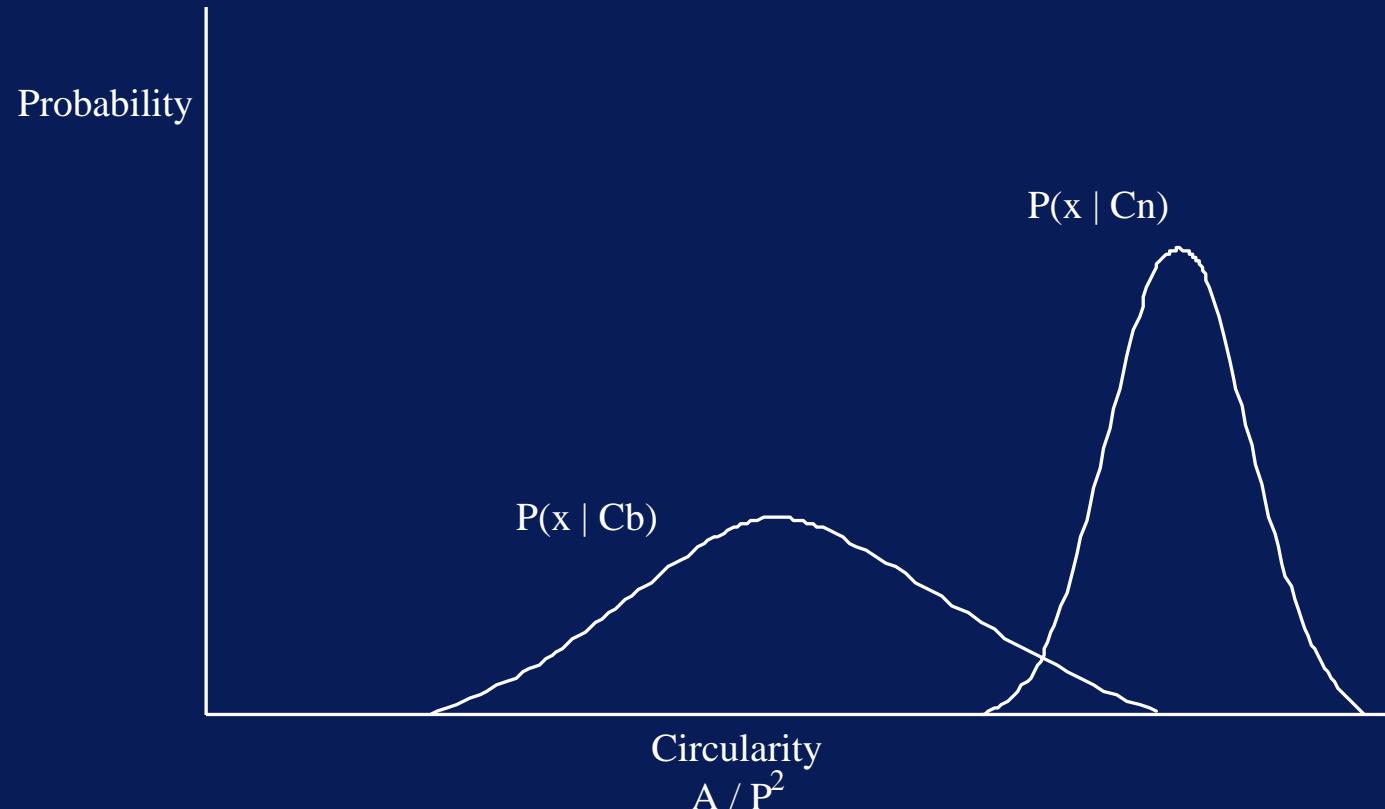
- We may know, for instance, that the class of nuts is, in general, likely to occur twice as often as the class of bolts.
- In this case we say that the a priori probability of the two classes are :
 $P(C_n) = 0.666$ and $P(C_b) = 0.333$.

- In fact, in this case, it is more likely that they will have the same *a priori* probabilities (0.5) since we usually have a nut for each bolt.

- The PDFs tell us the probability that the circularity x will occur, given that the object belongs to the class of nuts C_n in the first instance and to the class of bolts C_b in the second instance.

- This is termed the conditional probability of an object having a certain feature value, given that we know that it belongs to a particular class.

- Thus, the conditional probability, $P(x/C_b)$ enumerates the probability that a circularity x will occur, given that the object is a bolt.
- The two conditional probabilities $P(x/C_b)$ and $P(x/C_n)$ are shown in Figure 6.5, (*Slide 93*).



Conditional Probabilities

- Of course, this is not what are interested in at all.
- We want the probability that an object belongs to a particular class, given that a particular value of x has occurred (*i.e.* been measured), allowing us to establish its identity.

- This is called the *a posteriori* probability, $P(C_i/x)$, that the object belongs to a particular class i and is given by Bayes' theorem :

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

where

$$P(x) = \sum_{i=1}^2 P(x|C_i)P(C_i)$$

- $P(x)$ is a normalisation factor which is used to ensure that the sum of the *a posteriori* probabilities sum to one, for the same reasons as mentioned earlier.

- In effect, Bayes' theorem allows us to use
 - the *a priori* probability of objects occurring in the first place,
 - the conditional probability of an object having a particular feature value given that it belongs to a particular class and ...

- actual measurement of a feature value (to be used as the parameter in the conditional probability)
- to estimate the probability that the measured object belongs to a given class.

- Once we can estimate the probability that, for a given measurement, the object is a nut and the probability that it is a bolt, we can make a decision as to its identity, choosing the class with the higher probability.

- This is why it is called the maximum likelihood classifier.
- Thus, we classify the object as a bolt if :

$$P(C_b|x) > P(C_n|x)$$

- Using Bayes' theorem again, and noting that the normalising factor $P(x)$ is the same for both expressions, we can rewrite this test as :

$$P(x|C_b)P(C_b) > P(x|C_n)P(C_n)$$

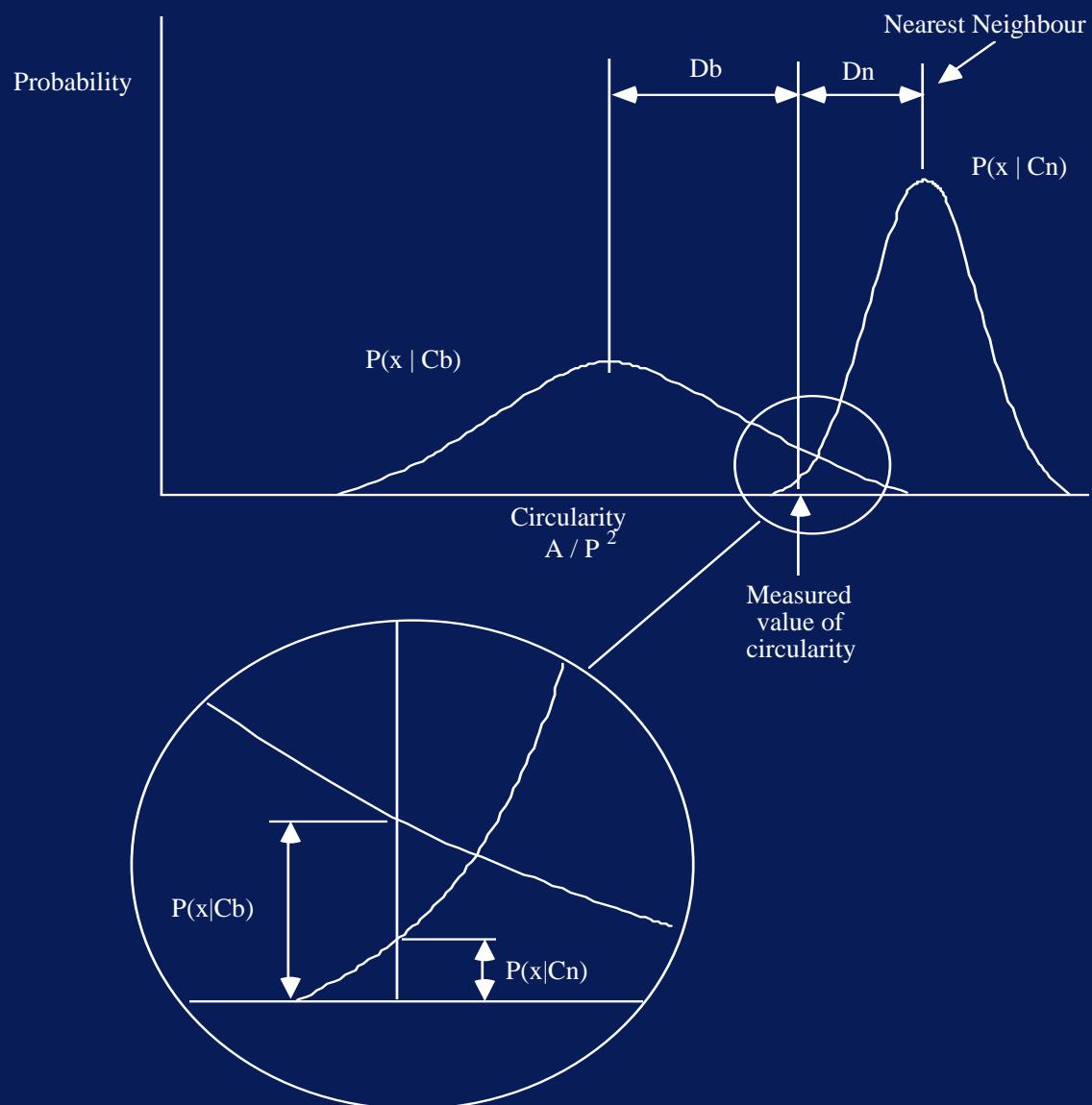
- If we assume that the chances of an unknown object being either a nut or a bolt are equally likely (*i.e.* $P(C_b) = P(C_n)$), then we classify the unknown object as a bolt if :

$$P(x|C_b) > P(x|C_n)$$

- For the example shown in Figure 6.6, (Slide 106), $P(x/C_b)$ is indeed greater than $P(x/C_n)$ for the measured value of circularity and we classify the object as a bolt.

- If, on the other hand, we were to use the nearest neighbour classification technique, we would choose the class whose mean value “is closer to” the measured value.

- In this case, the distance D_n from the measured value to the mean of the PDF for nuts is less than D_b , the distance from the measured value to the mean of the PDF for bolts; we would erroneously classify the object as a nut.



- We have restricted ourselves to a simple example with just one feature and a one-dimensional feature space.

- However, the argument generalises directly to an n -dimensional case, where we have n features in case the conditional probability density functions are also n -dimensional.

- In the two-dimensional case, the PDFs can be represented by grey-scale images : the grey-level encoding the probability.

The Hough Transform

- The Hough Transform is a technique which is used to isolate curves of a given shape in an image.

- The classical Hough transform requires that the curve be specified in some parametric form
 - is most commonly used in the detection of regular curves such as lines, circles and ellipses.

- Fortunately, this is not as restrictive as it might first seem since most manufactured parts do, in fact, have boundaries which are defined by such curves.

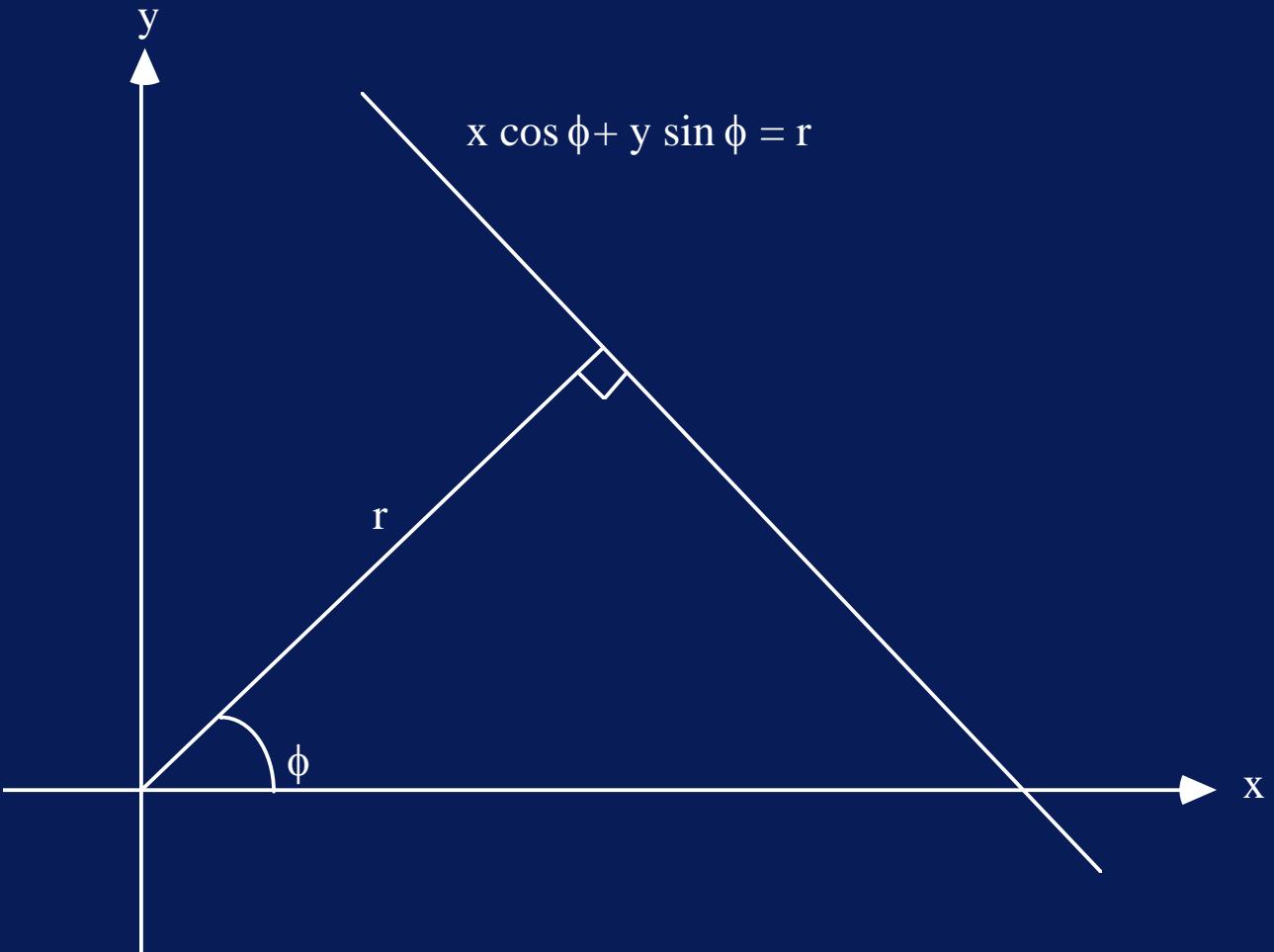
- However the Hough transform has been generalised so that it is capable of detecting arbitrary curved shapes :
 - very tolerant of gaps in the actual object boundaries or curves
 - relatively unaffected by noise.

Hough Transform for Line and Circle Detection

- We wish to detect a set of points lying on a straight line. The equation of a straight line is given in parametric form by the equation
 - :

$$x \cos \phi + y \sin \phi = r$$

- where
 - r is the length of a normal to the line from the origin
 - ϕ is the angle this normal makes with the X axis (refer to Figure 6.7) (*Slide 7*).



- *Parametric Representation of a Straight Line*

- If we have a point (x_i, y_i) on this line then

$$x_i \cos \phi + y_i \sin \phi = r$$

- For a given line, r and ϕ are constant.

- Suppose, however, that we do not know what line we want (*i.e.* r and ϕ are unknown) but we do know the co-ordinates of the point(s) on the line.

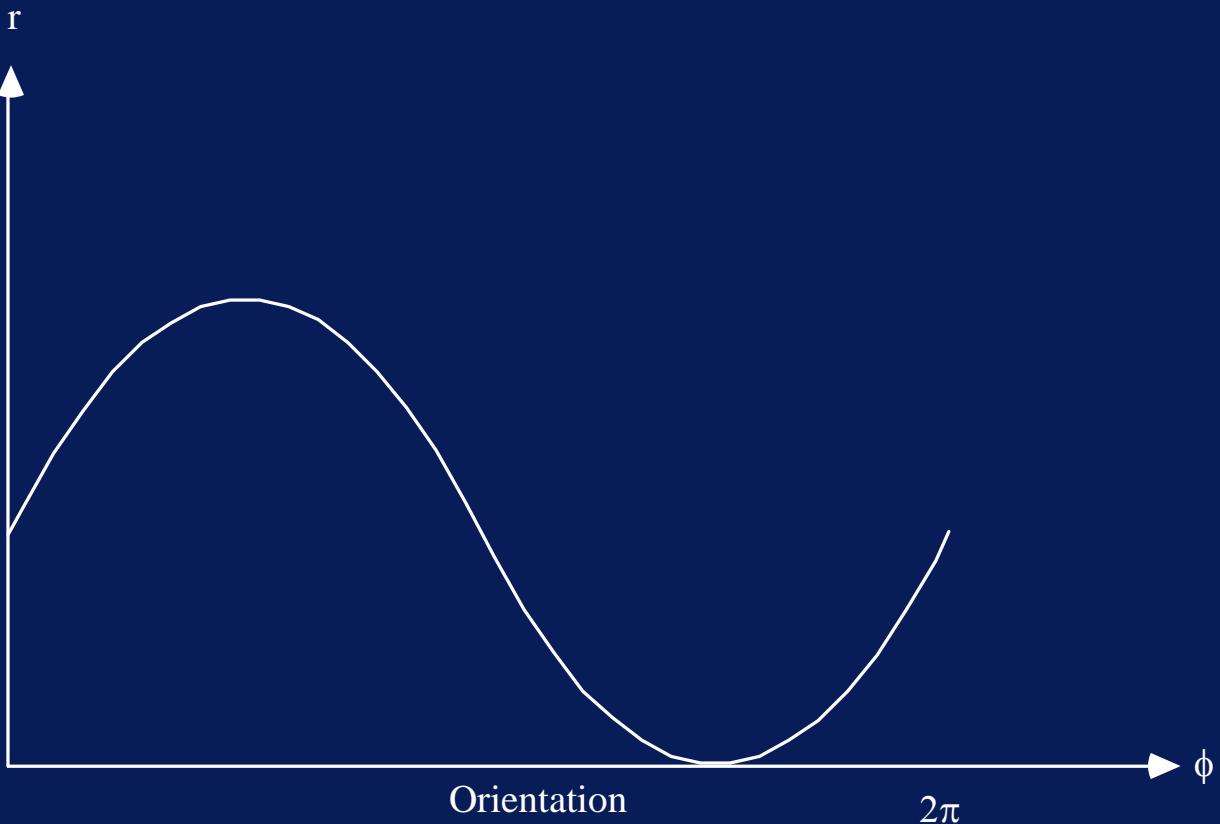
- Now we can consider r and ϕ to be variable and x_i and y_i to be constants. In this case, the equation :

$$x_i \cos \phi + y_i \sin \phi = r$$

- defines the values of r and ϕ such that the line passes through the point (x_i, y_i) .

- If we plot these values of r and ϕ , for a given point (x_i, y_i) , on a graph (see Figure 6.8), we see that we get a sinusoidal curve in $(r - \phi)$ space, *i.e.* in a space where r and ϕ are the variables.

- The transformation between the image plane (x and y co-ordinates) and the parameter space (r and ϕ co-ordinates) is known as the Hough Transform.
- Thus, the Hough transform of a point in the image plane is a sinusoidal curve in the Hough $(r - \phi)$ space.



- *Hough Transform of a Point (x_i, y_i)*

- However, collinear points in the image plane will give rise to transform curves which all intersect in one point since they share common r_i and ϕ_i and they all belong to the line given by :

$$x \cos \phi_i + y \sin \phi_i = r_i$$

- This, then, provides us with the means to detect collinear points, *i.e.* lines.

- First of all, we must sample the Hough transform space, *i.e.* we require a discrete representation of $(r - \phi)$ space.
 - Since ϕ varies between 0 and 2π radians, we need only decide on the required angular resolution to define the sampling.

- For example, a 6° resolution on the angle of the line might suffice, in which case we have a $360^\circ/6^\circ = 60$ discrete values of ϕ .

- Similarly, we can limit r by deciding on the maximum distance from the origin (which is effectively going to be the maximum size of the image, 256 pixels in length, say).

- Our representation of $(r - \phi)$ space is now simply a 2D array of size $256 * 60$, each element corresponding to a particular value of r and ϕ .

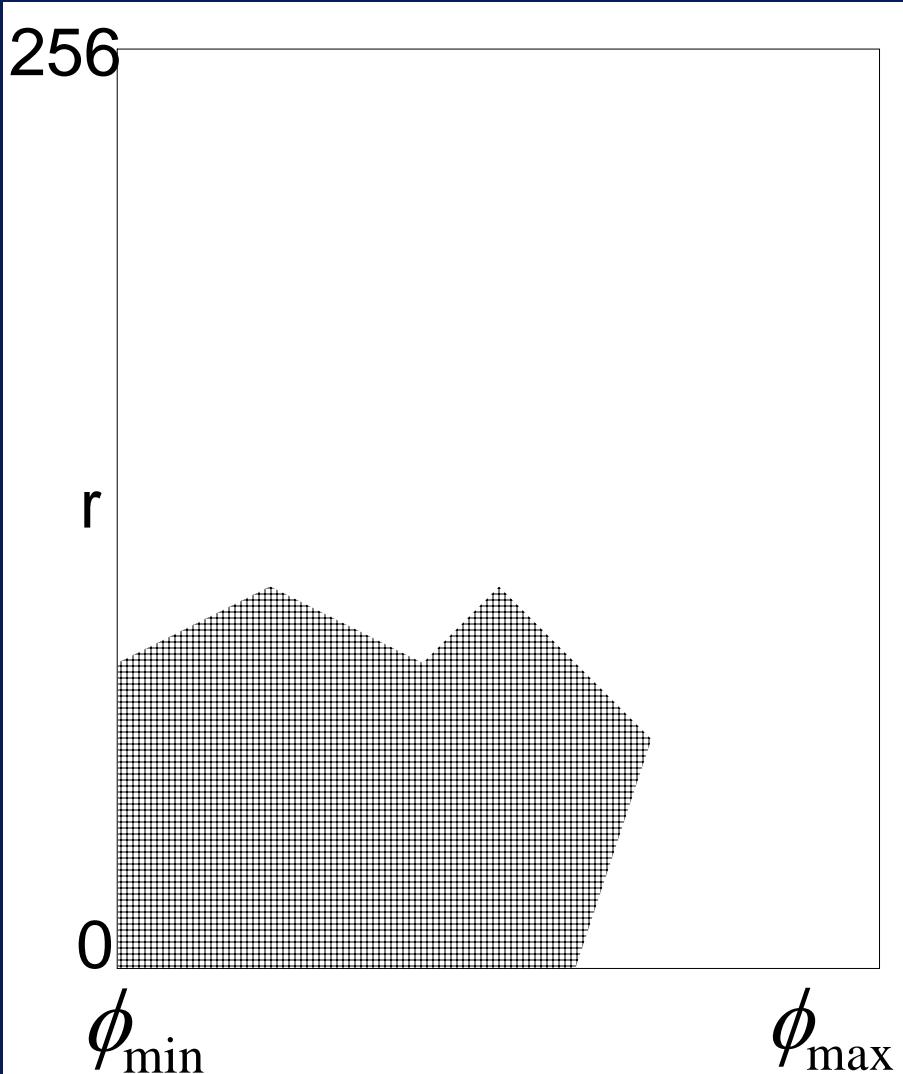
- This is called an *accumulator* since we are going to use it to collect or accumulate evidence of curves given by particular boundary points (x, y) in the image plane.

- For each boundary point (x_i, y_i) in the image, we increment all accumulator cells such that the cell co-ordinates (r, ϕ) satisfies the equation

$$x_i \cos \phi + y_i \sin \phi = r$$

- We have done this for all available
- (x_i, y_i) points, we scan the accumulator searching for cells which have a high count since these will correspond to lines for which there are many points in the image plane.

- In fact, because there is likely to be some errors in the actual position of the x and y co-ordinates, giving rise to errors in r and ϕ , we search for clusters of points in the accumulator having high counts, rather than searching for isolated points.



- *Hough Transform Accumulator Array*

* Some edge detectors, e.g. the Sobel operator, are usually formulated such that they directly yield the gradient direction which is equivalent to ϕ .

- Since edge detection processes are often employed in generating the candidate boundary points in the image and, in general, these yield not only the position of the edge (x_i, y_i) , but also its orientation* θ , where $\theta = \phi + 90^\circ$,

- ... we can use this information to simplify the Hough Transform and, knowing x_i , y_i and ϕ , use

$$x_i \cos \phi + y_i \sin \phi = r$$

- to compute r giving the co-ordinates of the appropriate accumulator cell to be incremented.

- ```
/* Pseudo-code for Hough Transform: Line Detection
*/

```
- Quantise the Hough transform space:  
Identify maximum and minimum values of  $r$  and  $\phi$   
and the total number of  $r$  and  $\phi$  values.
- Generate an accumulator array  $A(r, \phi)$ ; set all  
values to 0.

† Remember to normalise the result so that it lies in the interval  $0 - 2\pi$

- For all edge points  $(x_i, y_i)$  in the image

Do

compute the normal direction  $\phi$  (gradient direction or orientation - 90degrees)<sup>†</sup>

compute  $r$  from  $x_i \cos \phi + y_i \sin \phi = r$

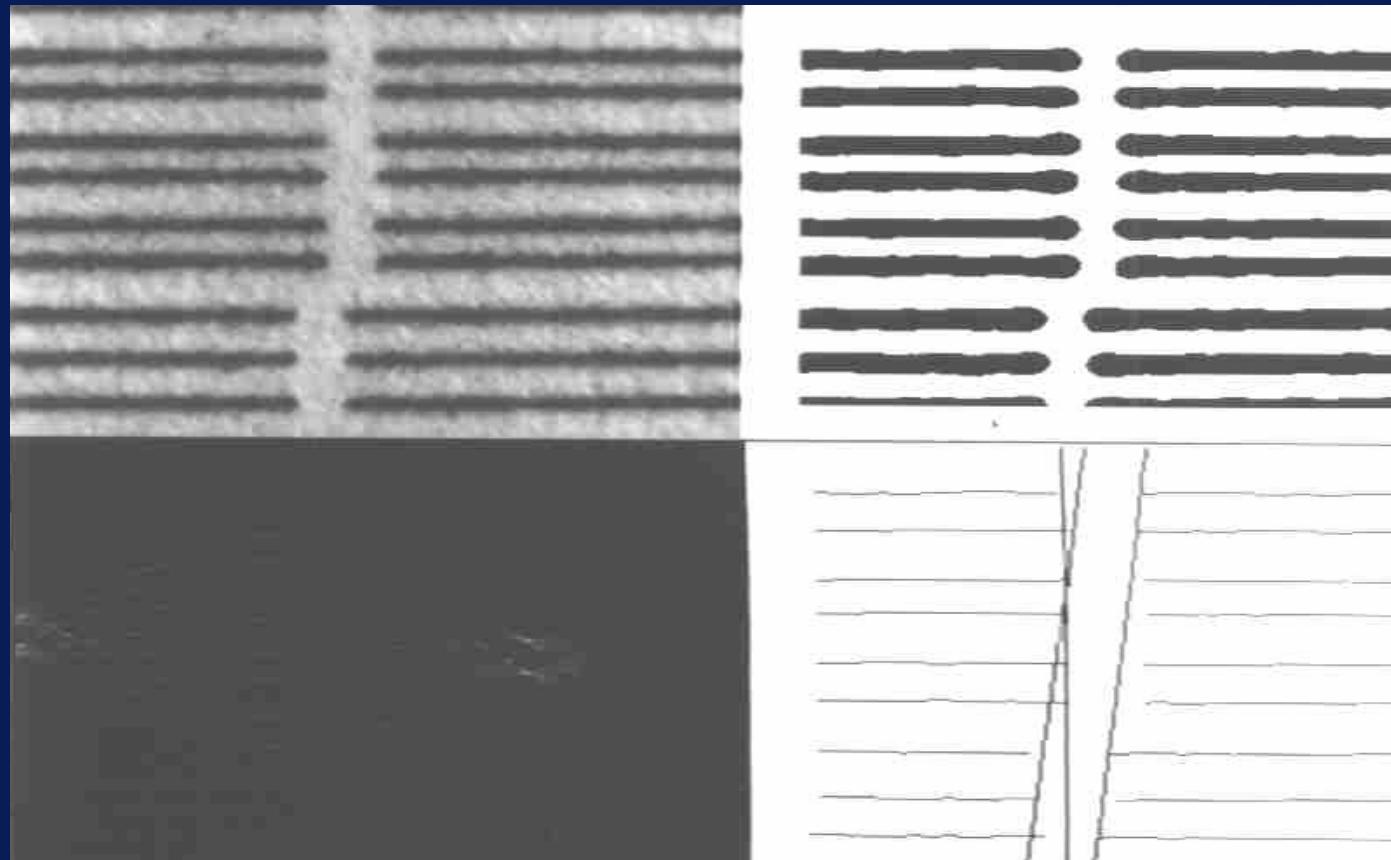
increment  $A(r, \phi)$

- For all cells in the accumulator array

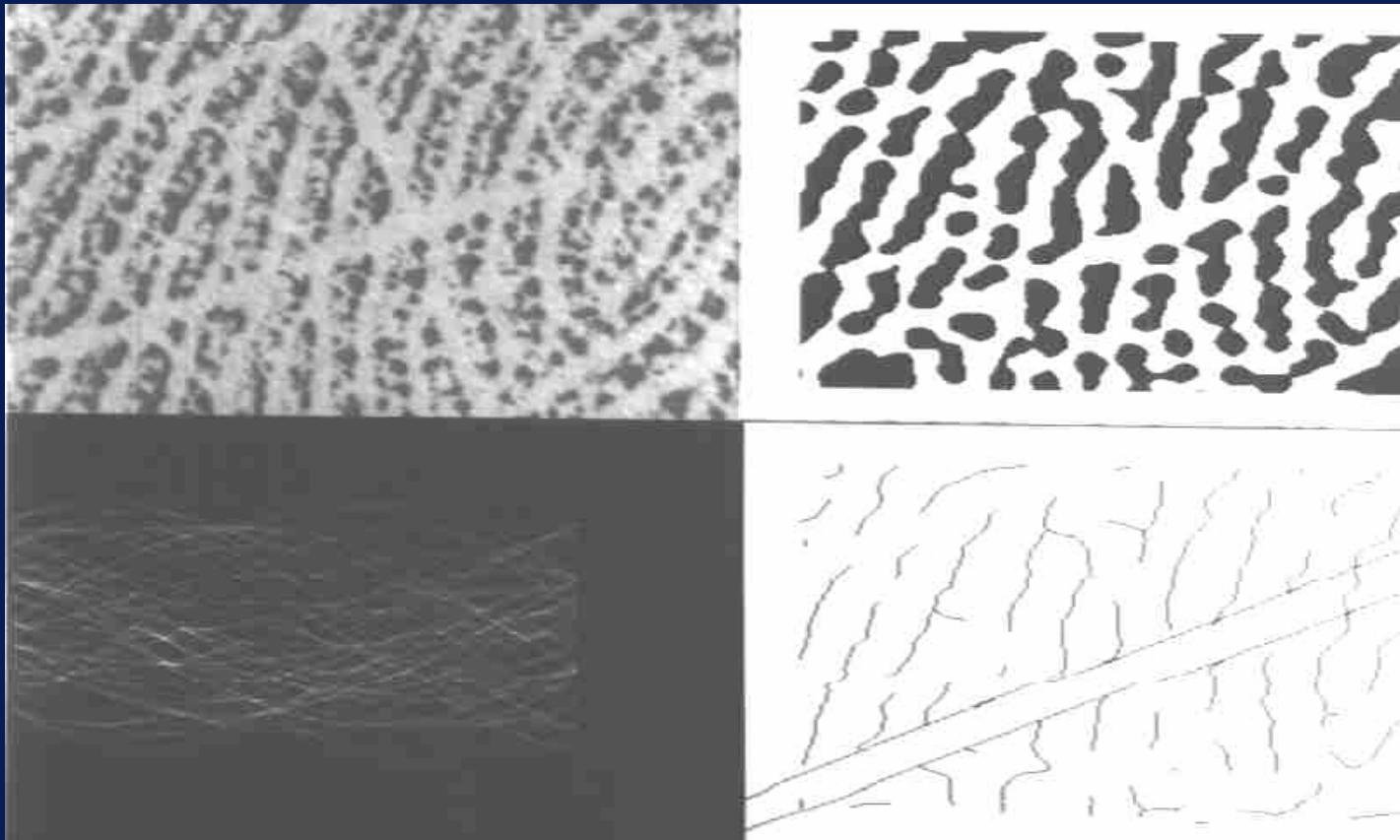
Do

Search for maximum values

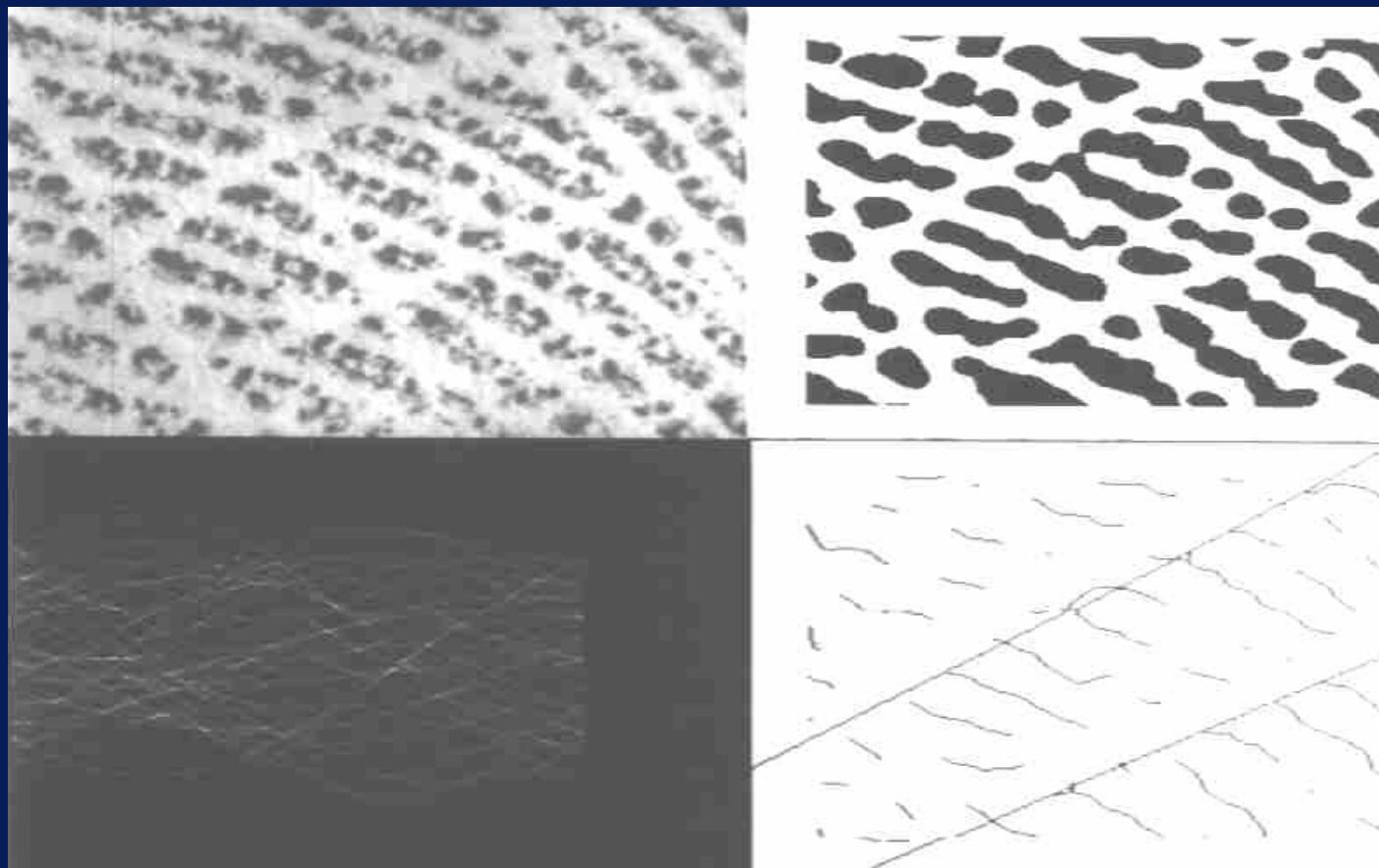
The co-ordinates  $r$  and  $\phi$  give the equation of the corresponding line in the image.



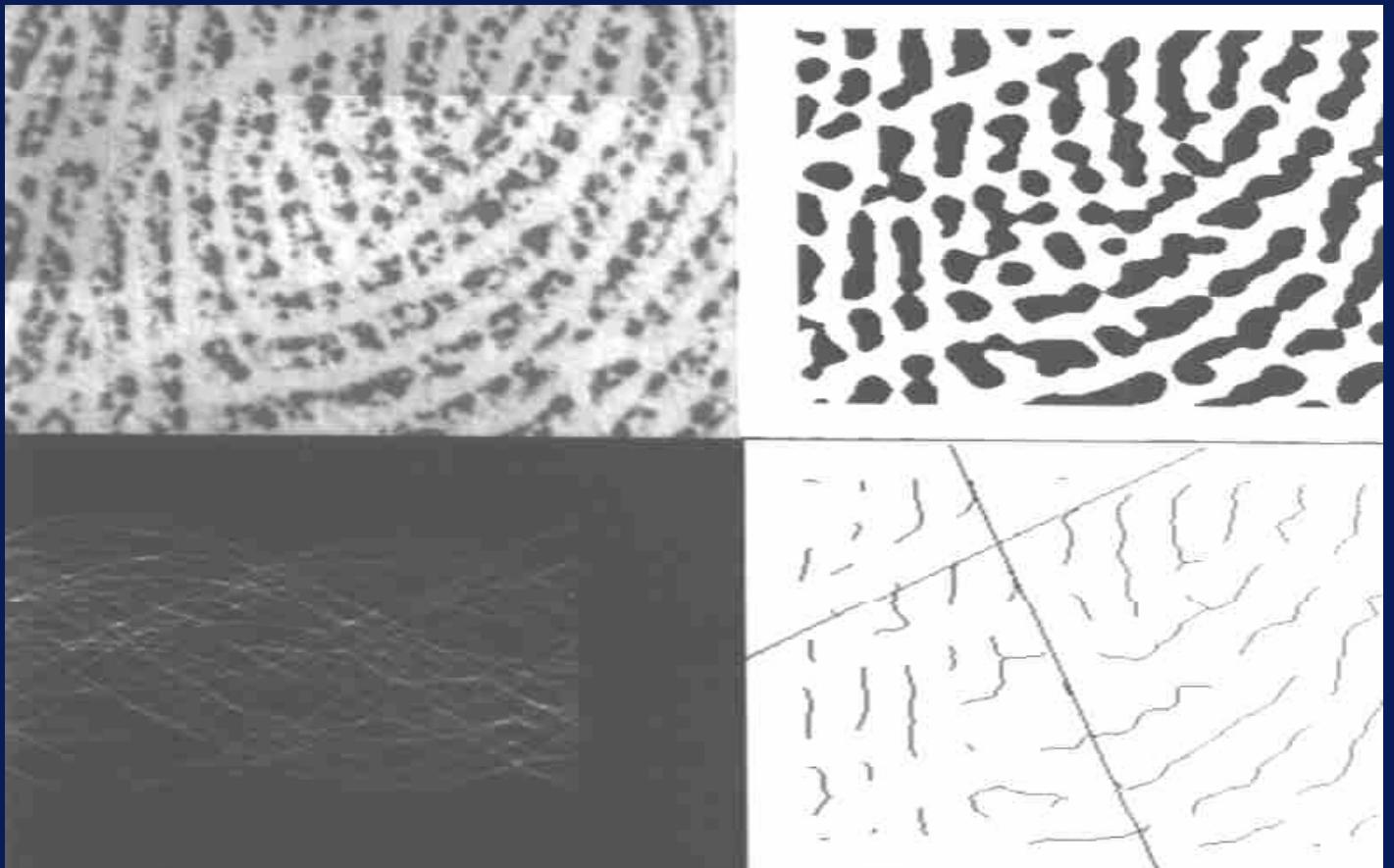
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



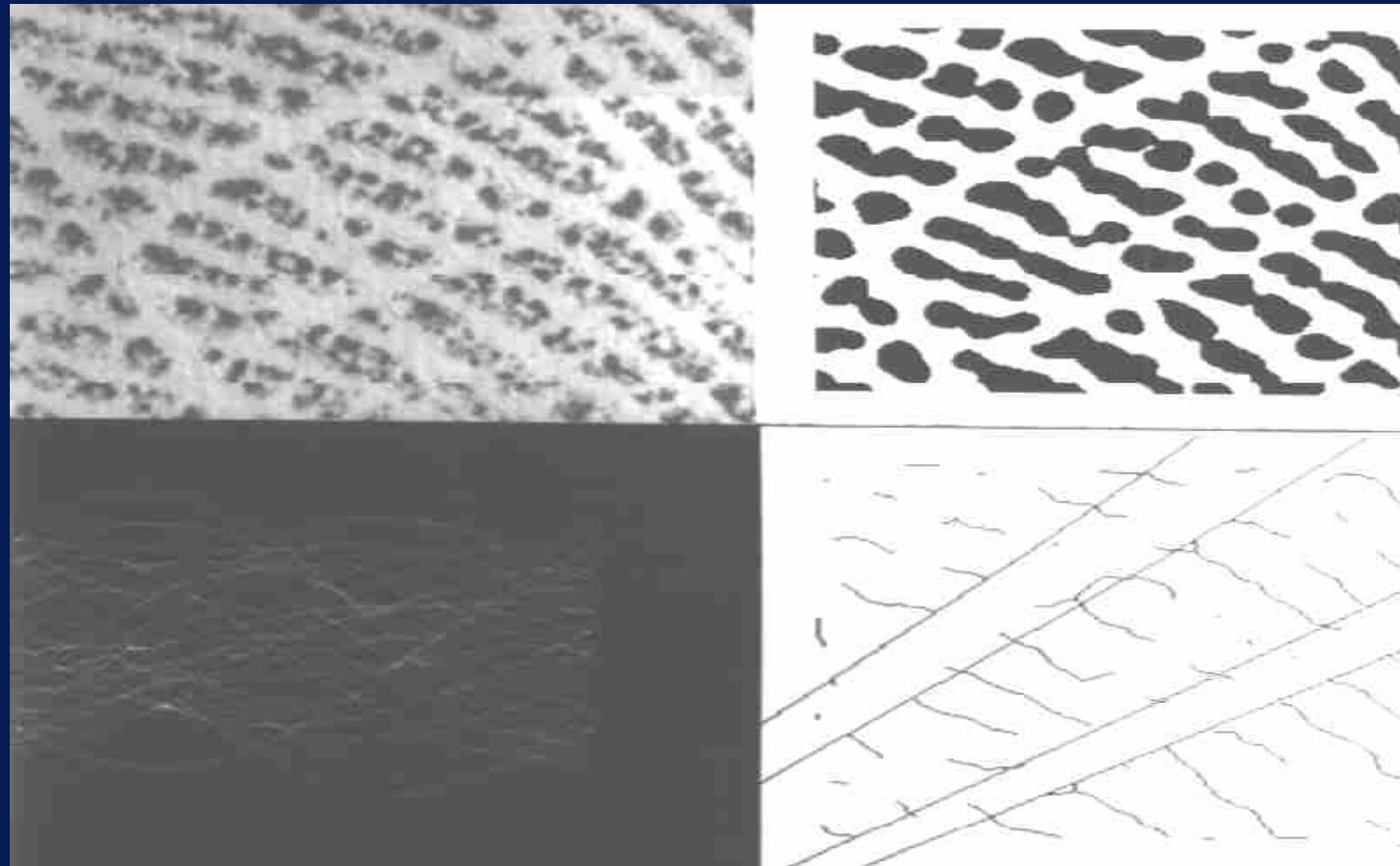
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



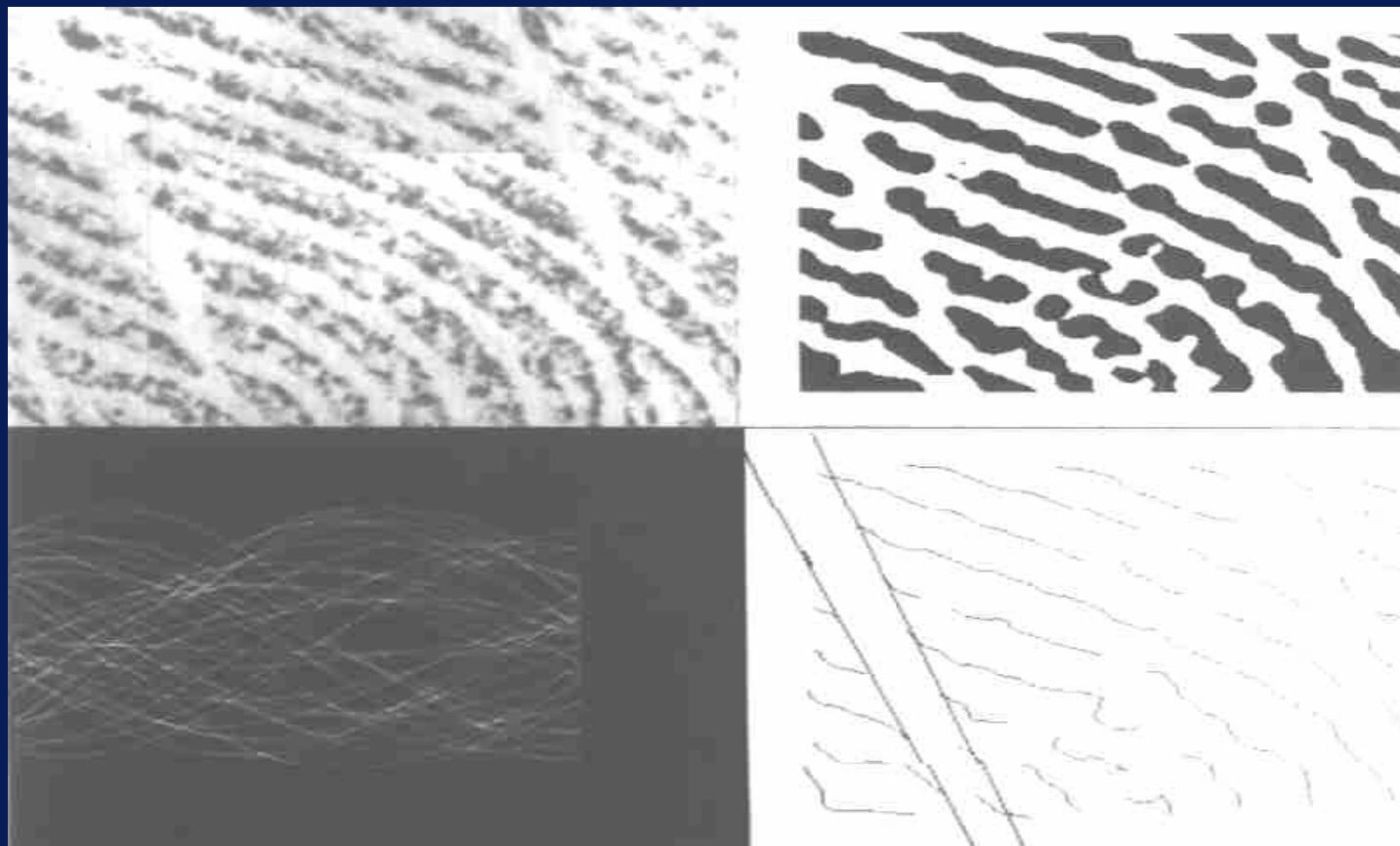
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



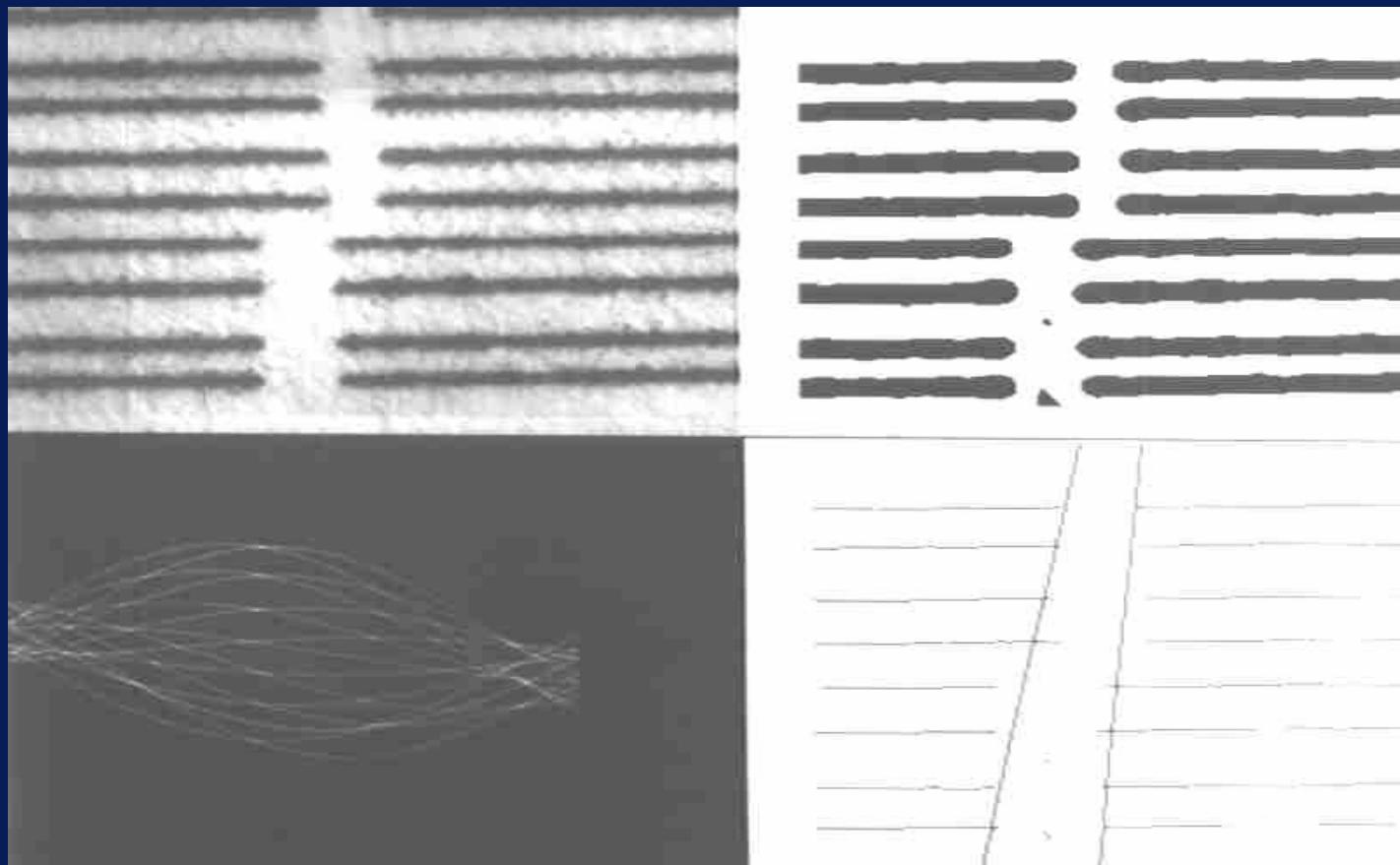
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



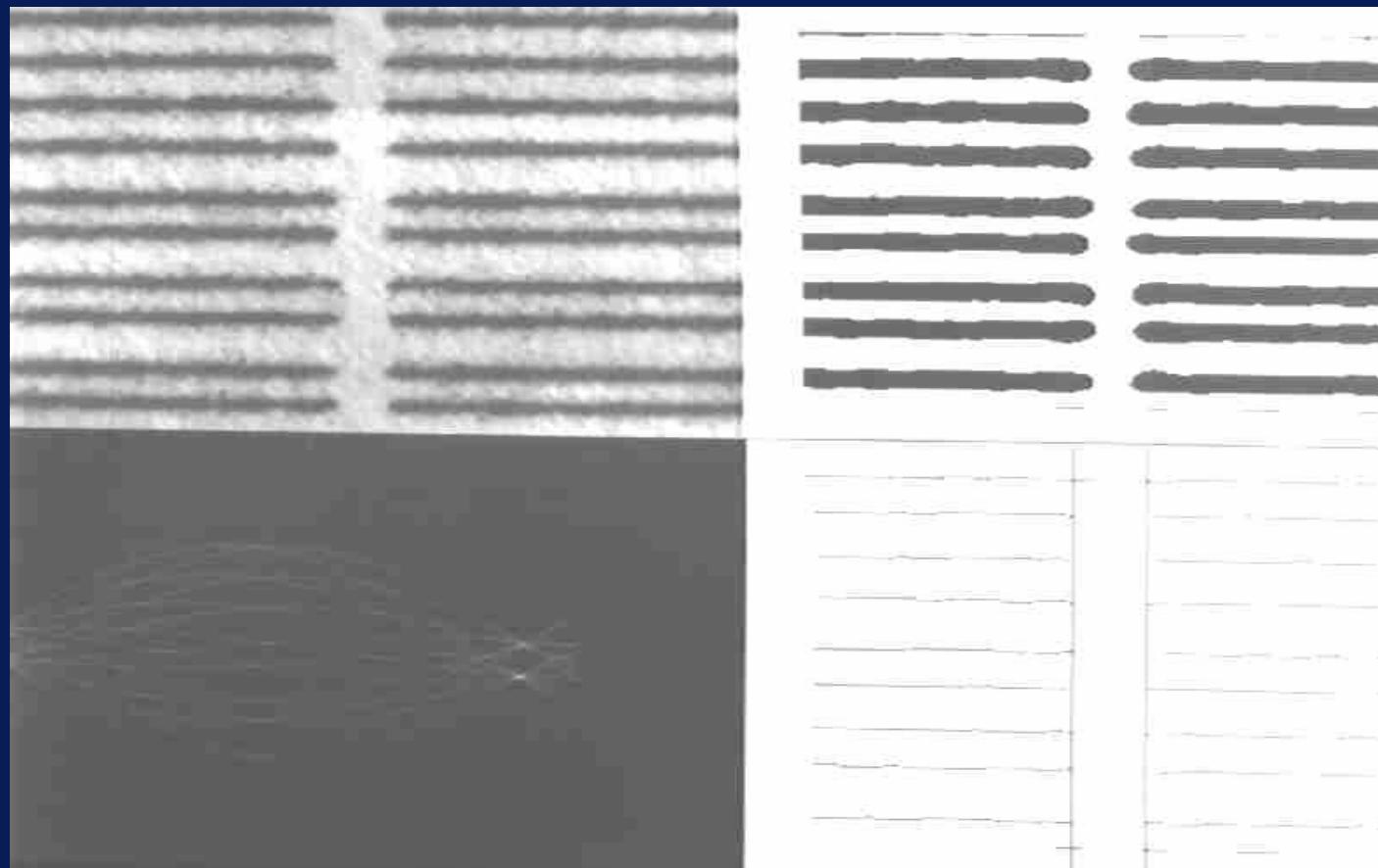
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



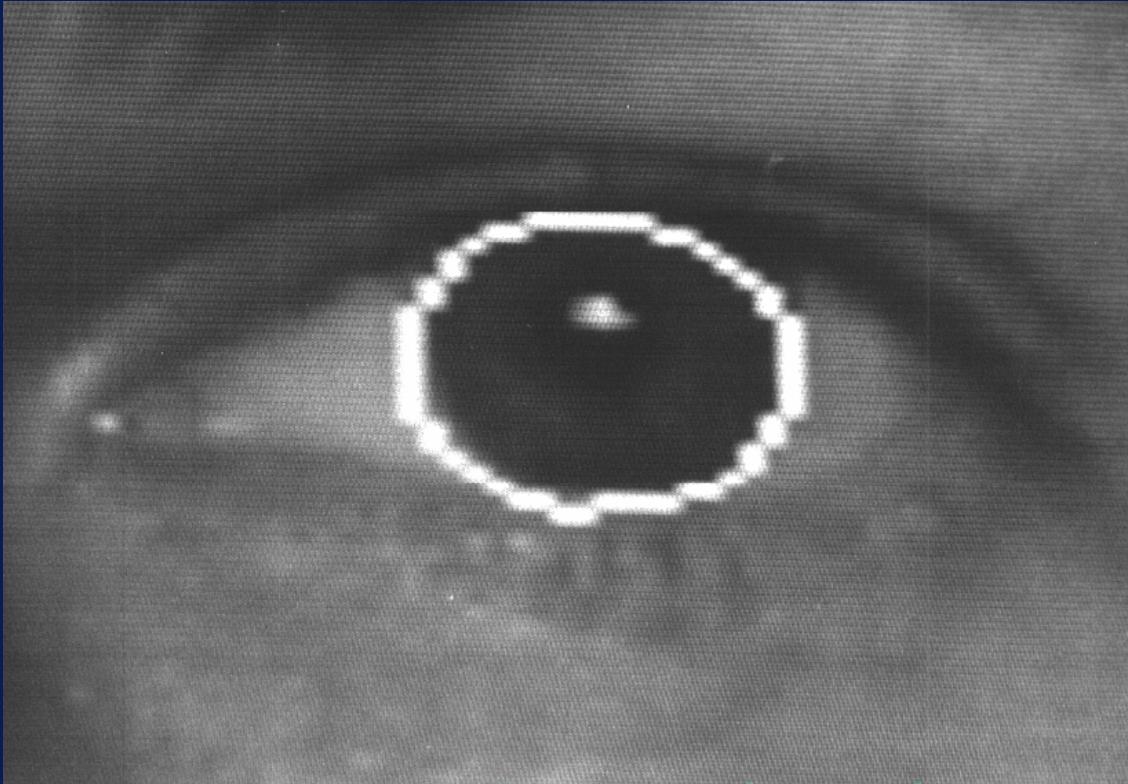
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))

- Just as a straight lines can be defined parametrically, so can a circle. The equation of a circle is given by :

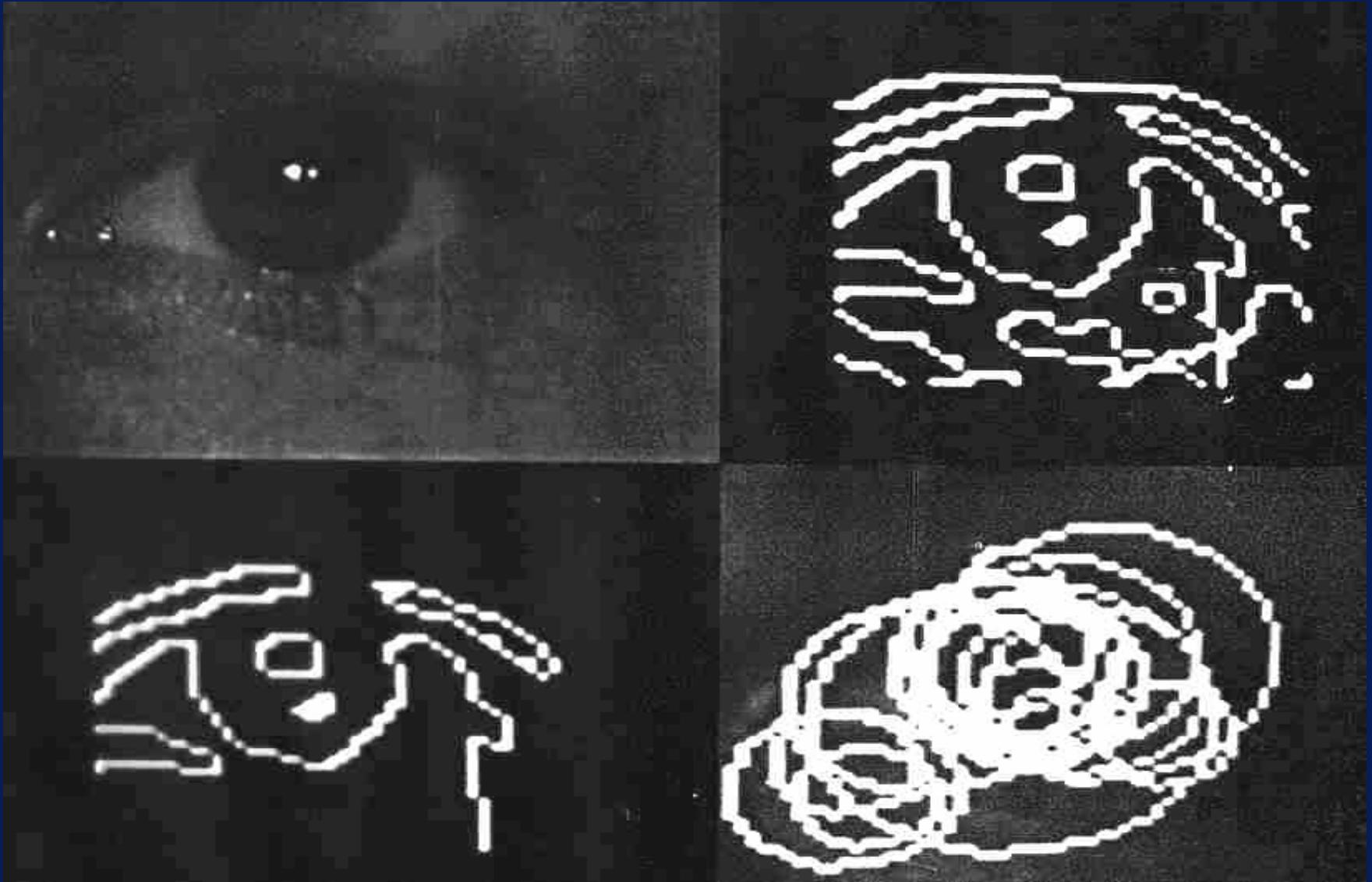
$$(x - a)^2 + (y - b)^2 = r^2$$

- where  $(a, b)$  are the co-ordinates of the centre of the circle and  $r$  is its radius.

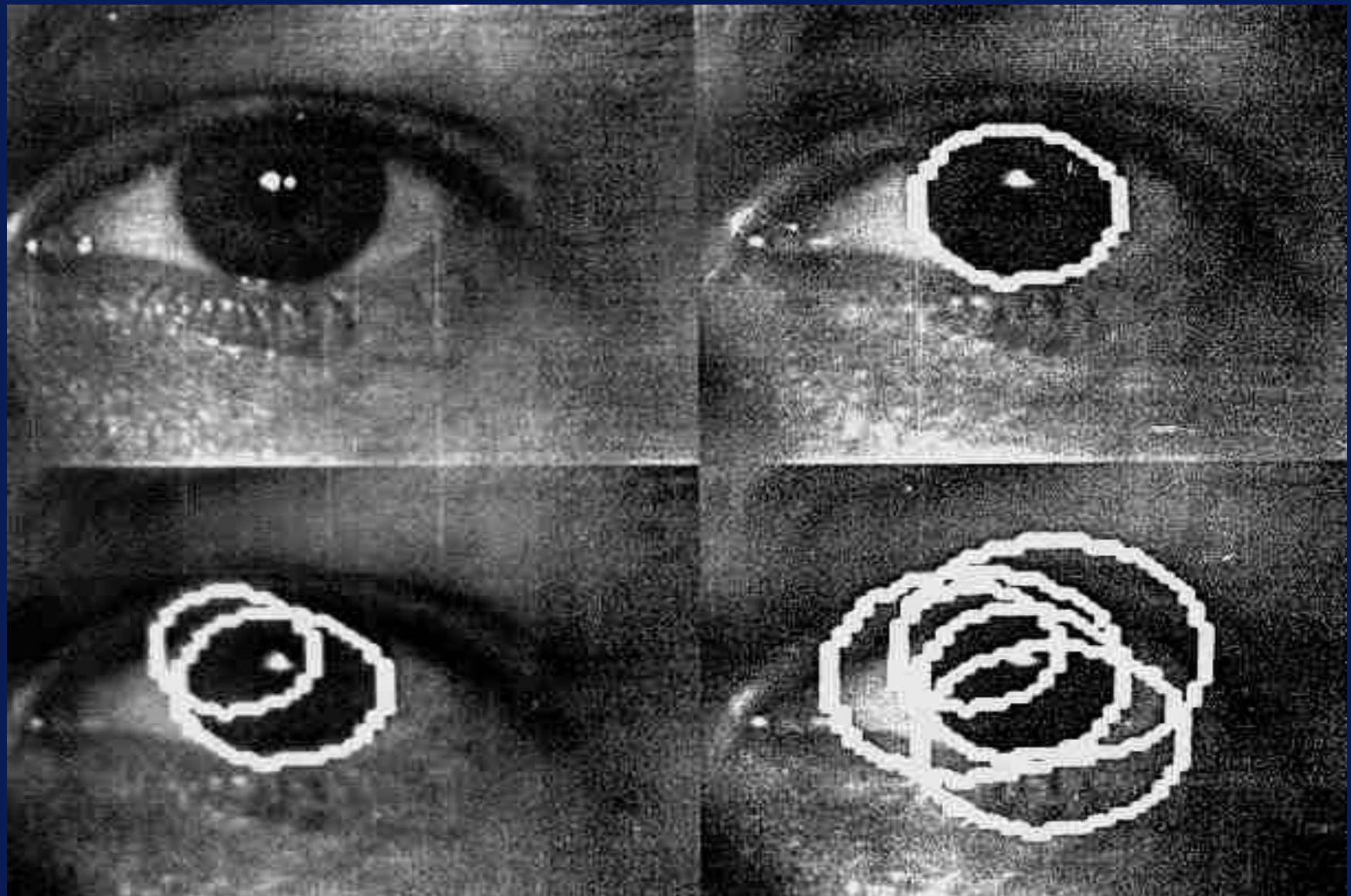
- In this case, we have three co-ordinates in the parameter space:  $a$ ,  $b$  and  $r$ . Hence, we require a 3D accumulator with an attendant increase in the computational complexity of the algorithm.



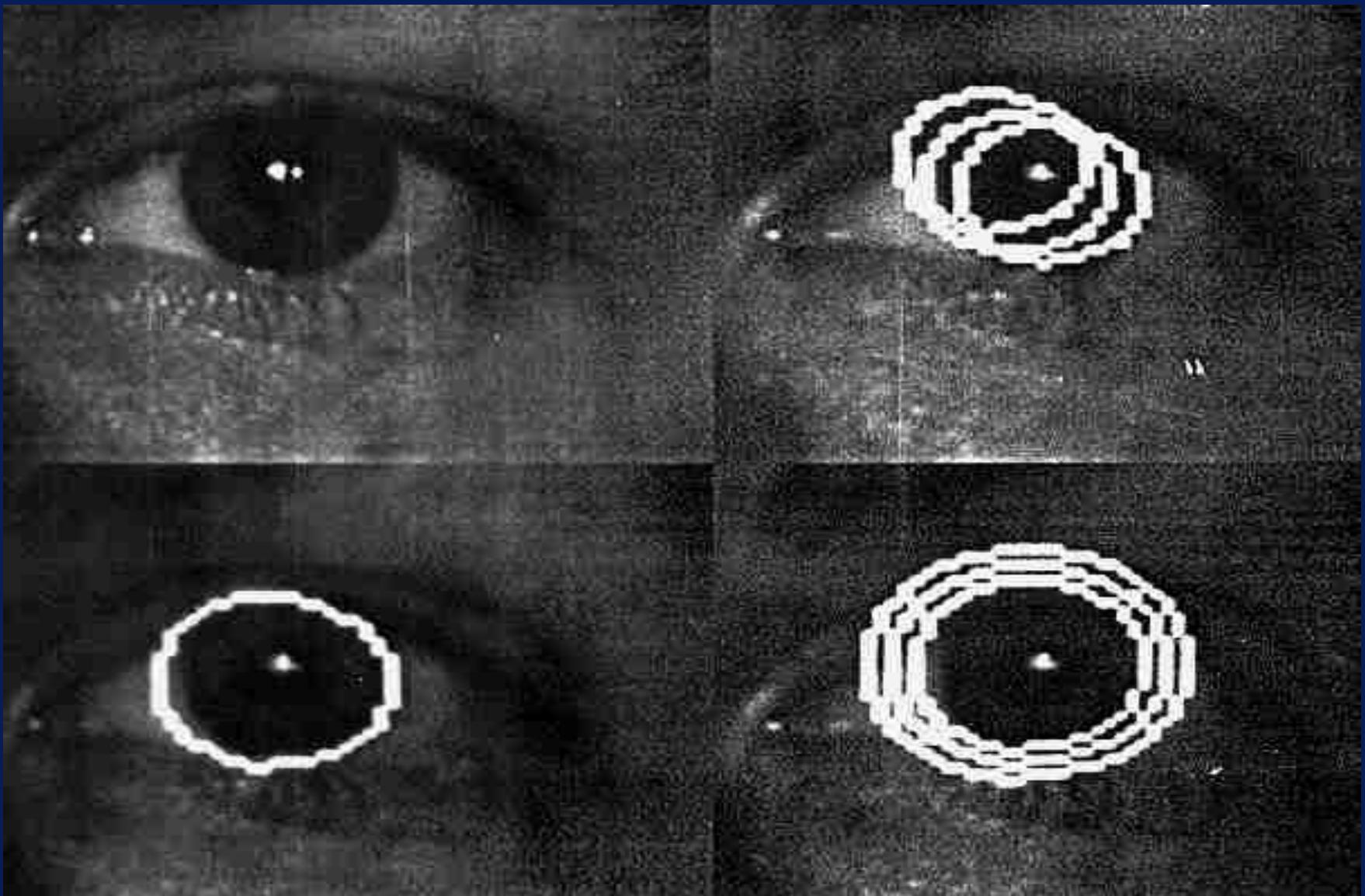
- *Hough Transform for*
- *detection of circular shapes.*



Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))



Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))

- One further point is worth noting :
  - the Hough Transform identifies the parameter of the curve (or line) which best fits the data (the set of edge points).
- However, the circles that are generated are complete circles and the lines are infinite.

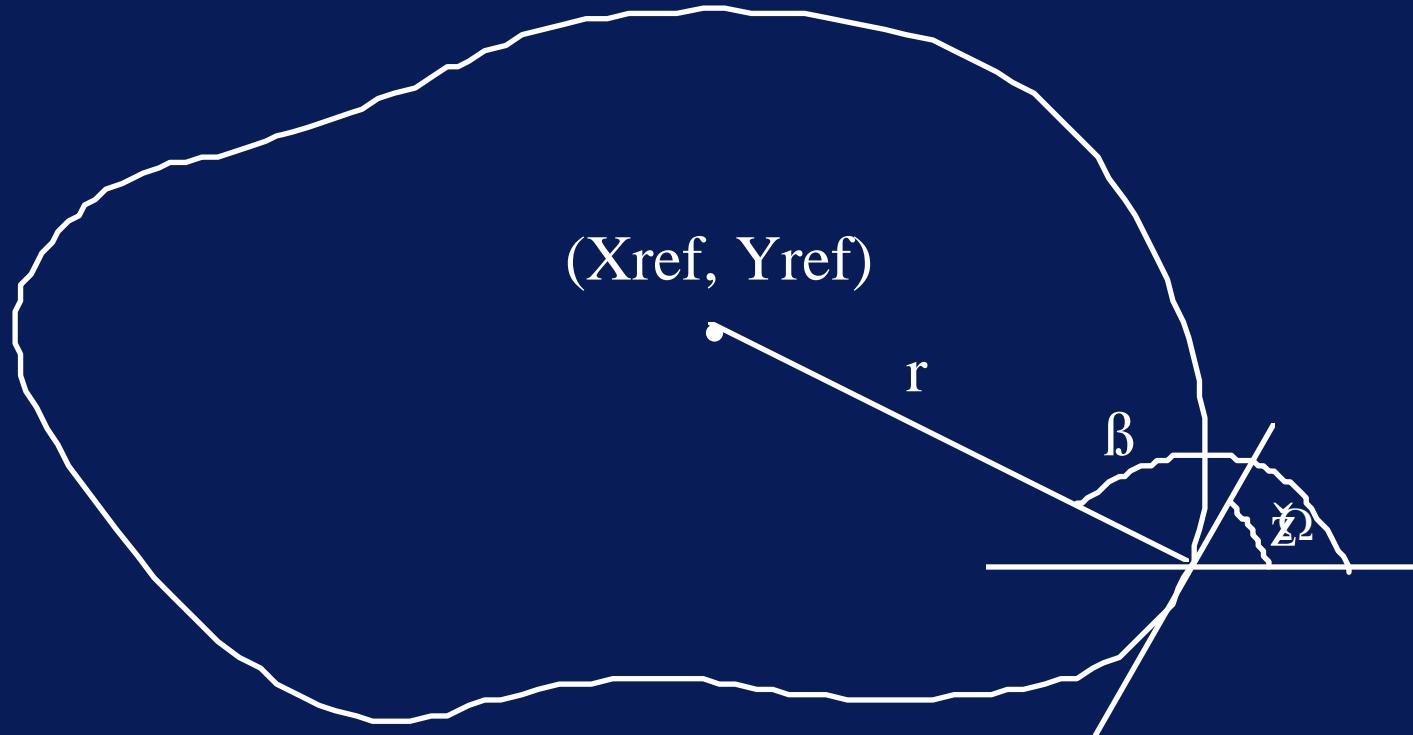
- If one wishes to identify the actual *line segments* or *curve segments* which generated these transform parameters, further image analysis will be required.

# The Generalised Hough Transform

- In the previous formulation of the classical Hough Transform, *we used the parametric equation of the shape to effect the transform image space to transform space.*

- In the case where the shape we wish to isolate does not have a simple analytic equation describing its boundary, we can still use a generalised form of the Hough transform.

- instead of using the parametric equation of the curve, we use a look-up table to define the relationship between the boundary and orientation, and the Hough parameters.
- the look-up table values must be computed during a training phase using a prototype shape.



- *Generalised Hough Transform*
- - *Definition of R-Table Components*

- Suppose we know the shape and orientation of the required object, *e.g.* see Figure 6.11. The first step is to select an arbitrary reference point
- $(x_{ref}, y_{ref})$  in the object.

- We now define the shape in terms of the distance and angle of lines from the boundary to this reference point.
- For all points of the boundary, we draw a line to the reference point.
- We then compute the orientation of the boundary,  $\Omega_i$  say,

- make a note in the look-up table of the distance and direction from the boundary point to the reference point at a location in the look-up table indexed by the boundary orientation  $\Omega_i$ .

- Since it is probable that there will be more than one occurrence of a particular orientation as we travel around the boundary, we have to make provision for more than one pair of distance and angle values.
- This look-up table is called an *R-Table*.

- The Hough transform space is now defined in terms of the possible positions of the shape in the image, *i.e.*, the possible ranges of  $x_{ref}$  and  $y_{ref}$  (instead of  $r$  and  $\phi$  in the case of the Hough transform for line detection).

- To perform the transform on an image we compute the point  $(x_{ref}, y_{ref})$  from the coordinates of the boundary point, the distance  $r$  and the angle  $\beta$ :
  - $x_{ref} = x + r \cos \beta$
  - $y_{ref} = y + r \sin \beta$

- The question is : what values of  $r$  and  $\beta$  do we use ?
- These are derived from the R-Table by computing the boundary orientation  $\Omega$  at that point and using it as an index into the R-table, reading off all the  $(r, \beta)$  pairs.

- The accumulator array cell  $(x_{ref}, y_{ref})$  is then incremented. We reiterate this process for all edge points in the image.
- As before, we infer the presence of the shape by identifying local maxima in the accumulator array.

- There is just one problem : we have assumed that we know the orientation of the shape.
- If this is not the case, we have to extend the accumulator by incorporating an extra parameter  $\phi$  to take change in orientation into consideration.

- Thus, we now have a 3D accumulator indexed by  $(x_{ref}, y_{ref}, \phi)$  and we compute :
  - $x_{ref} = x + r \cos (\beta + \phi)$
  - $y_{ref} = y + r \sin (\beta + \phi)$

- /\* Pseudo-code for Hough Transform \*/
  - Train the Shape by building the R-Table
    - For all points on the boundary
    - Compute orientation  $\Omega$  (gradient direction + 90°),
    - Compute  $r$  and  $\beta$ ,
    - Add an  $(r, \beta)$  entry into the R-table at a location indexed by  $\Omega$ .

- Quantise the Hough transform :  
Identify maximum and minimum values of  $x_{ref}$ ,  $y_{ref}$  and  $\phi$  and identify the total number of  $x_{ref}$ ,  $y_{ref}$  and  $\phi$  values.
- Generate an accumulator array  $A(x_{ref}, y_{ref}, \phi)$ ;  
set all values to 0.

- For all edge points  $(x_i, y_i)$  in the image
  - Do
    - Compute the orientation  $\Omega$  (gradient direction +  $90^\circ$ )
    - Compute possible reference points  $x_{ref}, y_{ref}$
    - For each table entry, indexed by  $\Omega$ 
      - For each possible shape orientation  $\phi$ 
        - Compute  $x_{ref} = x_i + r \cos (\beta + \phi)$
        - $y_{ref} = y_i + r \cos (\beta + \phi)$
      - Increment  $A(x_{ref}, y_{ref}, \phi)$

- For all cells in the accumulator array

Do

Search for maximum values;

the co-ordinates  $x_{ref}$ ,  $Y_{ref}$  and  $\phi$  give the position and orientation of the shape in the image.

# Histogram Analysis

- The grey-level histogram of an image often contains sufficient information to allow analysis of the image content and, in particular, to discriminate between objects and to distinguish objects with defects.

- *It has the distinct advantage that it is not necessary to segment the image first and it is not dependent on the location of the object in the image.*

- The analysis is based exclusively on the visual appearance of the scene or image as a whole. As a simple example, consider the case where one is inspecting a bright shiny object, *i.e.* one which exhibits specular reflectivity.

- In bright field illumination, where the camera and the light source are approximately aligned, a great deal of light will be reflected and the histogram of the imaged object will be biased toward the bright end of the scale.
-

- Blemishes, or unwanted surface distortions, will tend to diffuse the light and less bright specular reflections will be imaged.
- In this case, the histogram will be biased more toward the dark end of the spectrum.

- In effect, the two cases can be distinguished by considering the distribution of grey-levels in the image, *i.e.* by analysis of the histogram.

- There are two ways to consider histogram analysis :
  1. By extracting features which are descriptive of the shape of the histogram.
  2. By matching two histogram signatures.

- In the former case, discrimination can be achieved using the classification techniques we discussed in section 6.3, whereas in the latter case, the template matching paradigm of section 6.2 is more appropriate.

- The following (statistical) features are frequently used as a means of describing the shape of histograms.

- Mean :

$$\bar{b} = \sum_{b=0}^{L-1} b P(b)$$

- Variance :

$$\sigma_b^2 = \sum_{b=0}^{L-1} (b - \bar{b})^2 P(b)$$

- Skewness :

$$b_s = \frac{1}{\sigma_b^3} \sum_{b=0}^{L-1} (b - \bar{b})^3 P(b)$$

- Kurtosis :

$$b_k = \frac{1}{\sigma_b^4} \sum_{b=0}^{L-1} (b - \bar{b})^4 P(b) - 3$$

- Energy :

$$b_N = \sum_{b=0}^{L-1} [P(b)]^2$$

- These features are based on ‘normalised’ histograms,  $P(b)$ , defined as :

$$P(b) = \frac{N(b)}{M}$$

- where  $M$  represents the total number of pixels in the image,  $N(b)$  is a conventional histogram ...

- (*i.e.* a function which represents the number of pixels of a given grey level  $b$ ).
- Note that  $L$  is the number of grey-levels in the grey-scale.

# An Overview of Techniques for Shape Description

- It is very desirable that a vision system should be able to deal with random 3D presentation of objects
  - it is also, in general, beyond the current capabilities of most commercial systems.

- if, however, the objects are (to an approximation) two-dimensional, the problem is more tractable, requiring the description and classification of planar shapes which may, or may not, be partially occluded.

- The issue of shape description and representation is very far from being resolved;
  - The difficulty with shape is that it is not clear what exactly we are trying to abstract or represent: it seems to depend strongly on the use to which we intend putting the resulting representation (see, for example, [Brady 1983]).

- Practitioners in some areas (*e.g.* mathematical morphology) go a little further and put forward the idea that shape is not an objective entity, in the sense of having an independent existence and depends both on the observed and the observer.

- *Shape as an issue in Computer Vision is both ubiquitous and ill-defined; the solution to the riddle of a general understanding of shape will probably shed light on many other areas in advanced computer vision, if only because it epitomises the stonewall with which we are currently presented when we wish to develop truly adaptive robust visual systems.*

# A Taxonomy of Shape Descriptors

- Pavlidis (1978) classifies shape descriptors
  - according to whether they are based on external or internal properties of the shape
  - according to whether they are based on scalar transform techniques or on space domain techniques.

- *External Descriptors* are typically concerned with properties based on the boundary of the shape.
- *Internal Descriptors* take cognisance of the complete region comprising the shape.

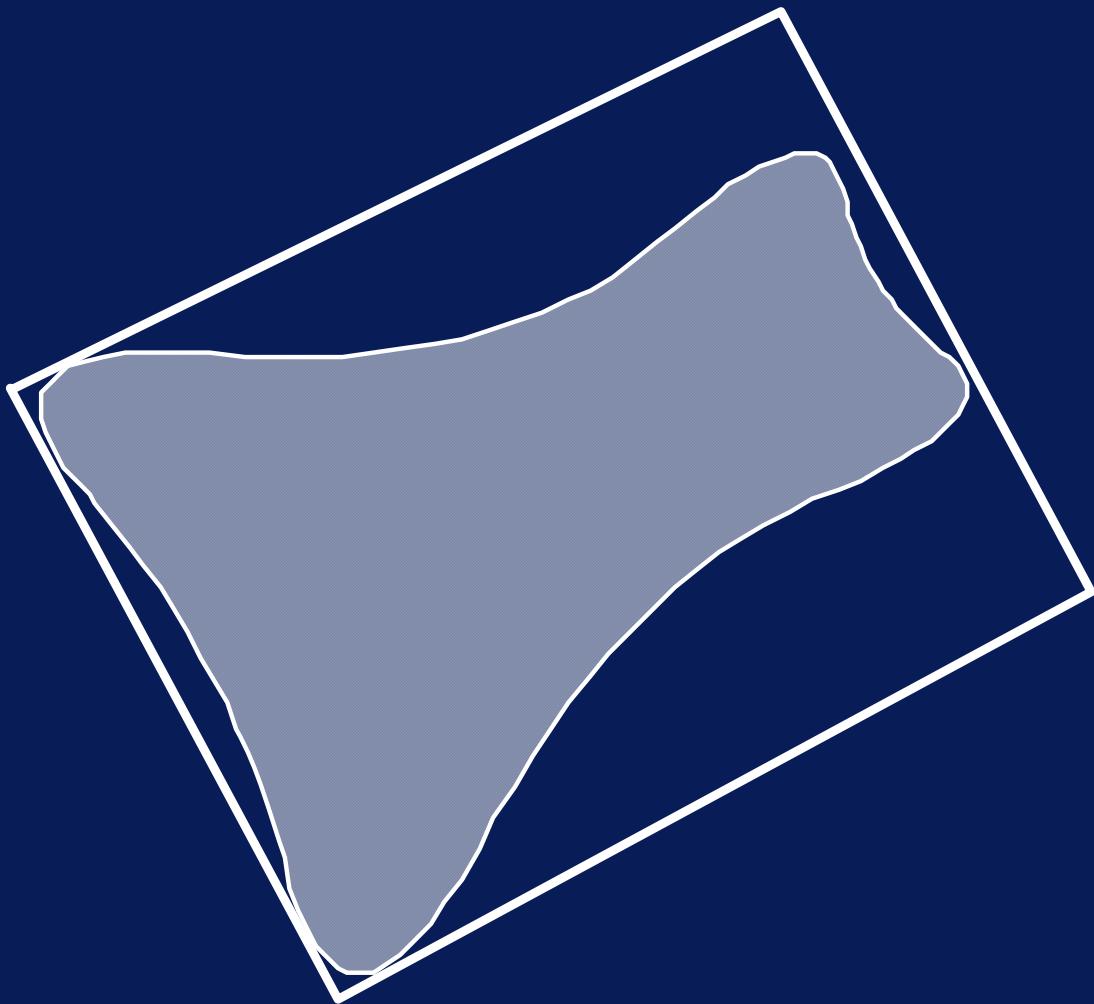
- Scalar transform techniques generate vectors of scalar features.
- Space domain techniques generate spatial or relational descriptions of certain characteristics of features of the shape.

- Thus, Pavlidis identifies four distinct types of shape descriptor :
  1. External Scalar Transform techniques utilising features of the shape boundary.
  2. Internal Scalar Transform techniques utilising features of the shape region.

3. External Space Domain techniques utilising the spatial organisation of the shape boundary.
4. Internal Space Domain techniques utilising the spatial organisation of the shape.

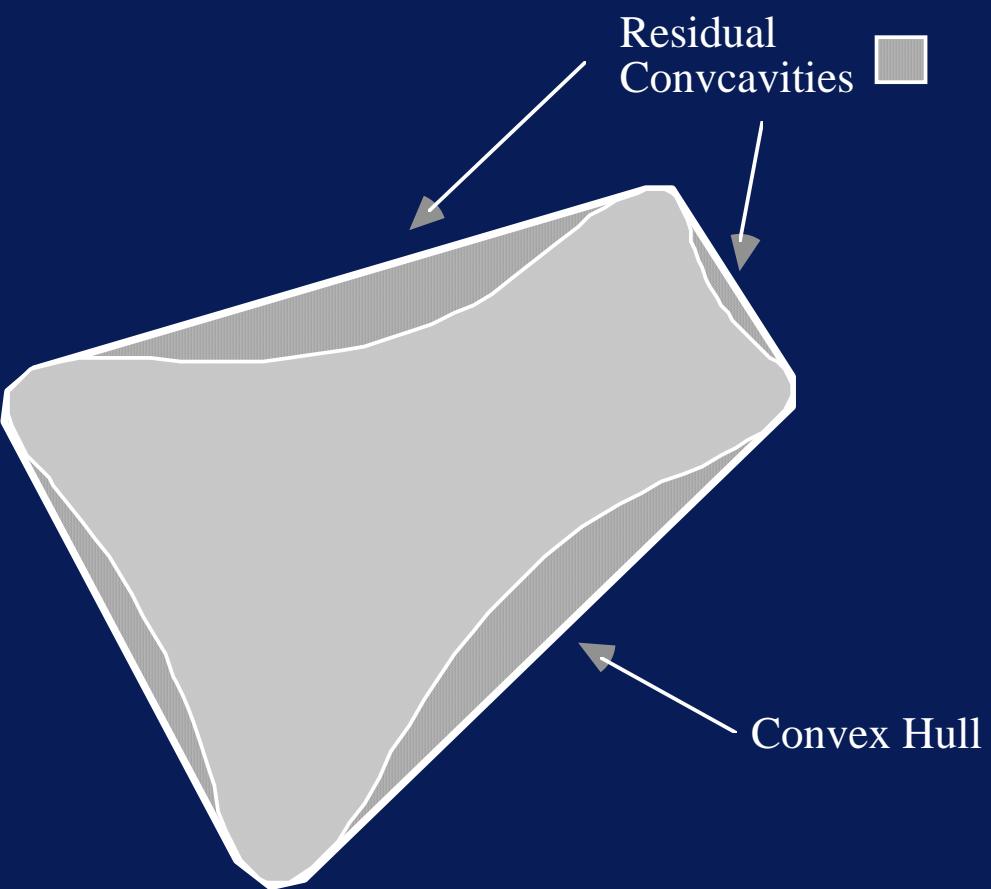
## External Scalar Transform Descriptors: Features of the Boundary

- External scalar transform descriptors are based on scalar features derived from the boundary of an object. Simple examples of such features include :
  - the perimeter length
  - the ratio of the major to minor axis of the minimal bounding rectangle of the shape (see Figure 7.1) (*Slide 12*).



- *Bounding Rectangle of a Simple 2D Shape.*

- the number and size of residual concavities lying within the bounds of the shape's convex hull (see Figure 7.2) (*Slide 14*).
- the ratio of the area of a shape to the square of the length of its perimeter ( $A/P^2$ ): this is a measure of circularity which is maximised by circular shapes.



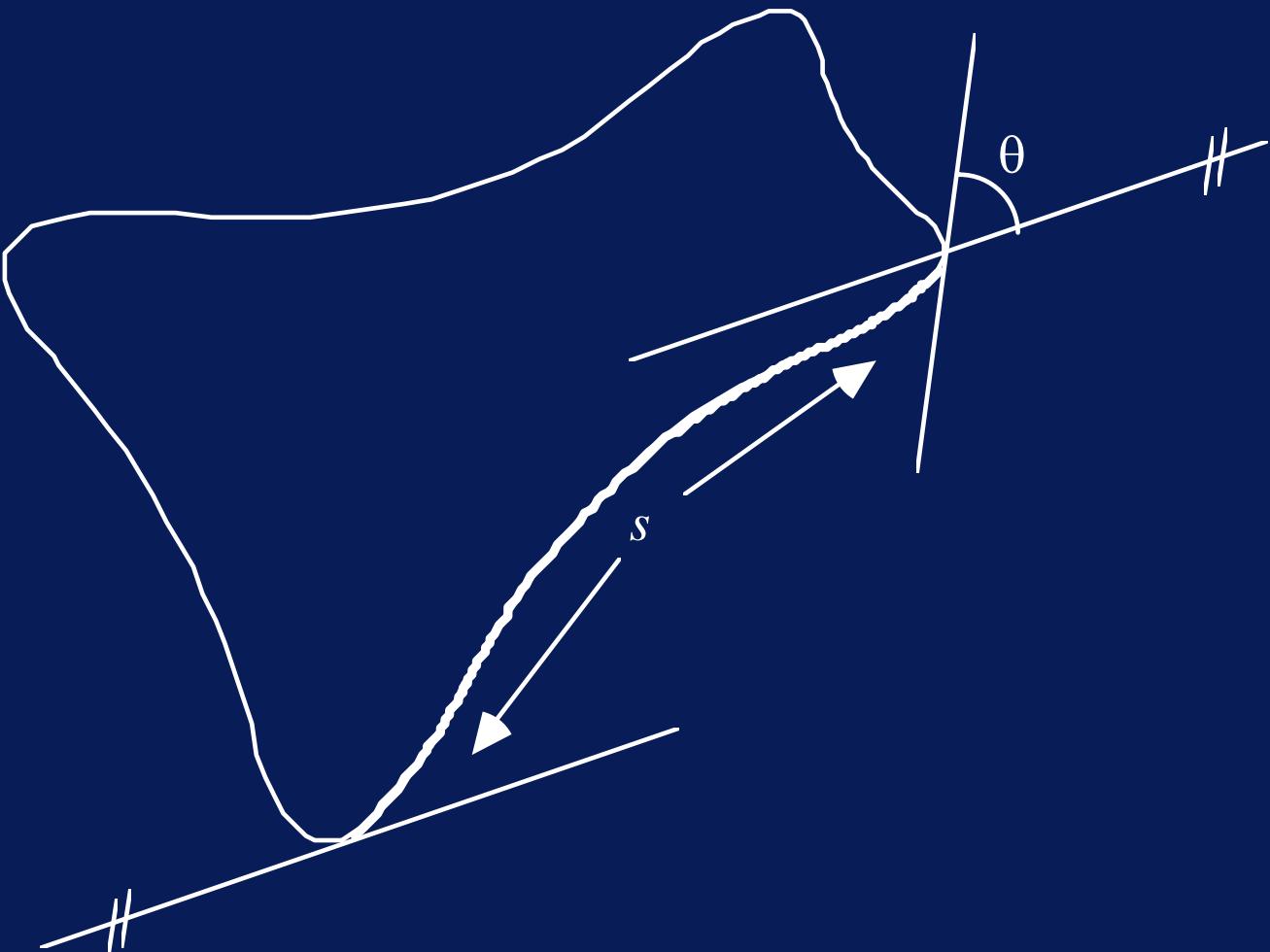
- *Residual concavities within the convex hull of a simple 2D shape*

- the ratio of the area of a shape to the area of the minimal bounding rectangle: this is a measure of rectangularity and is maximised for perfectly rectangular shapes.

- More sophisticated scalar transform techniques are often based on the Fourier series expansion of a periodic function derived from the boundary.

- For example, consider the shape depicted in Figure 7.3 (*Slide 19*).
  - The rotation  $\theta$  of the tangent at the boundary of the object will vary between 0 and  $2\pi$  radians as the boundary is traversed.

- $\theta$  will vary with the distance,  $s$ , around the perimeter and can be expressed as a function  $\theta(s)$ .
- If  $L$  is the length of the boundary of the shape,  $\theta(0) = 0$  and  $\theta(L) = -2\pi$ .
- Unfortunately, this is obviously not a periodic function and so we cannot express it in terms of a Fourier series expansion.



- *Rotation of tangent to a boundary of a shape.*

- However, an alternative formulation, suggested by Zahn and Roskies, can be :
- Let  $t = (2\pi/L)s$  (thus :  $0 \leq t \leq 2\pi$ ) and define a new function  $\phi(t)$  :

$$\phi(t) = \theta\left(\frac{Lt}{2\pi}\right) + t$$

- Now,  $\phi(0) = \phi(2\pi) = 0$ .

- This function is not dependent on the size, position or orientation of a shape and, hence, the low-order coefficients of its Fourier series expansion can be used as features for translation, rotation and scale invariant shape recognition.

- Unfortunately, it suffers from the disadvantage, common to all transform techniques, of difficulty in describing local shape information, *e.g.*, it would have difficulty in discriminating between two shapes where the only dissimilarity is a small notch in the perimeter.

## Internal Scalar Transform Descriptors : Features of the Region

- Internal scalar transform techniques generate shape descriptors based on the entire shape. One of the most popular is the ***Method of Moments***. The standard two-dimensional moments  $m_{uv}$  of an image intensity function  $g(x, y)$  are defined:

$$m_{uv} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) x^u y^v dx dy \quad u, v = 0, 1, 2, 3\dots$$

- which, in the discrete domain of digital images becomes:

$$m_{uv} = \sum_x \sum_y g(x, y) x^u y^v \quad u, v = 0, 1, 2, 3\dots$$

- summed over the entire sub-image within which the shape lies.

- Unfortunately, these moments will vary for a given shape depending on where the shape is positioned, *i.e.*, they are computed on the basis of the absolute position of the shape.

- To overcome this, we can use the central moments :
- $m_{uv} = \sum \sum g(x, y)(x - \bar{x})^u(y - \bar{y})^v$
- $u, v = 0, 1, 2, 3\dots$
- where  $\bar{x} = \frac{m_{10}}{m_{00}}$  and  $\bar{y} = \frac{m_{01}}{m_{00}}$

- That is,  $\bar{x}$  and  $\bar{y}$  are the co-ordinates of the centroid of the shape. Thus, these moments take the centroid of a shape as their reference point and hence are position invariant.

- Assuming that the intensity function
- $g(x, y)$  has a value of one everywhere in the object (*i.e.* one is dealing with a simple segmented binary image), the computation of  $m_{00}$  is simply a summation yielding the total number of pixels within the shape.

- If one also assumes that a pixel is one unit area, then  $m_{00}$  is equivalent to the area of the shape.

- Similarly,  $m_{10}$  is effectively the summation of all the  $x$  co-ordinates of pixels in the shape and  $m_{01}$  is the summation of all the  $y$  co-ordinates of pixels in the shape; hence
  - $m_{10}/m_{00}$  is the average  $x$  co-ordinate and
  - $m_{01}/m_{00}$  is the average  $y$  co-ordinate,
  - *i.e. the co-ordinates of the centroid.*

- The central moments up to order three are  
...  
...

$$\mu_{00} = m_{00}$$

$$\mu_{10} = 0$$

$$\mu_{01} = 0$$

$$\mu_{20} = m_{20} - \bar{x}m_{10}$$

$$\mu_{02} = m_{02} - \bar{y}m_{01}$$

$$\mu_{11} = m_{11} - \bar{y}m_{10}$$

$$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2m_{10}$$

$$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2m_{01}$$

$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{01} + 2\bar{y}^2m_{10}$$

$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2m_{01}$$

- These central moments can be normalised, defining a set of normalised central moments  $\eta_{ij}$  :

$$\eta_{ij} = \frac{\mu_{ij}}{(\mu_{00})^k}$$

- where  $k = ((i + j) / 2) + 1$  /  $i + j \geq 2$

- However, moment invariants (linear combinations of the normalised central moments) are more frequently used for shape description as they generate values which are invariant with position, orientation and scale changes.

- These seven invariant moments are defined as ...

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} + 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\begin{aligned}\phi_5 = & (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta_{12}) \left\{ (\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2 \right\} + \\ & (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \left\{ 3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 \right\}\end{aligned}$$

$$\begin{aligned}\phi_6 = & (\eta_{20} + 3\eta_{12}) \left\{ (\eta_{30} + \eta_{12}) - (\eta_{21} + \eta_{03})^2 \right\} \\ & + 4\eta_{11}(\eta_{30} - \eta_{12})(\eta_{21} + \eta_{03}) \\ \phi_6 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) \left\{ (\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2 \right\} \\ & - (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) \left\{ 3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 \right\}\end{aligned}$$

- The logarithm of  $\phi_1$  to  $\phi_7$  is normally used to reduce the dynamic range of the values when using these moment invariants as features in a feature classification (*i.e.* shape recognition) scheme.

- Shape descriptors based on moment invariants convey significant information for simple objects but fail to do so for complicated ones.

- Since we are discussing internal scalar transform descriptors, it would seem that these moment invariants can only be generated from the entire region.

- However, they can also be generated from the boundary of the object by exploiting Stokes' theorem or Green's theorem, both of which relate the integral over an area to an integral around its boundary.

- We will return to this in the next section on external space domain shape descriptors when describing the BCC (Boundary Chain Code).

## External Space Domain Descriptors: Spatial Organisation of the Boundary

- One popular technique is the use of syntactic descriptors of boundary primitives, *e.g.*, short curves, line segments and corners.

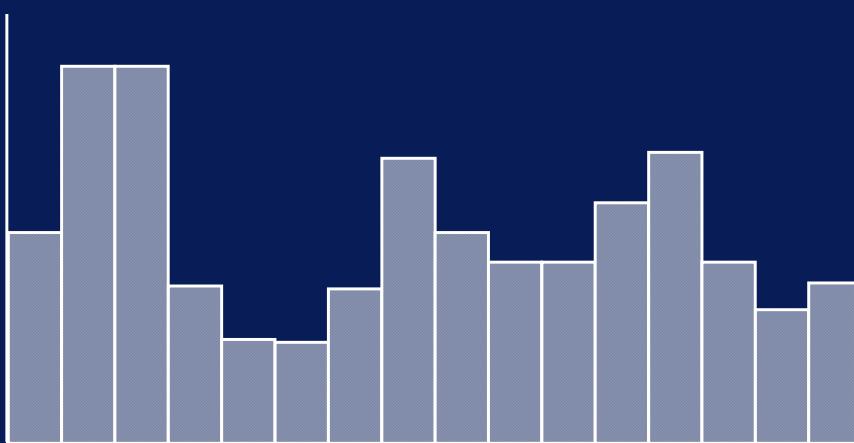
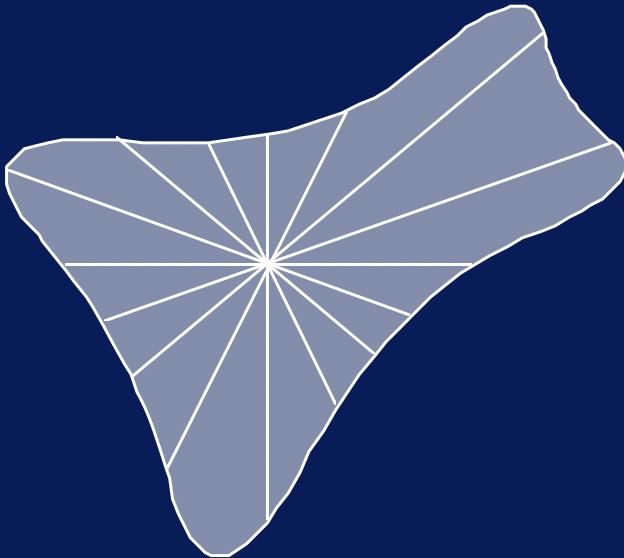
- Thus, the shape descriptor is a list or string of primitive shapes
  - the formation of the list or string must adhere to given rules: the shape syntax or grammar.

- Syntactic patterns are recognised by parsing the string of primitive patterns in a manner somewhat similar to the way a compiler parses a computer program to check its syntax.

- Descriptors based on external space domain are
  - generally efficient,
  - require minimal storage requirements,

- in the case of the more sophisticated syntactic techniques expounded by Fu et al. are based on well-developed general methodologies such as the theory of formal languages.

- A polar radii signature, encoding the distance from the shape centroid to the shape boundary as a function of ray angle, is another much simpler external space domain descriptor (see Figure 7.3)



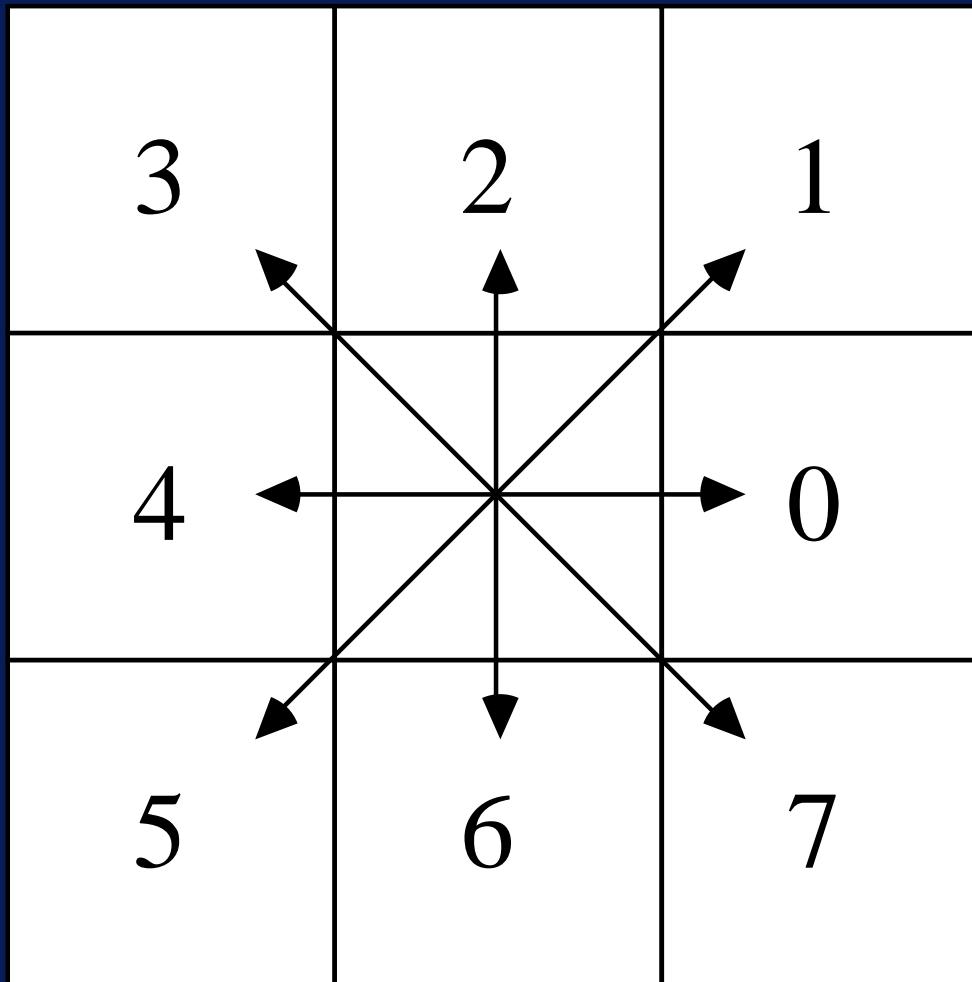
Copyright © 2007 David Vernon ([www.vernon.eu](http://www.vernon.eu))

- In the case of the simple external space domain descriptors, *e.g.* radii signatures, the recognition strategy is based on correlation or matching of template signatures, rather than on sophisticated parsing techniques.

- One of the most common external space domain descriptors is the *Boundary Chain Code (BCC)*.
- The BCC is more useful for shape representation rather than shape recognition. The BCC encodes piecewise linear curves as a sequence of straight-line segments called *Links*.

- A link  $a_i$  is a directed straight line segment of length  $T(\sqrt{2})^P$  and of angle  $a_i * 45^\circ$  (referenced to the X-axis of a right-handed Cartesian coordinate system).
- $T$  is the grid spacing and is normally set equal to unity.

- $a_i$  may be any integer in the range 0 to 7 and represents the (coarsely quantised) direction of the link.
- $p$  is the modulo two value of  $a_i$ ; i.e.,  $p = 0$  if  $a_i$  is even and  $p = 1$  if  $a_i$  is odd. Thus, the link length in directions 1, 3, 5 and 7 is equal to  $\sqrt{2}$  and is equal to 1 in directions 0, 2, 4 and 6.



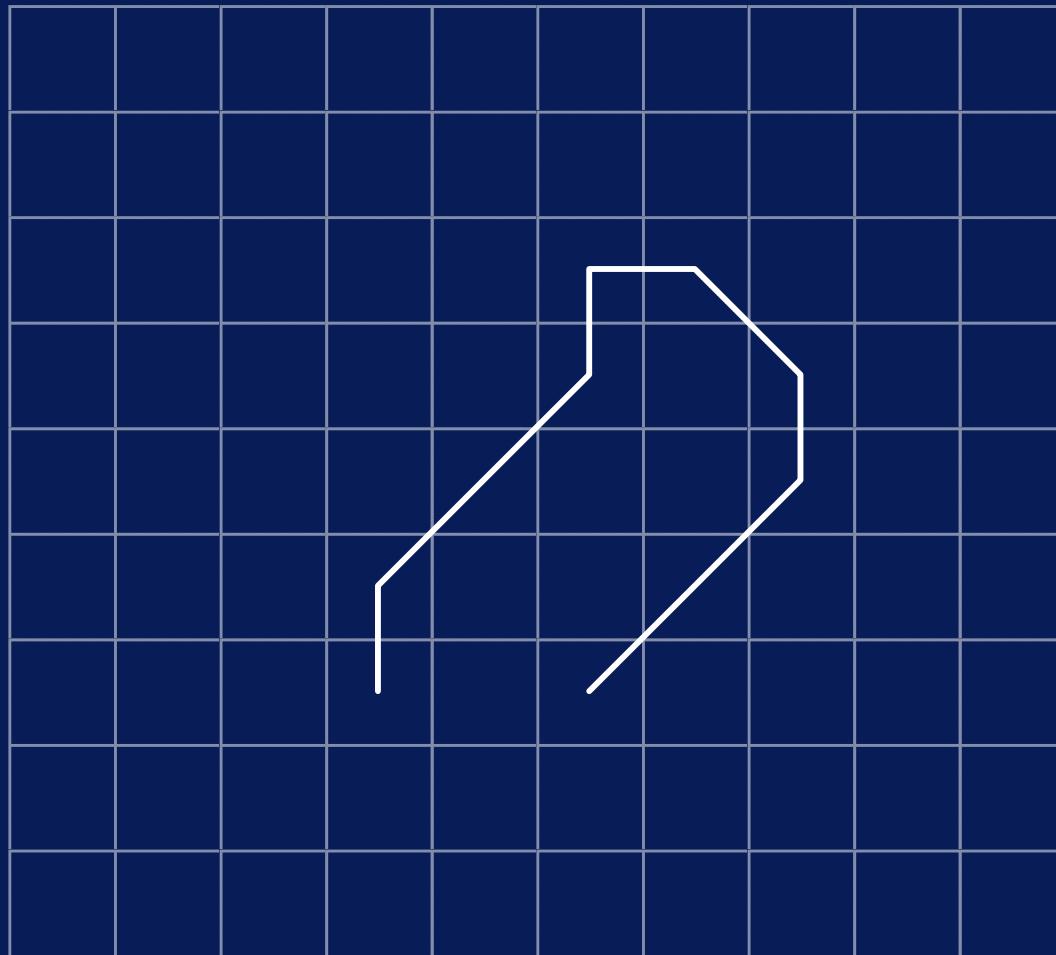
- *Boundary Chain Code (BCC) Directions*

- A BCC is a *non-uniformly sampled* function
  - the distance between the sample points along the boundary may be either 1 or  $\sqrt{2}$  depending on whether the neighbouring boundary points are horizontal/vertical neighbours or diagonal neighbours, respectively.

- Thus, a BCC is dependent on the orientation of the object boundary on two distinct bases :
  - a. Each link encodes the absolute direction of the boundary at that point.
  - b. The link length (1 or  $\sqrt{2}$ ) varies with the boundary direction.

- Any shape descriptor which is derived from this non-uniformly sampled BCC is inherently sensitive to changes in orientation.

- To alleviate this rotational variance it is necessary to remove the dependency on link length, ensuring that the link lengths of the BCC are all equal, specifically by re-sampling the BCC in uniformly-spaced intervals.



BCC: 2 1 1 2 0 7 6 5 5

- *A BCC representation of a simple shape*

- Note that this re-sampling technique will generate non-integer co-ordinate values for the pixels they represent;
  - the actual image pixel values can be obtained for shape reconstruction simply by rounding the real-valued co-ordinates.

- The fact that the node co-ordinate values are non-integer is not important if the descriptor is being used for the purposes of shape recognition since
- one is interested only in correspondence between link values (or the corresponding transformed property) and not the absolute position that they might represent in the image.

# An Algorithm for Re-sampling Boundary Chain Codes

- Let  $NUS\_BCC$  and  $US\_BCC$  represent the non-uniformly sampled and uniformly sampled BCCs, respectively.
- Let a point given by a non-uniformly sampled BCC be represented by
  - $(nus\_x, nus\_y)$ .

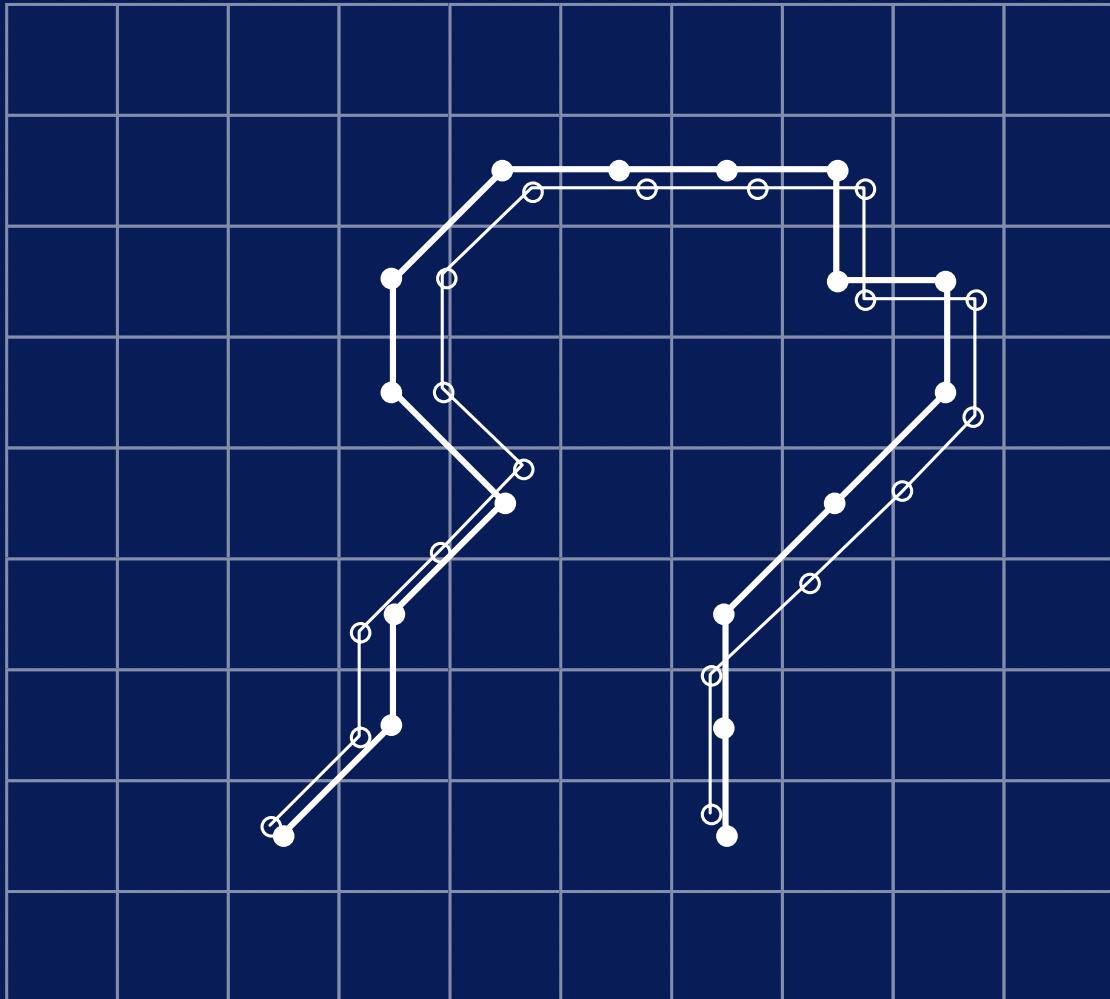
- Let a point given by a uniformly sampled BCC be represented by  $(us_x, us_y)$ .
- $NUS\_BCC$  and  $US\_BCC$  start at the same point on the contour :
- $(nus_x, nus_y) = (us_x, us_y)$ , initially.

```
.WHILE there are more NUS_BCC links to be
resampled DO
 • Generate the next NUS_BCC points:
 • (nus_x, nus_y)

./* resample */
```

- REPEAT
- /\* generate new *US\_BCC* link and append to  
                                                        *US\_BCC* \*/
- Generate three candidate uniformly sampled  
      points : (*us\_x1*, *us\_y1*), (*us\_x2*, *us\_y2*),  
              (*us\_x2*, *us\_y3*) at a distance of 1 unit from  
              the current uniformly sampled point
- (*us\_x*, *us\_y*) in directions corresponding to  
      the *NUS\_BCC* link direction, ±1.

- Test all three points and choose the point which is closest to (*nus\_x*, *nu\_y*) (using the Euclidean distance metric)
- Reassign the current (*us\_x*, *us\_y*) to be this point
- Append to the *US\_BCC* a link with a direction corresponding to this chosen point
- UNTIL  $|us_x - nus_x| < 0.5$  AND  $|us_y - nus_y| < 0.5$
- /\* (*us\_x*, *us\_y*) now lies within the bounds of the grid pixel given by (*nus\_x*, *nus\_y*) \*/



Original BCC



Resampled BCC

- As mentioned above, the BCC is most useful as a method for the representation of shapes and recognition is normally based upon other descriptors derived from the BCC.

- For example, the moment shape descriptors discussed previously can be generated from the boundary points given by a BCC.

$$m_{00} = \frac{1}{2} \sum_{i=1} A_i$$

$$m_{11} = \frac{1}{4} \sum_{i=1}^n A_i \left( x_i y_i - \frac{1}{2} x_i \Delta y_i - \frac{1}{2} y_i \Delta x_i + \frac{1}{3} \Delta x_i \Delta y_i \right)$$

$$m_{02} = \frac{1}{4} \sum_{i=1}^n A_i \left( y_i^2 - y_i \Delta y_i + \frac{1}{3} \Delta y_i^2 \right)$$

$$m_{10} = \frac{1}{3} \sum_{i=1}^n A_i \left( y_i - \frac{1}{2} \Delta y_i \right)$$

$$m_{01} = \frac{1}{3} \sum_{i=1}^n A_i \left( x_i - \frac{1}{2} \Delta x_i \right)$$

$$m_{20} = \frac{1}{4} \sum_{i=1}^n A_i \left( x_i^2 - x_i \Delta x_i + \frac{1}{3} \Delta x_i^2 \right)$$

- ... where  $x_{i-1}$  and  $y_{i-1}$  are the co-ordinates of a point on the perimeter of the shape and  $x_i$  and  $y_i$  are the co-ordinates of the subsequent point on the perimeter, as given by the BCC.
-

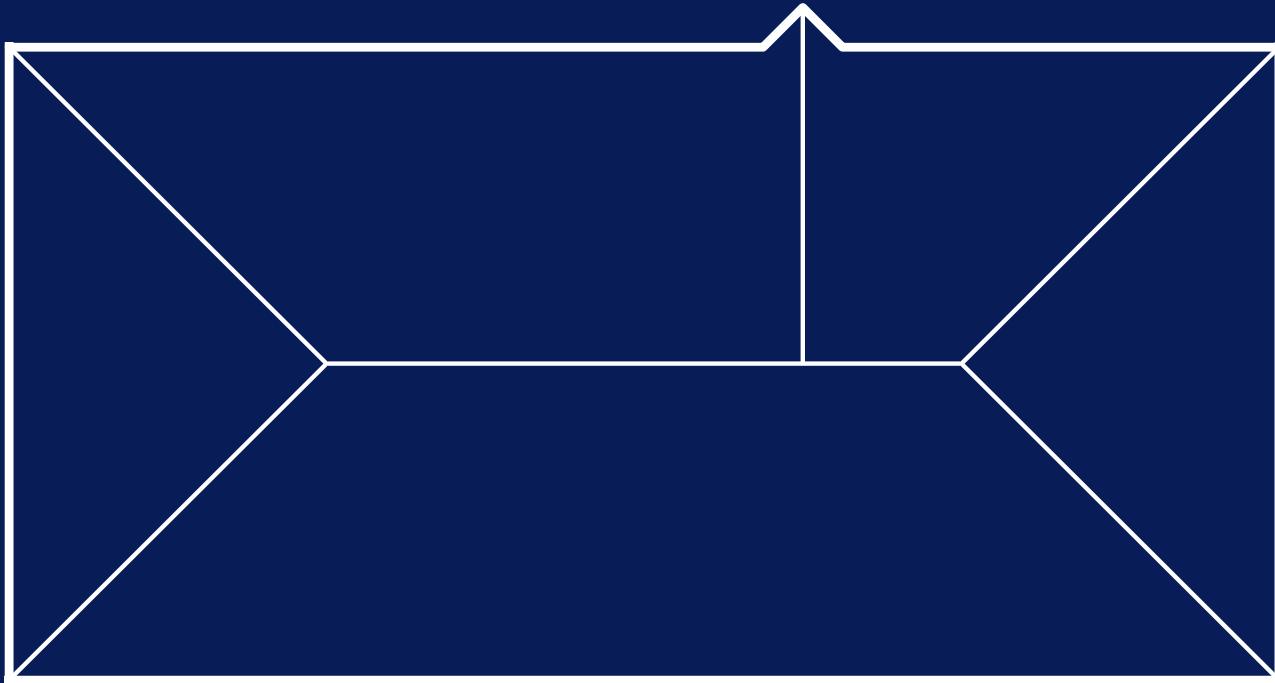
- $\Delta x_i$  is defined to be  $(x_i - x_{i-1})$ ,  $\Delta y_i$  is defined to be  $(y_i - y_{i-1})$  and  $A_i$  is defined to be
  - $(x_i \Delta y_i - y_i \Delta x_i)$ .  $n$  is the number of points on the boundary (*i.e.* the number of BCC links).

# Internal Space Domain Descriptors : Spatial Organisation of the Region

- Internal space domain techniques comprise descriptors which utilize structural or relational properties derived from the complete shape.
- The Medial Axis Transform (MAT)
  - a skeletal line-drawing from a two-dimensional shape.

- A point in the shape is on the medial axis if and only if it is the centre of a circle which is a tangent to the shape boundary at two non-adjacent points.
- Each point on the medial axis has a value associated with it which indicates the radius of this circle.

- This represents the minimum distance to the boundary from that point and thus facilitates the reconstruction of the object.



- *Medial Axis Transform (MAT) of a rectangular shape with a “bump”*

- There are various methods for generating the medial axis, the most intuitive of which is one that is often referred to as the *Prairie Fire Technique*;
  - setting fire to the boundary of a dry grassy field,
  - letting the flame burn inwards.
  - The points at which the flame fronts meet are on the medial axis.

- The MAT is sensitive to local distortions of the contour and small deviations can give to extraneous skeletal lines.

- Other descriptors can be derived using integral geometry:
  - an object shape can be intersected by a number of chords in different directions,
  - the locations and the length of the intersection can be used in various ways as a shape descriptor.

- One example of this type of descriptor is the *Normal Contour Distance (NCD)* shape descriptor
  - a one-dimensional signature,

- each signature value represents an estimate of the distance from a point on an object's boundary, with a local orientation of  $m$ , to the opposing boundary point which lies on a path whose direction is normal to  $m$ .
- The NCD signature is evaluated over the length of the available boundary.

- The NCD signature does not require knowledge of the complete boundary and can be used for recognition of partially-occluded objects.

- Note that, if the contour represents a partial boundary, there may not be another boundary point which lies on a path which is normal to the contour and, hence, some segments of the NCD may be undefined.

## Smoothed Local Symmetries (SLS) Representation

- This is a sophisticated and interesting technique because it represents shape with both region-based descriptors and contour-based descriptors.

- As such, it is more than an internal space domain descriptor and does not require the entire shape to be visible for extraction of useful shape description primitives.

- An SLS representation has three components:
  1. A set of Spines which are the loci local symmetries of the boundary of the shape.

## 2. A description of the local shape of the boundary contour

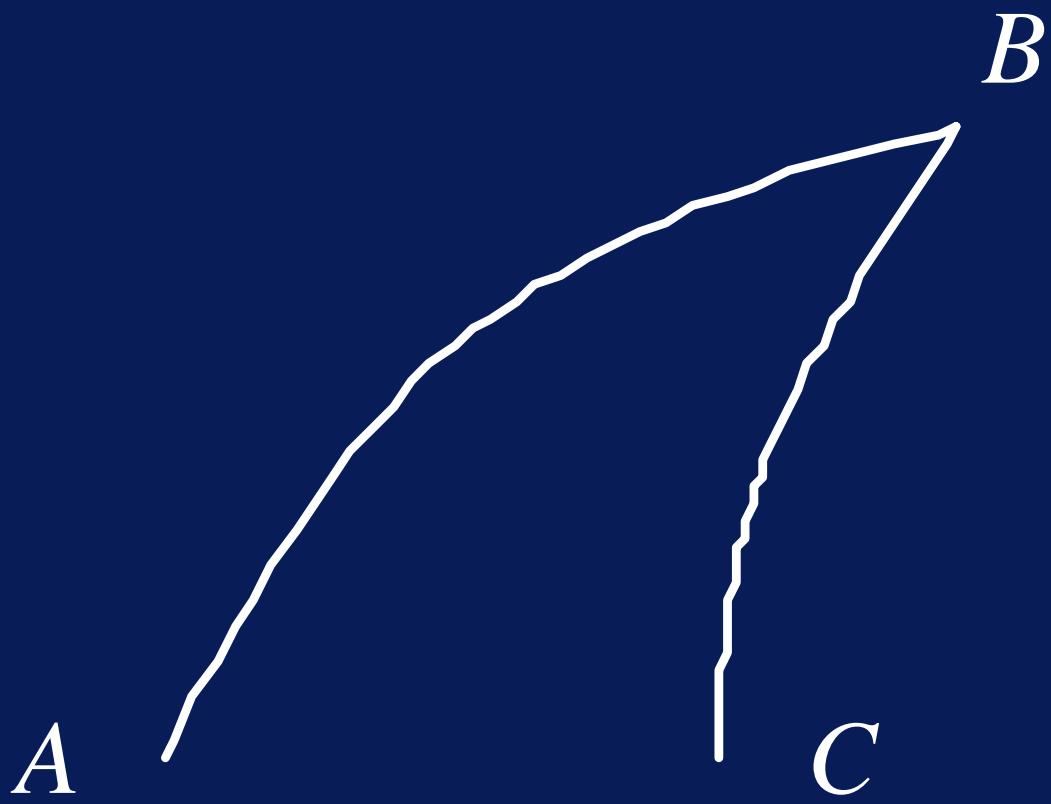
- » in terms of parametric curves (Brady uses circles and spirals),
- » in terms of primitives of curvature discontinuity.

These primitives effectively describe the manner in which the local contour curves are joined together to form the complete boundary shape.

3. A description of the region subtended by two locally symmetric curves.

This is effected by a small number of region labels (e.g. cup, sector, wedge and so on), each of which has several attributes associated with it.

They include the width of the region and the curvature of the associated spine.



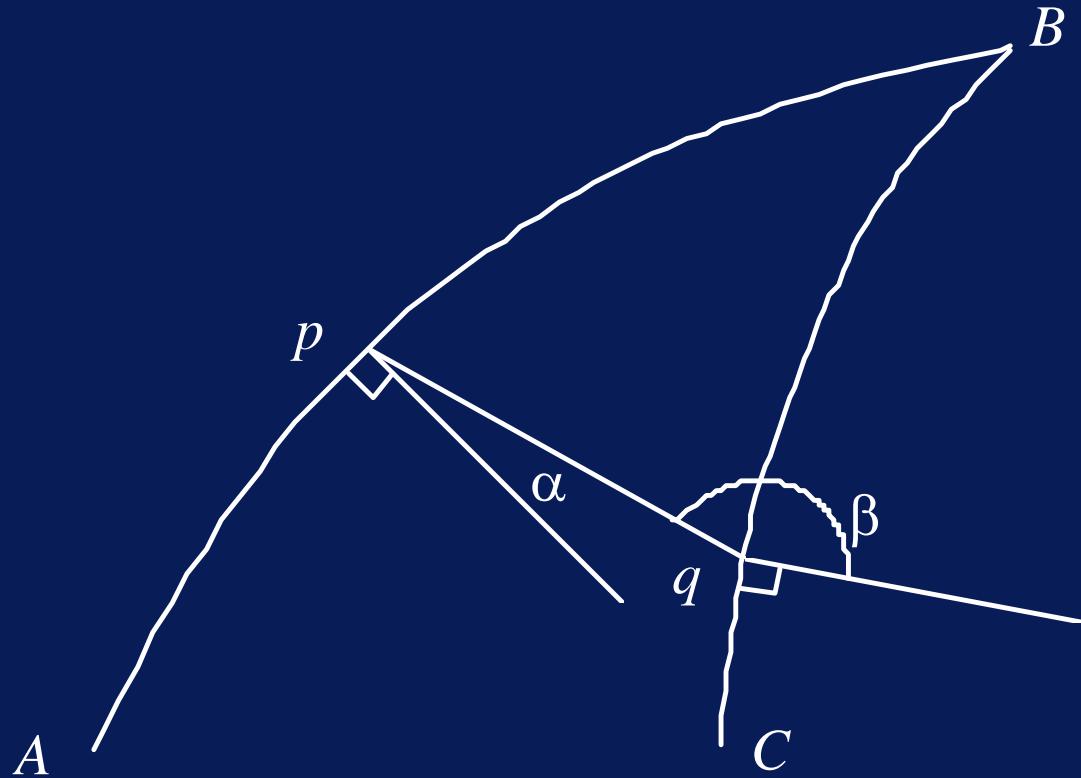
- (a) *Dorsal Fin of a Shark*

- For example, consider Figure 7.8(a) (*Slide 90*) which depicts the dorsal fin of a shark. The shape of the fin is defined by two curves  $AB$  and  $BC$ .
  - The curve  $AB$  is convex and is described by a circular arc.
  - The curve  $BC$  is concave and is also described by a circular arc.

- The junction between the two curves at  $B$  is an acute-angled *Corner*.
- *These three items form the contour-based description.*

- The descriptor of the region between the two curves  $AB$  and  $BC$  is labelled a *break* (the labelling that is used is based on the relative concavity/convexity of the two sides of a local symmetry)
  - the spine curves to the right.

- This raises the issue of how the spine is identified. Referring to Figure 7.8(b) (*Slide 96*),
  - let  $\alpha$  be the angle subtended by the inward normal to the contour at a point  $p$  on the curve  $AB$  and
  - let  $\beta$  be the angle subtended by the outward normal to the contour at a point  $q$  on the curve  $BC$ .



- (b) *Geometry defining a*
- *Smoothed Local Symmetry (SLS)*

- A point  $q$  on the curve  $BC$  forms a local symmetry with the point  $p$  on the curve  $AB$  if the sum of their respective angles is equal to  $180^\circ$ , *i.e.* if  $\alpha + \beta = 180^\circ$ .
- Several such points  $q$  can exist for a given point  $p$ .

- The spine of the SLS is effectively formed by the set mid-points of chords  $pq$  which satisfy this condition for local symmetry.
- At this stage, we might remark on the similarity between the SLS and the MAT.

- However, the SLS differs in several important ways:
  - the local symmetry which is implicit in the MAT is made explicitly in the SLS;
  - global symmetries, such as that which lies between a long fork (see Figure 7.9) (*Slide 98*), are not made explicit by the MAT;

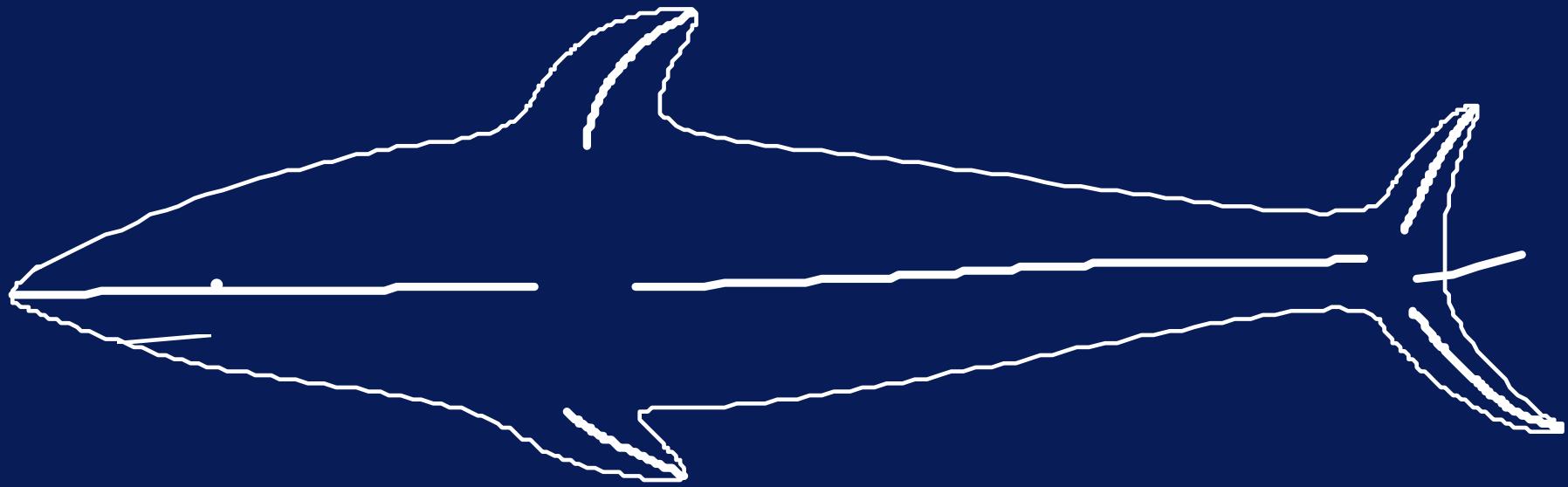


- *Global SLS of a Forked Object*

- the SLS includes the attributed contour and regions descriptors discussed above - the SLS is more than just a collection of spines.

- Finally, let us note that the SLS representation is also intended to be used as a descriptor of complex shapes, which can be viewed as a network of locally-symmetric parts (see Figure 7.8(c)) (*Slide 103*).

- The connections in the network are relationships between the sub-parts and carry information (or attributes) which can help disambiguate between similar objects.



- (c) *Schematic SLS for a Complex Shape*

# Robot Programming and Robot Vision

- All information about position and orientation of objects has been derived in an image frame for reference
- However, objects only exists in the so-called real world.

- We have to address the relationship between images and the 3D environment from which they are derived.

- Even then, this is typically not enough since there is little point in knowing where things are if we don't know how to manipulate and move them.

- describe 3D position and orientation of objects relative to any other object
- how to go about configuring a task to effect their manipulation
- how, given their image position and orientation, we can derive their 3D pose.

- **Robot Sensors :**
- Robot systems employ a huge range of sensors - both internally and externally.
- Internal sensors allow the robot to provide feedback on its configuration (*e.g.* using potentiometers, tachometers, optical switches, etc.).

# A Brief Review of Robot Programming Methodologies

- *Modern Robotic Manipulators*
- There are, broadly speaking, three main categories of robot programming system which are, in order of the level of sophistication ...

- Guiding Systems
- Robot-Level or Explicit-Level Systems and
- Task Level Systems

- Guiding systems are typified by the manual lead-through approach in which the manipulator is trained by guiding the arm through the appropriate positions using, for example, a teach-pendant and recording the individual joint positions.

- Task execution is effected by driving the joints to these recorded positions.
- 
- This type of manual teaching is the most common of all programming systems.

- Robot-level programming systems, for the most part, simply replace the teach pendant with a robot programming language :
  - manipulator movements are still programmed by explicitly specifying joint positions.

- However, several languages also facilitate robot control in a three dimensional cartesian space, rather than in the joint space.
  - *Inverse Kinematic Solution* of the manipulator arm

- the Kinematic Solution allows you to compute the position of the end-effector or gripper in a 3D Cartesian frame of reference, given the manipulator joint positions.
- the Inverse Kinematic Solution allows you to compute the joint positions for a given position and orientation of the end-effector.

- The more advanced of these languages incorporate structured programming control constructs.
- They make extensive use of *co-ordinate transformations* and *co-ordinate frames*.

- With this approach
  - the robot control is defined in terms of transformations on a co-ordinate frame (a set of *XYZ* axes) associated with, and embedded in, the robot hand

- off-line programming is more feasible as long as the transformations representing the relationships between the frames describing the objects in the robot environment are accurate.

- Task-level robot programming languages attempt to describe assembly tasks as sequences of goal spatial relationships between objects
  - they focus on the objects rather than on the manipulator
  - the robot is merely a mechanism to achieve these goals.

- they typically require the use of task planning, path planning, collision avoidance and world-modelling.
- *This level of sophistication is not yet widely available on a commercial basis.*

# Description of Object Pose with Homogenous Transformations

- Robot manipulation is concerned, in essence, with the spatial relationships between several objects, between objects and manipulators, and with the reorganisation of these relationships.
- We will use *homogenous transformations* to represent these spatial relationships.

- However, we first need to review a little vector algebra to ensure that we are equipped with the tools to develop this methodology.

- A 3D vector,  $v = ai + bj + ck$ , where  $i, j$  and  $k$  are unit vectors along the  $X, Y$  and  $Z$  axes is represented in homogenous co-ordinates as  $v = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$
- where  $a = \frac{x}{w}, b = \frac{y}{w}$  and  $c = \frac{z}{w}$

- Thus, the additional fourth co-ordinate  $w$  is just a scaling factor and means that a single 3D vector can be represented by several homogenous co-ordinates.

- For example,  $3i + 4j + 5k$  can be represented by

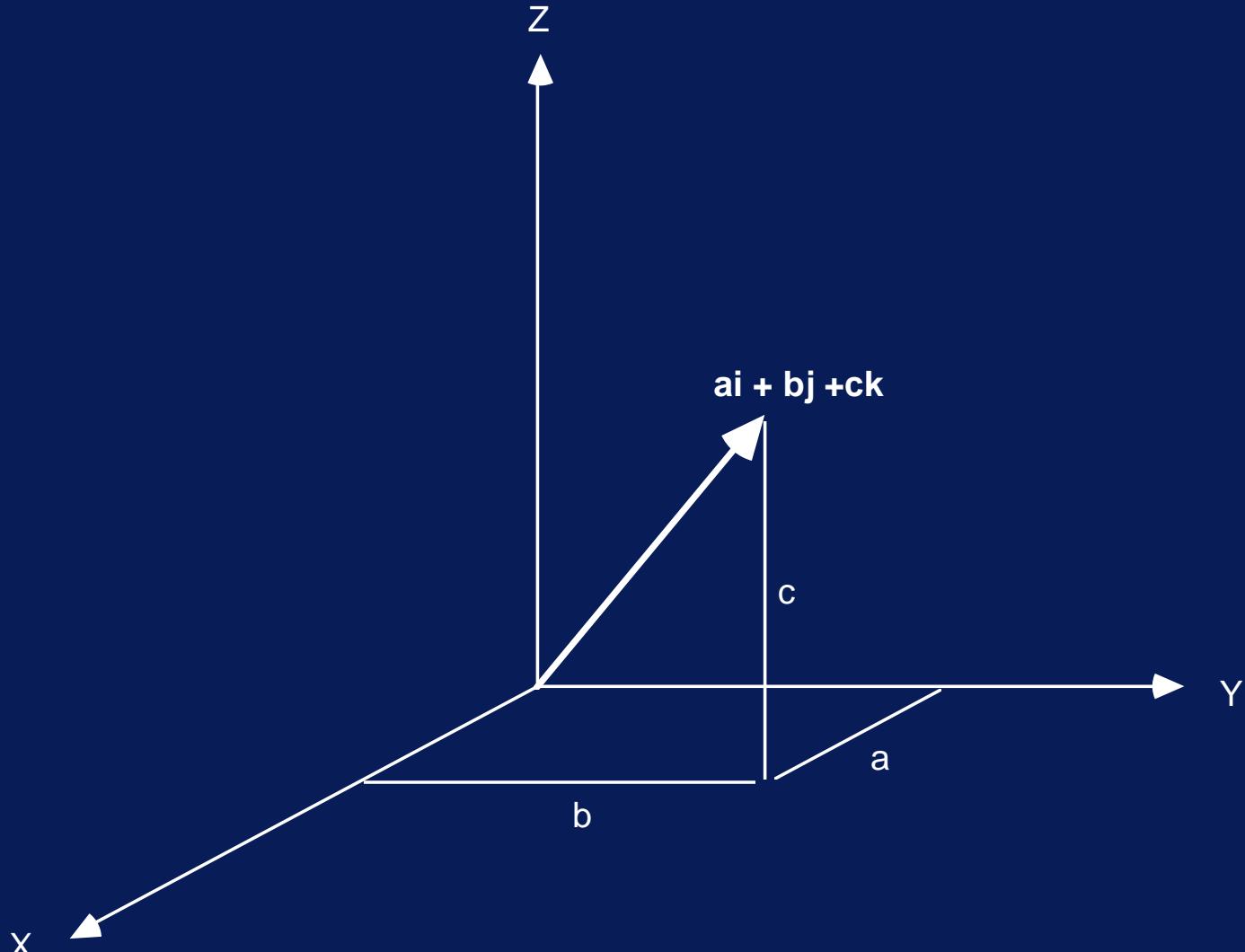
$$\begin{bmatrix} 3 \\ 4 \\ 5 \\ 1 \end{bmatrix}$$

or by

$$\begin{bmatrix} 6 \\ 8 \\ 10 \\ 2 \end{bmatrix}.$$

- Note that, since division of zero is indeterminate, the vector

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



- *Co-ordinate Reference Frame*

- Vectors can be combined in two simple ways. Given two vectors  $a$  and  $b$ ,
- $$a = a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k}$$
- $$b = b_x \mathbf{i} + b_y \mathbf{j} + b_z \mathbf{k}$$
- the *Vector Dot Product* is defined:
- $$a \cdot b = a_x b_x + a_y b_y + a_z b_z$$
- and is a scalar quantity.

- The *Vector Cross Product* is defined:

$$a \times b = (a_y b_z - a_z b_y) \mathbf{i} + (a_z b_x - a_x b_z) \mathbf{j} + (a_x b_y - a_y b_x) \mathbf{k}$$

- It may also be written as :
- that is, as the expansion of
- this  $3 \times 3$  determinant.

$$a \times b = \begin{vmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

- A general transformation  $H$ , in 3D space, representing translation, rotation, stretching and perspective distortions, is a  $4*4$  matrix in homogenous formulation.
- Given a point represented by the vector  $u$ , its transformation  $v$  is represented by the matrix product :  $v = \textcolor{blue}{H}u$ .

- The transformation  $H$  corresponding to a translation by a vector  $a_i+b_j+c_k$  is

$$H = \text{TRANS}(a, b, c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- For example : to transform  $u = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$  by  $H$  :

$$v = Hu = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x + aw \\ y + bw \\ z + cw \\ w \end{bmatrix} = \begin{bmatrix} x/w + a \\ y/w + b \\ z/w + c \\ 1 \end{bmatrix}$$

- The transformation corresponding to rotations about  $X$ ,  $Y$  or  $Z$  axes by an angle  $\theta$  are :

$$Rot(X, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(Y, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(Z, \theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- We can interpret the homogenous transformation as a *co-ordinate reference frame*.
- In particular, a homogenous transformation describes the position and orientation of a co-ordinate frame with respect to another previously defined co-ordinate frame.

- Thus, the homogenous transformation represents, not only transformations of vectors (points) but also positions and orientations.

- Specifically, a co-ordinate frame is defined by four things: the position of its origin and the direction of its  $X$ ,  $Y$  and  $Z$  axes.
  - the first three columns of the homogenous transformation represent the direction of the  $X$ ,  $Y$  and  $Z$  axes of the co-ordinate frame with respect to the base co-ordinate reference frame

- the fourth column represents the position of the origin.
- A homogenous transformation, which can be a combination of many simpler homogenous transformations, applies equally to other homogenous transformations as it does to vectors.

- Thus, we can take a co-ordinate reference frame and move it elsewhere by applying an appropriate homogenous transformation.

- If the co-ordinate frame to be “moved” is originally aligned with the so-called base co-ordinate reference frame
  - the homogenous transformation is
  - a description of how to transform the base co-ordinate frame to the new co-ordinate frame
  - and ...

- a description of this new co-ordinate frame with respect to the base co-ordinate reference frame.

- For example, consider the following transformation :
- $H Trans(10,10,0) Rot(Y,90)$

$$= Trans(10,10,0) \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 10 \\ 0 & 1 & 0 & 10 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly, the transformation of the null vector, *i.e.* the vector which performs no translation and thus defines the origin of the base co-ordinate frame, is given by :

$$\begin{bmatrix} 10 \\ 10 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 10 \\ 0 & 1 & 0 & 10 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- The transformation of the three vectors corresponding to the unit vectors along the  $X$ ,  $Y$  and  $Z$  axes are ...

$$\begin{bmatrix} 0 & 0 & 1 & 10 \\ 0 & 1 & 0 & 10 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ -1 \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 1 & 10 \\ 0 & 1 & 0 & 10 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 11 \\ 0 \\ 1 \end{bmatrix} \text{ and}$$

$$\begin{bmatrix} 0 & 0 & 1 & 10 \\ 0 & 1 & 0 & 10 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 11 \\ 10 \\ 0 \\ 1 \end{bmatrix} \text{ respectively.}$$

- The direction of these (transformed) unit vectors is formed by
  - subtracting the vector representing the origin of this co-ordinate frame
  - extending the vectors to infinity by reducing the scale factor to zero.

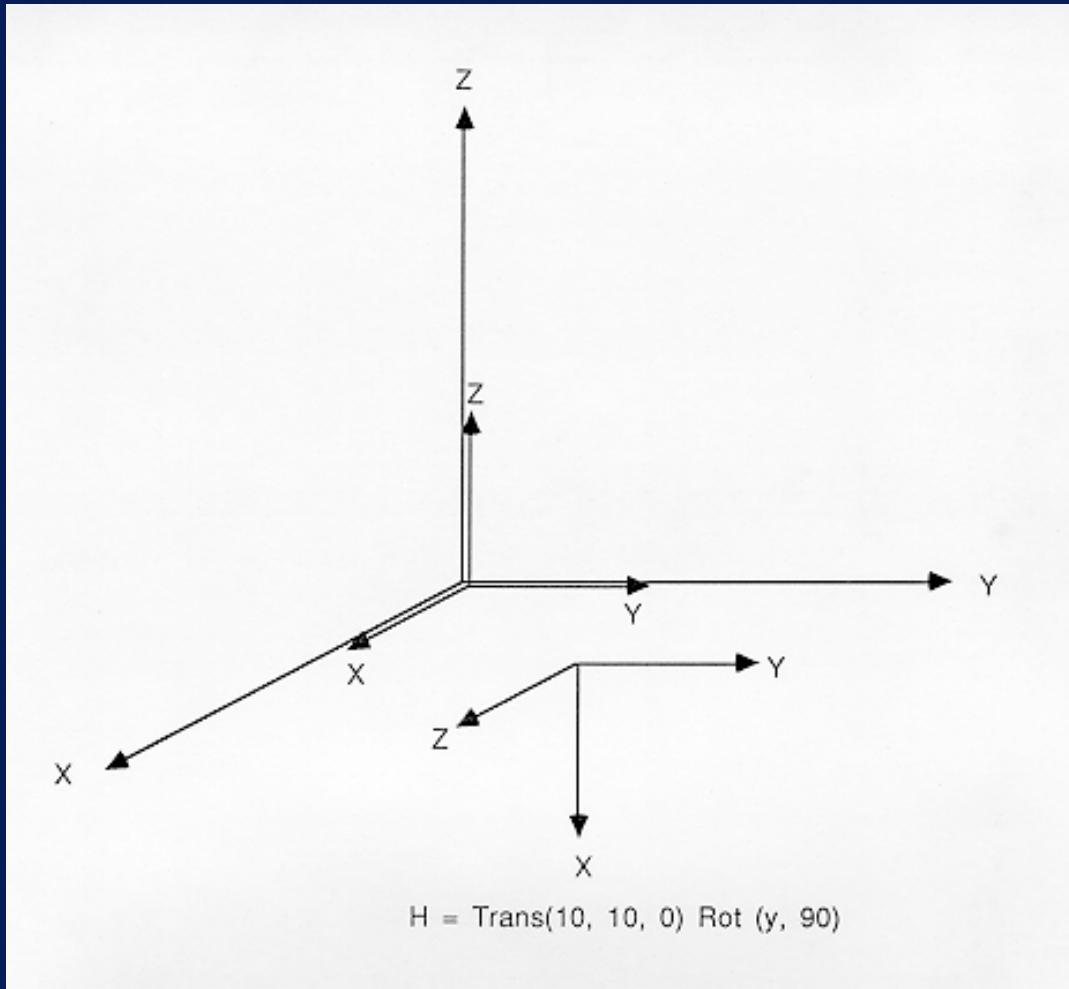
- Thus, the direction of the X, Y and Z axes of this (new) frame are

$$\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

- these four results show us that the new origin is at co-ordinates  $(10, 10, 0)$ ;
- the new  $X$  axis is directed along the  $Z$  axis of the base co-ordinate reference frame in the negative direction

- the new  $Y$  axis is directed along the  $Y$  axis of the base co-ordinate reference frame in the positive direction
- and the new  $Z$  axis is directed along the  $X$  axis of the base co-ordinate reference frame in the positive direction.

- Remember when deciding in which sense to make a rotation that:
  - a positive rotation about the  $X$  axis takes the  $Y$  axis toward the  $Z$  axis;
  - a positive rotation about the  $Y$  axis takes the  $Z$  axis toward the  $X$  axis;
  - a positive rotation about the  $Z$  axis takes the  $X$  axis toward the  $Y$  axis.



- *Interpreting a homogenous transformation*
  - *as a co-ordinate frame.*

- The rotations and translations we have been describing have all been made relative to the fixed base reference frame.
- Thus, in the transformation given by :
  - $H = \text{Trans} (10, 10, 0) \text{Rot} (\text{Y}, 90)$

- The frame is first rotated around the reference  $Y$  axis by  $90^\circ$  and then translated by  $10i + 10j + 0k$ .

- This operation may also be interpreted in reverse order, from left to right, viz:
  - the object (frame) is first translated by  $10i + 10j + 0k$ ;
  - it is then rotated by  $90^\circ$  around the station frame axis ( $Y$ ).

- In this instance, the effect is the same, but in general it will not be.
- This second interpretation seems to be the most intuitive since we can forget about the base reference frame and just remember “where we are”: our current station coordinate reference frame.

- We then just need to decide what transformations are necessary to get us to where we want to be based on the orientation of the station axes.

- In this way, we can get from pose to pose by incrementally identifying the appropriate station transformations,  $H_1, H_2, H_3, \dots H_n$ , which we apply sequentially, as we go, and the final pose is defined with respect to the base simply as  $H = H_1 * H_2 * H_3 * \dots * H_n$ .

- In order to clarify the relative nature of these transformations, each of these frames/transformations is normally written with a leading superscript which identifies the co-ordinate frame with respect to which the (new) frame/transformation is defined.

- The leading superscript is omitted if the defining frame is the base frame.
- Thus, the above transform equation is more correctly written :
  - $H = H_1 * H_1 H_2 * H_2 H_3 * \dots * H_{n-1} H_n$

- As a general rule :
  - If we post-multiply a transform representing a frame by a second transformation describing a rotation and/or translation we make that rotation/transformation with respect to the frame axis described by the first transformation.

- If we pre-multiply the frame transformation representing a rotation/transformation then the rotation/transformation is made with respect to the base reference co-ordinate frame.

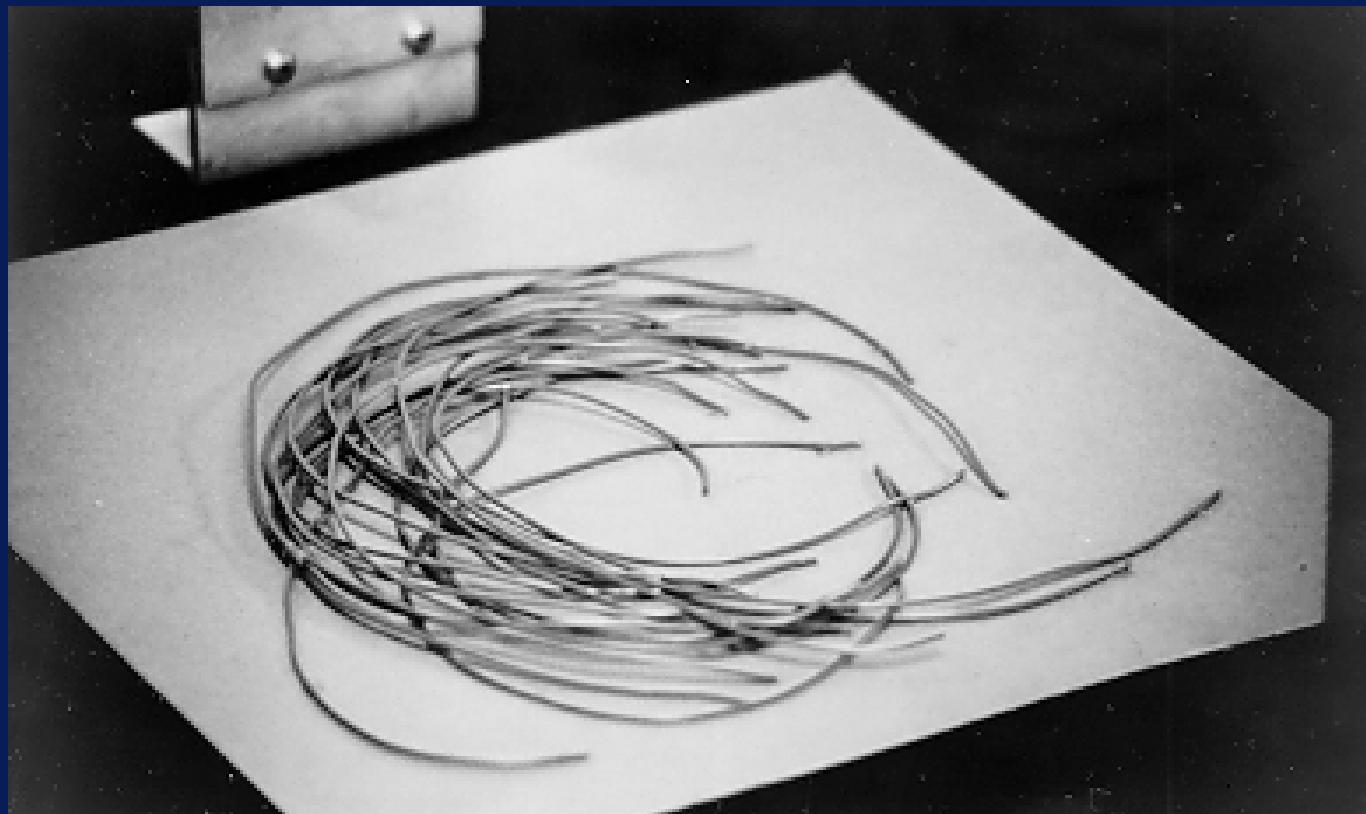
- At this stage, we have developed a system where we can specify the position and orientation of co-ordinate reference frames anywhere with respect to each other and with respect to a given base frame.

- This, in itself, is quite useless since the world you and I know does not have too many co-ordinate reference frames in it.
- What we really require is a way of identifying the pose of objects.

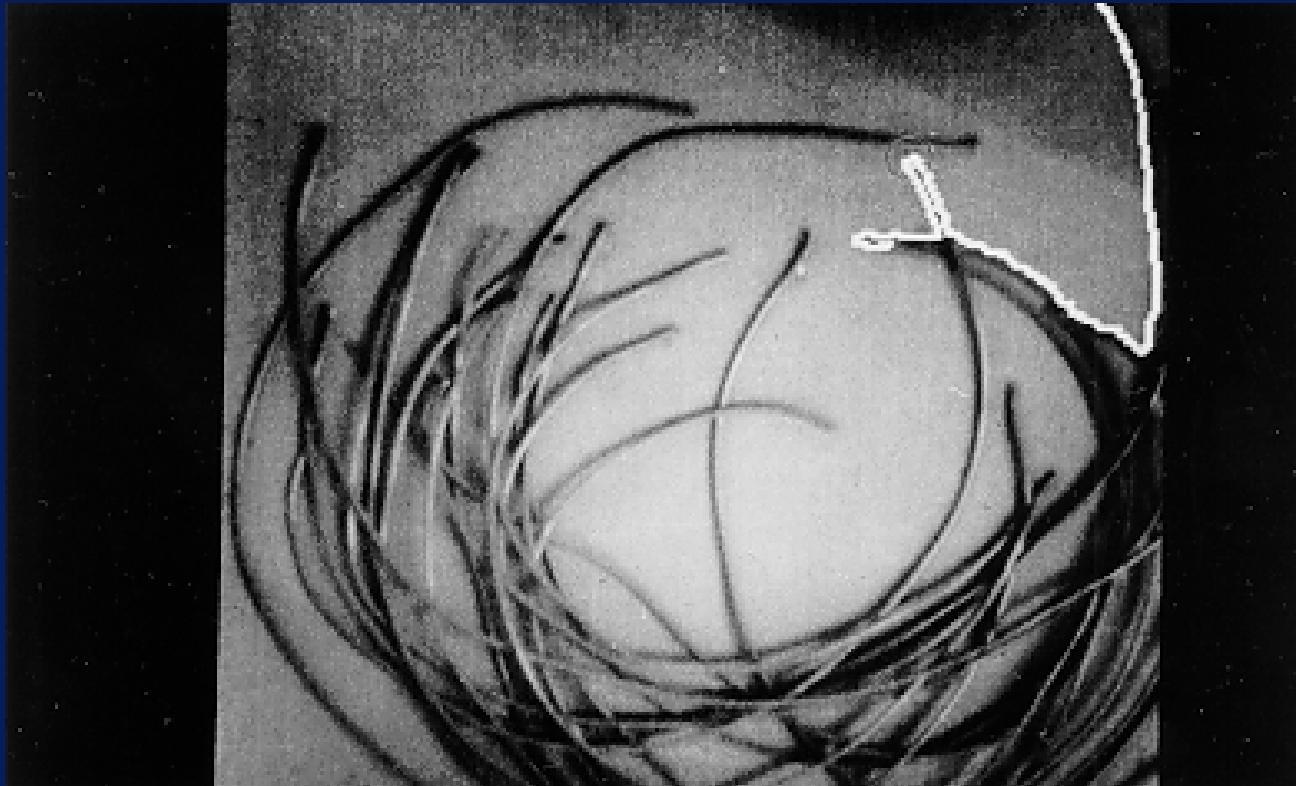
- The trick, and it is no more than a trick, is to *attach* a co-ordinate frame to an object, *i.e.* symbolically glue an  $XYZ$  frame into an object simply by defining it to be there.
- Now, as we rotate and translate the co-ordinate frame, so too do we rotate and translate objects.

# Robot Programming : A Wire Crimping Task Specification.

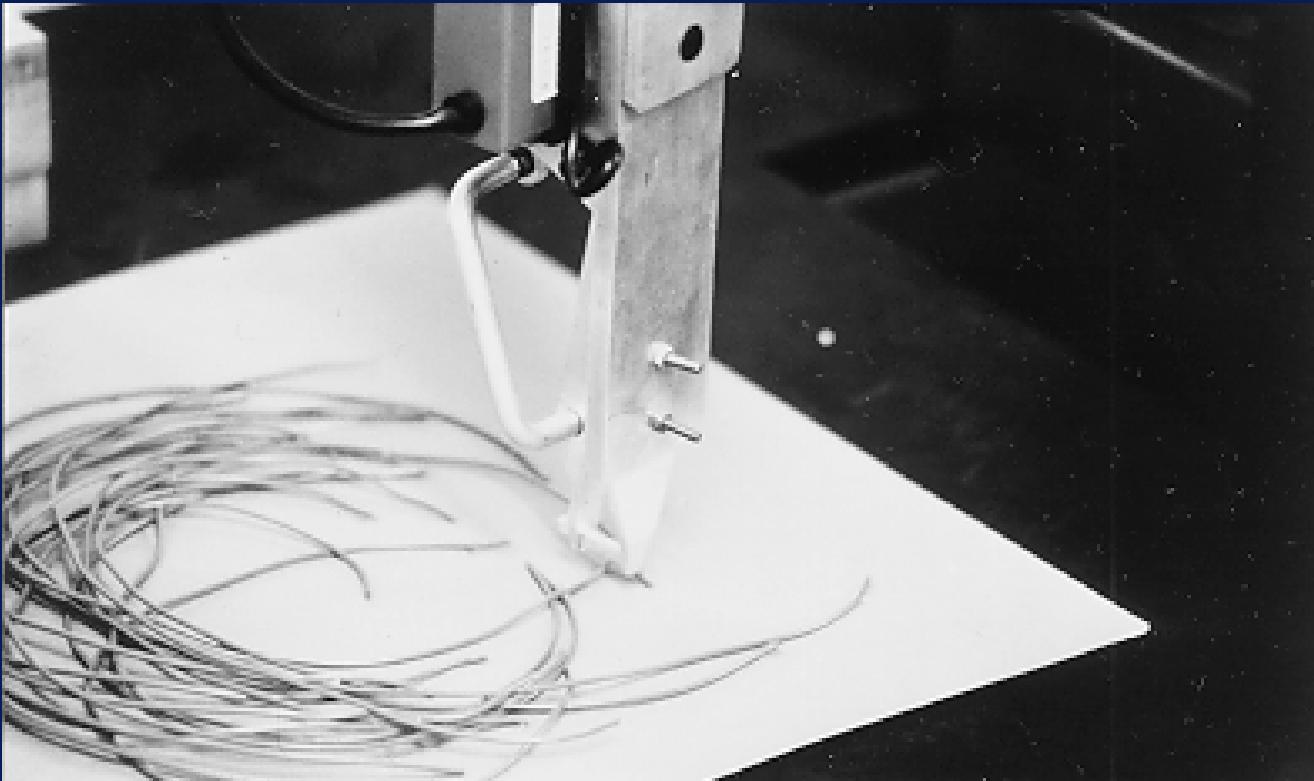
- By defining a series of manipulator end-effector positions  $M_n$ , say, this task can be described as a sequence of manipulator movements and actions referred to these defined positions.
- For example, the task might be formulated as follows:



- *A tray of flexible electrical wires*



- *An image of which yields a point on a wire which is suitable for grasping*



- *Allowing the robot to pick up the wire*
  - *and manipulate it.*

- M0:** Move out of the field of view of the camera.  
Determine the position and orientation of  
the wire-end and the grasp point using the  
vision system.
- M1:** Move to a position above the centre of the  
tray of wires.
- M2:** Move to an approach position above the  
grasp point.

- M3:** Move to the grasp position. Grasp the wire.
- M4:** Move to the depart position above the grasp point.
- M5:** Move to the approach position in front of the crimp.
- M6:** Move to a position in which the wire-end and the crimp are in contact.

- M7:** Move to a position such that the wire-end is inserted in the crimp. Actuate the crimping machine.
- M8:** Move to the depart position in front of the crimping machine.
- M9:** Move to a position above the collection bin. Release the wire.

- This process is repeated until there are no more wires to be crimped.

- One of the problems with this approach is that we are specifying the task in terms of movements of the robot while it is the wire and the crimp in which we are really interested.

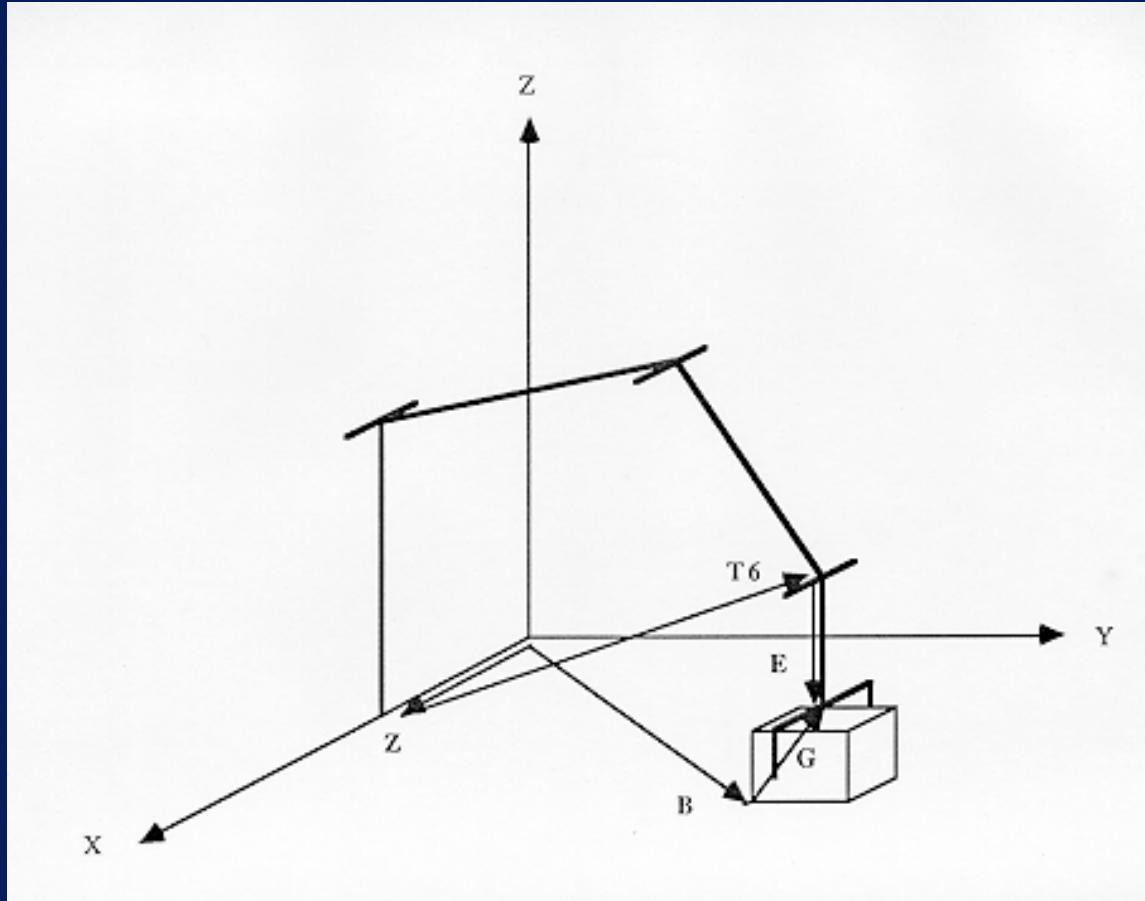
- The object movements are implicit in the fact that the manipulator has grasped it.

- However, we will try to make up for this deficiency somewhat when we describe the structure of the task by considering the structure of the task's component objects:

- the manipulator
- the end-effector
- the wire grasp position
- the wire-end
- the crimping machine
- and the crimp.

- *In particular, we will use the explicit positional relationships between these objects to describe the task structure.*

- Since co-ordinate frames can be used to describe object position and orientation and since we may need to describe a co-ordinate frame in two or more ways (there is more than one way to reach any given position and orientation), we will use transform equations to relate the two descriptions.



- *A manipulator grasping a block*

- Consider the situation, depicted in Figure 8.4 (*Slide 15*) of a manipulator grasping a toy block.
- The co-ordinate frames which describe this situation are as follows:

- Z** is the transform which describes the position of manipulator with respect to the base co-ordinate reference frame.
- zT6** describes the end of the manipulator (*i.e.* the wrist) with respect to the base of manipulator, *i.e.* with respect to Z.

- $T^6E$**  describes the end-effector with respect to the end of the manipulator, *i.e.* with respect to  $T_6$ .
- $B$**  describes a block's position with respect to the base co-ordinate reference frame.

**BG** describes the manipulator end-effector with respect to the block, *i.e.* with respect to **B**.

- In this example, the end-effector is described in two ways, by the transformations leading from the base to the wrist to the end-effector :
  - $Z * {}^ZT6 * {}^6E$
- and by the transformations leading from the block to the end-effector grip position:
  - $B * {}^B G$

- Equating these descriptions, we get the following transform equation :
  - $Z * {}^Z T_6 * {}^{T_6} E = B * {}^B G$

- Solving for  $T6$  by multiplying across by the inverse of  $Z$  and  ${}^T6E$  :
  - ${}^ZT6 = Z^{-1} * B * {}^BG * {}^T6E^{-1}$
- $T6$  is a function of the joint variables of the manipulator and, if known, then the appropriate joint variables can be computed using the ***Inverse Kinematic Solution***.

- $T6$  then is the co-ordinate which we wish to program in order to effect the manipulation task.
- An arm position and orientation specified by  $T6$  is, thus, equivalent to our previous informal movement  $M_n$ :

- Move  $Mn = \text{Move } {}^zT6$ 
  - since we can compute  $T6$  in terms of our known frames
  - we now have an arm movement which is specified in terms of the frames which describe the task structure.

- Assigning the appropriate value to  $T6$  and moving to that position, implicitly using the Inverse Kinematic Solution :

$${}^ZT6 := Z^{-1} * B * {}^BG * {}^{T6}E^{-1}$$

Move  ${}^ZT6$

- What we have not yet done, and we will omit in this instance, is to fully specify each of these frames by embedding them in the appropriate objects and specifying the transformations which define them.

- Before proceeding, it is worth noting that, in this example, as the position of the end-effector with respect to the base reference system is represented by :

$$Z * {}^Z T_6 * {}^T E$$

- this allows you to generate general-purpose and reusable robot programs.

- In particular, the calibration of the manipulator to the workstation is represented by  $Z$ , while if the task is to be performed with a change of tool, only  $E$  need be altered.

- Returning again to the wire crimping application, the transforms (*i.e.* frames) which are used in the task are as follows.

- As before :

$Z$  is the transform which describes the position of manipulator with respect to the base co-ordinate reference frame.

${}^ZT_6$  describes the end of the manipulator (*i.e.* the wrist) with respect to the base of manipulator, *i.e.* with respect to  $Z$ .

$T^6E$  describes the end-effector with respect to the end of the manipulator, *i.e.* with respect to  $T_6$ .

- We now define :

*OOV* : the position of the end-effector out of the field of view of the camera with respect to the base co-ordinate reference system.

*CEN* :

the position of the end-effector centred over table defined with respect to the base co-ordinate reference system.

*WDUMP* :

the position of the end-effector over the bin of crimped wires, defined with respect to the base co-ordinate reference system.

$W$  : the position of the wire-end, defined with respect to the base co-ordinate reference system.

$^W\!WG$  : the position of end-effector holding wire, defined with respect to the wire end.

*WGWA* : the position of end-effector approaching grasp position, defined with respect to the wire-grasp position.

*WGWD* : the position of end-effector departing grasp position (having grasped the wire), defined with respect to the original wire-grasp position.

*CM* : the position of the crimping machine, defined with respect to the base coordinate reference system.

*CMC* : the position of the crimp (ready to be attached), defined with respect to the crimping machine.

$cCA$  : the position of the wire-end approaching crimp, defined with respect to the crimp.

$cCA$  : the position of wire-end in contact with the crimp, defined with respect to the crimp.

*CCI* : the position of the wire-end inserted in the crimp, defined with respect to the crimp.

*CCD* : the position of the wire-end departing from the crimping machine (the crimp having been attached), defined with respect to the crimp.

- The manipulator movements  $M0$  through  $M9$  can now be expressed as combinations of these transforms :

$$M0 : \quad T6 = Z^{-1} * OOV * E^{-1}$$

$$M1 : \quad T6 = Z^{-1} * CEN * E^{-1}$$

$$M2 : \quad T6 = Z^{-1} * W * WG * WA * E^{-1}$$

$$M3 : \quad T6 = Z^{-1} * W * WG * E^{-1}$$

*M4* :  $T6 = Z^{-1} * W * WG * WD * E^{-1}$

*M5* :  $T6 = Z^{-1} * CM * C * CA * WG * E^{-1}$

*M6* :  $T6 = Z^{-1} * CM * C * CC * WG * E^{-1}$

*M7* :  $T6 = Z^{-1} * CM * C * CI * WG * E^{-1}$

*M8* :  $T6 = Z^{-1} * CM * C * CD * WG * E^{-1}$

*M9* :  $T6 = Z^{-1} * WDUMP * E^{-1}$

- Note that  $WA$ ,  $WD$ ,  $CA$ ,  $CI$  and  $CD$  are all translation transformations concerned with approaching and departing a particular object.

- In order to allow smooth approach and departure trajectories, these translation distances are iterated from zero to some maximum value or from some maximum value to zero (in integer intervals) depending on whether the effector is approaching or departing.

- For example:  ${}^{WG}WA$  is the approach position of the end-effector before grasping the wire and is (to be) defined as a translation, in the negative Z direction of the  $WG$  frame, of the approach distance  $z_{approach}$ , say.

- Thus,

$$WGWA = Trans(0, 0, -(z\_approach))$$

- where:

$$z\_approach = z\_approach\_initial$$

$$z\_approach\_initial - delta$$

$$z\_approach\_initial - 2 * delta$$

:

0

- *It should be noted well that this type of explicit point-to-point approximation of continuous path control would not normally be necessary with a commercial industrial robot programming language since they usually provide facilities for specifying the end-effector trajectory.*

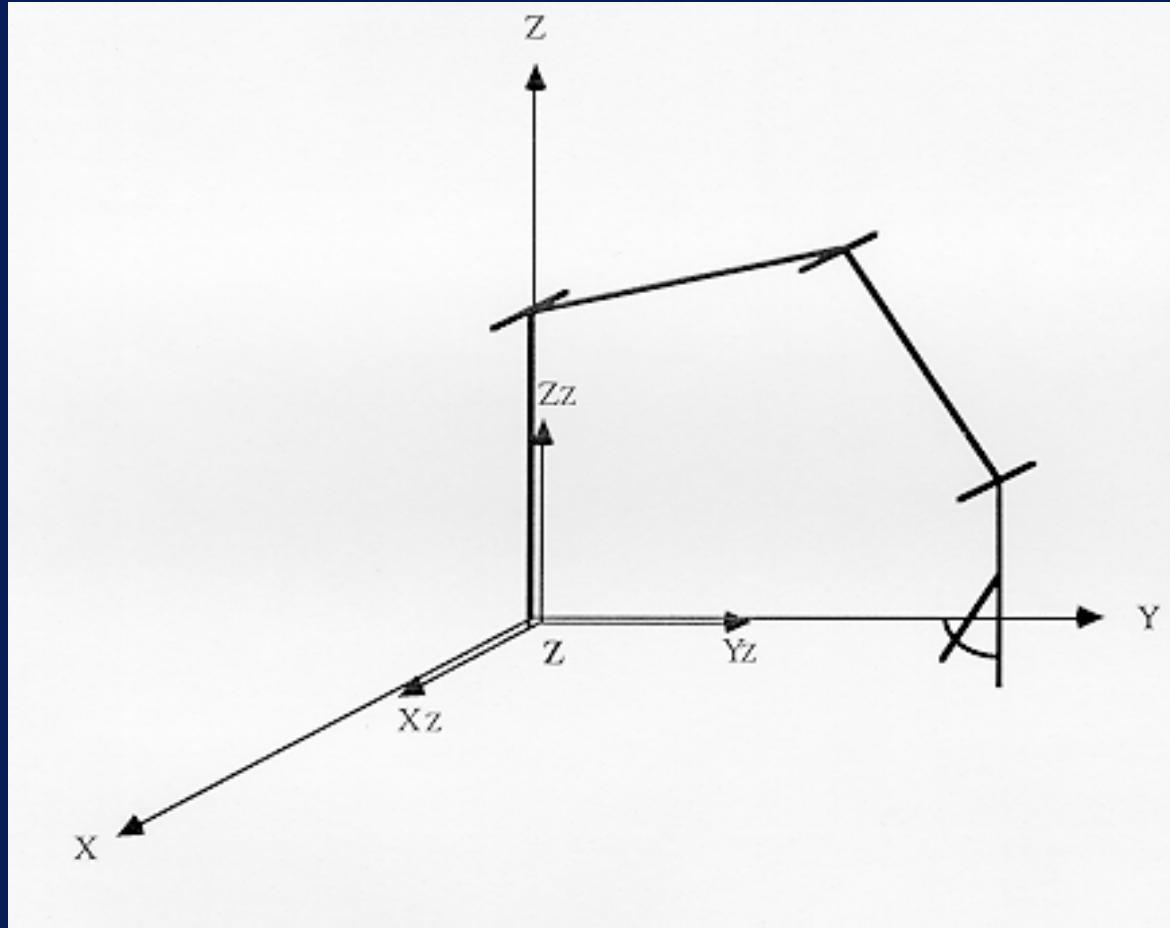
- To complete the task specification, we now have to define the rotations and translations associated with these transforms/frames.
- Most can be determined by empirical methods, embedding a frame in an object and measuring the object position and orientation.

- Others,  $W$  and  $WG$  in particular, are defined here and their components determined by visual means at run time.

**Z** : The position of the position of manipulator with respect to the base co-ordinate reference frame.

- We will assume that the base co-ordinate system is aligned with the frame embedded in the manipulator base, as shown in Figure 8.5 (*Slide 51*).
- Thus
  - $Z = I = \text{Identity Transform}$

- Note that the frame defining the manipulator base is dependent on the kinematic model of the robot manipulator.



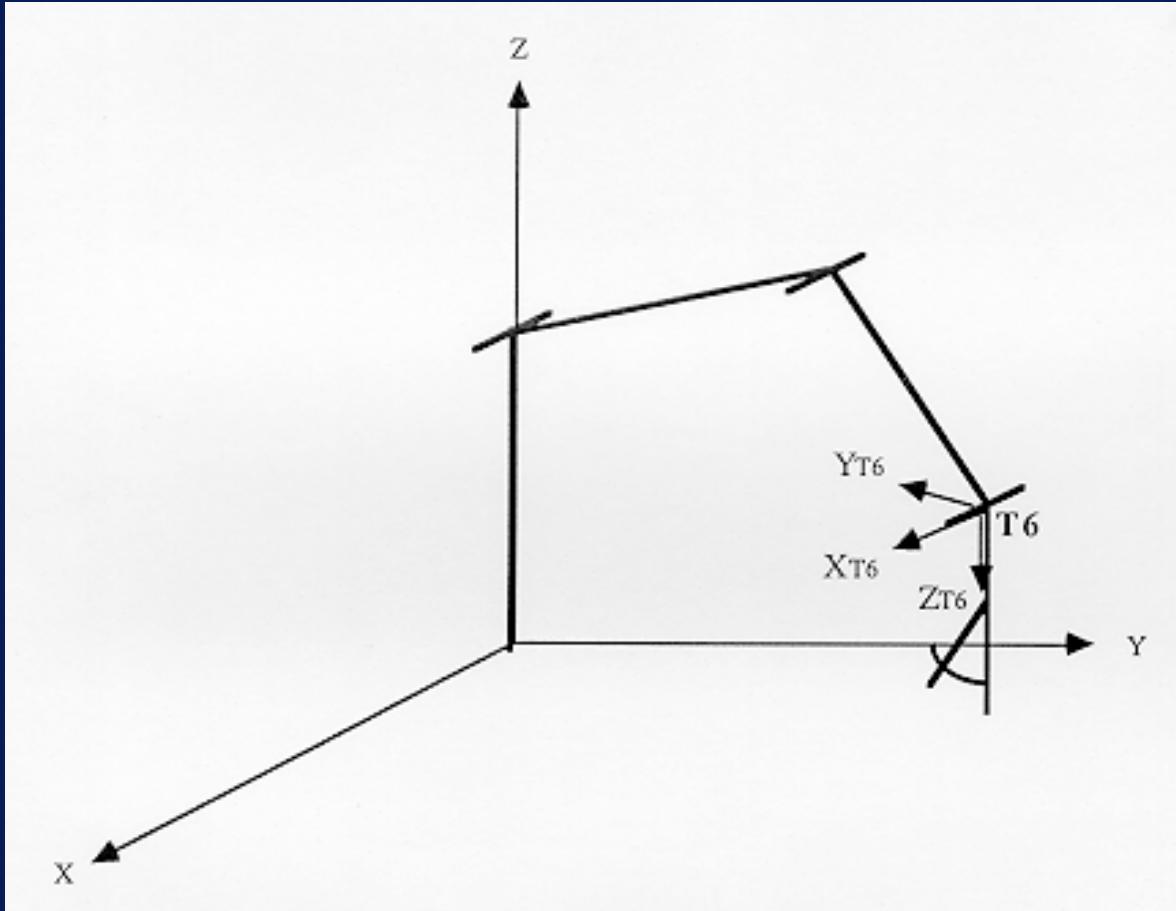
- *Z - the base of the manipulator*

*T6 : The position of the end of the manipulator with respect to its base at Z.*

- The  $T6$  frame is a computable function of the joint variables.
- Again, the frame which defines the end of the manipulator is based on the kinematic model.

- However, there is a convention that the frame should be embedded in the manipulator
  - with the origin at the wrist
  - with the Z axis directed outward from the wrist to the gripper

- with the  $Y$  axis directed in the plane of movement of the gripper when it is opening and closing
- with the  $X$  axis making up a right-hand system.



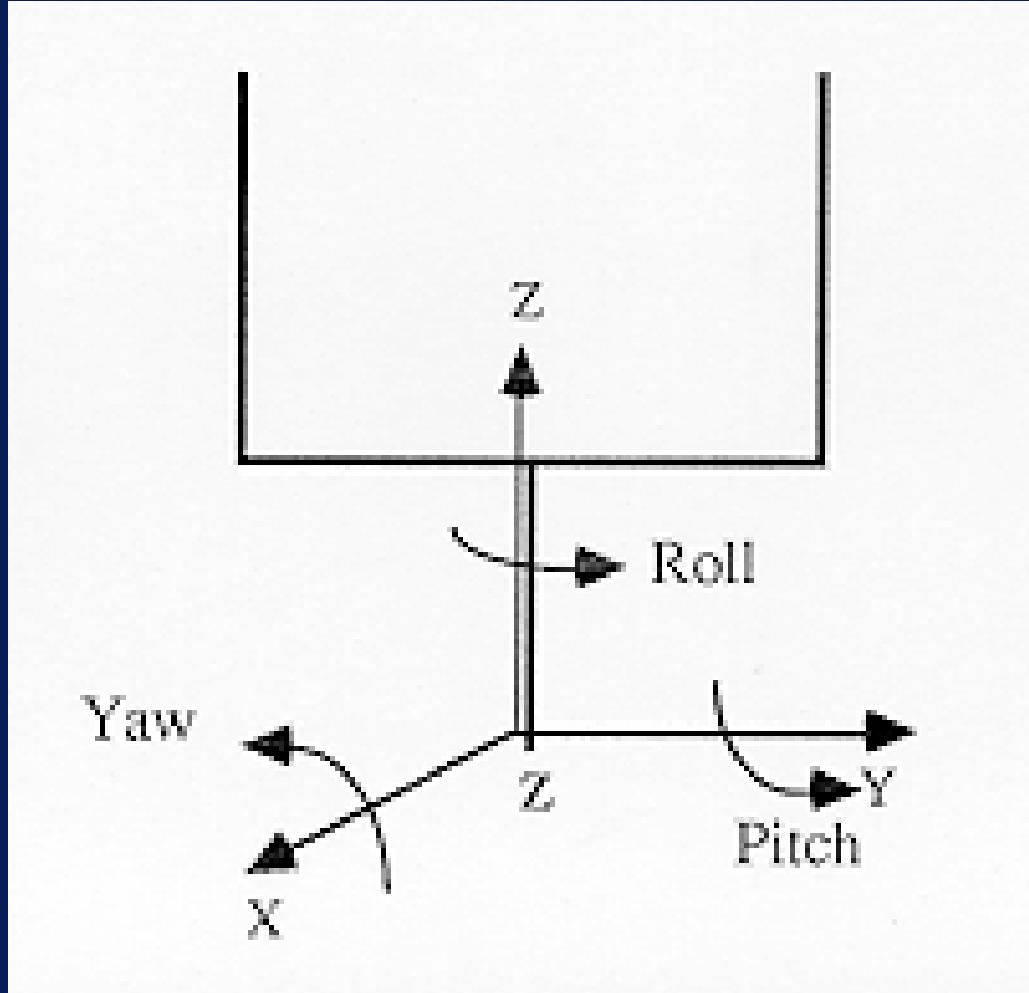
- *T<sub>6</sub> - the end of the manipulator*

- Although we will specify the orientation of  $T_6$  by solving for it in terms of other frames/transforms in the task specification, there is a commonly-used convention for specifying the orientation of objects, in general, and  $T_6$ , in particular.

- This convention identifies three rotations about the station co-ordinate frame embedded in the object which are applied in turn and in a specified order.

- The rotations are referred to
  - as a roll of  $\phi$  degrees about the station  $Z$  axis,
  - a pitch of  $\theta$  degrees about the station  $Y$  axis,
  - yaw of  $\psi$  degrees about the station  $X$  axis.

- The order of rotation is specified as
  - $RPY(\phi, \theta, \psi) = Rot(Z, \phi) Rot(Y, \theta) Rot(X, \psi)$
  - Thus, the object is first rotated  $\phi^\circ$  about the  $Z$  axis, then  $\theta^\circ$  about the current station  $Y$  axis, and finally  $\psi^\circ$  about the station  $X$  axis;
  - Refer again to Figure 8.7 (*Slide 60*)

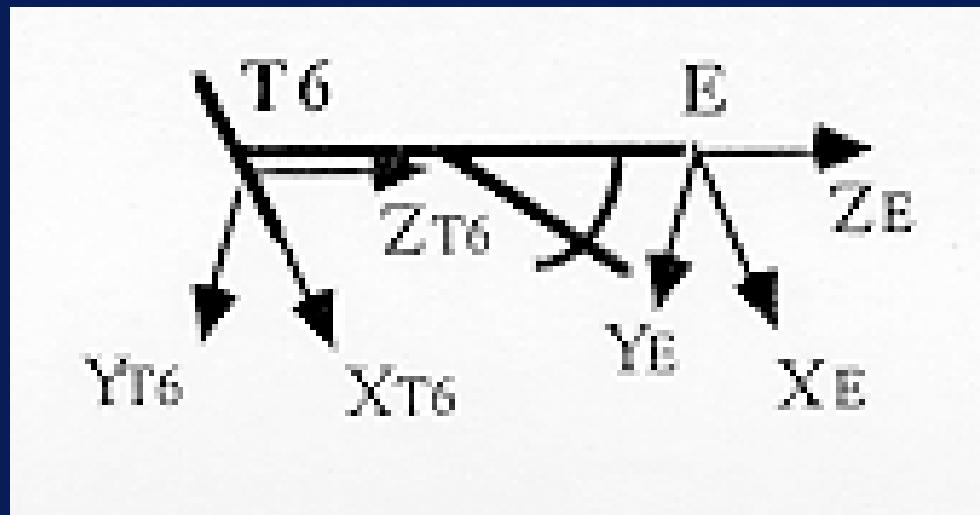


- *Specifying the orientation of T6 using Roll, Pitch and Yaw angles*

*E : the position of the end-effector with respect to the end of the manipulator, i.e. with respect to T6.*

- The frame  $E$  representing a special-purpose end-effector for grasping wires is embedded in the tip of the effector, as shown in Figure 8.8 (*Slide 64*) and hence is defined by a translation of 209mm along the  $Z$  axis of the  $T_6$  frame and a translation of -15mm along the  $Y$  axis of the  $T_6$  frame.

- Thus :
  - $T^6 E = \text{Trans} (0, -15, 209)$

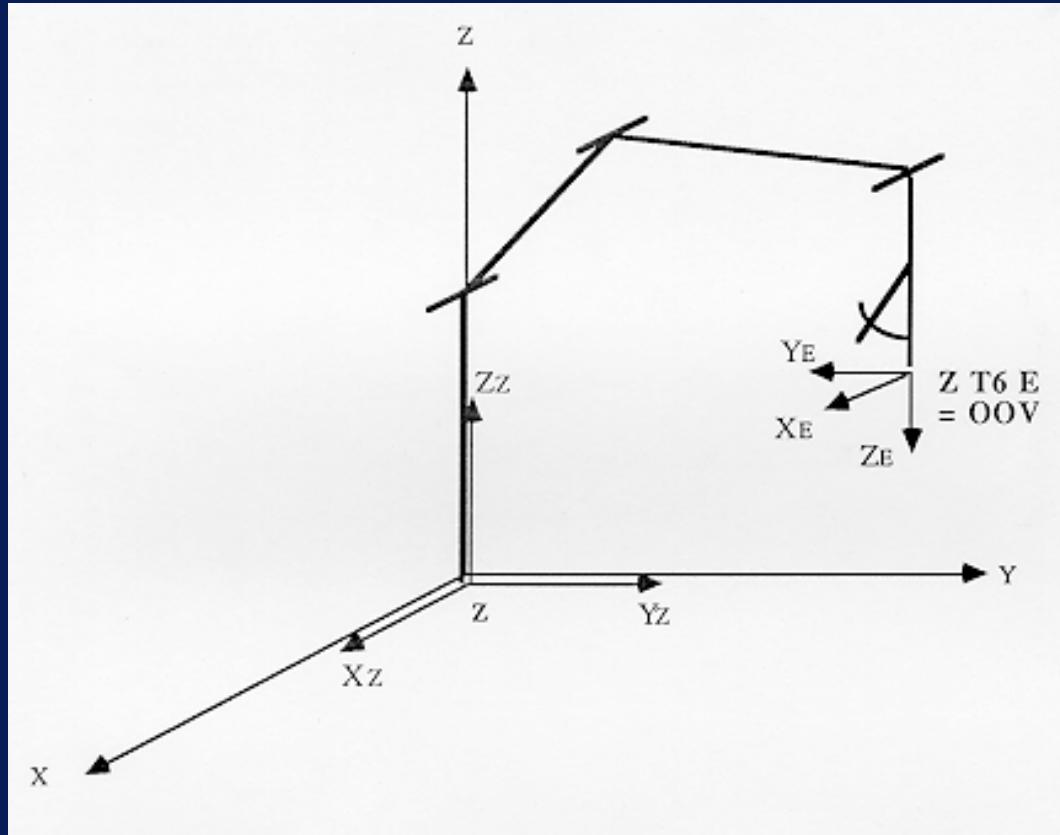


- *E - the end-effector*

*OOV*: *the position of the end-effector out of the field of view of the camera.*

- This position is defined, with respect to the base co-ordinate system, such that the end-effector is directed vertically downwards, as shown in Figure 8.9 (*Slide 67*).

- Thus,  $OOV$  is defined by a translation (of the origin) to the point given by the coordinates (150, 300, 150) followed by a rotation of -180° about the station X axis :
- $OOV = Trans(150, 300, 150) Rot(X, -180)$

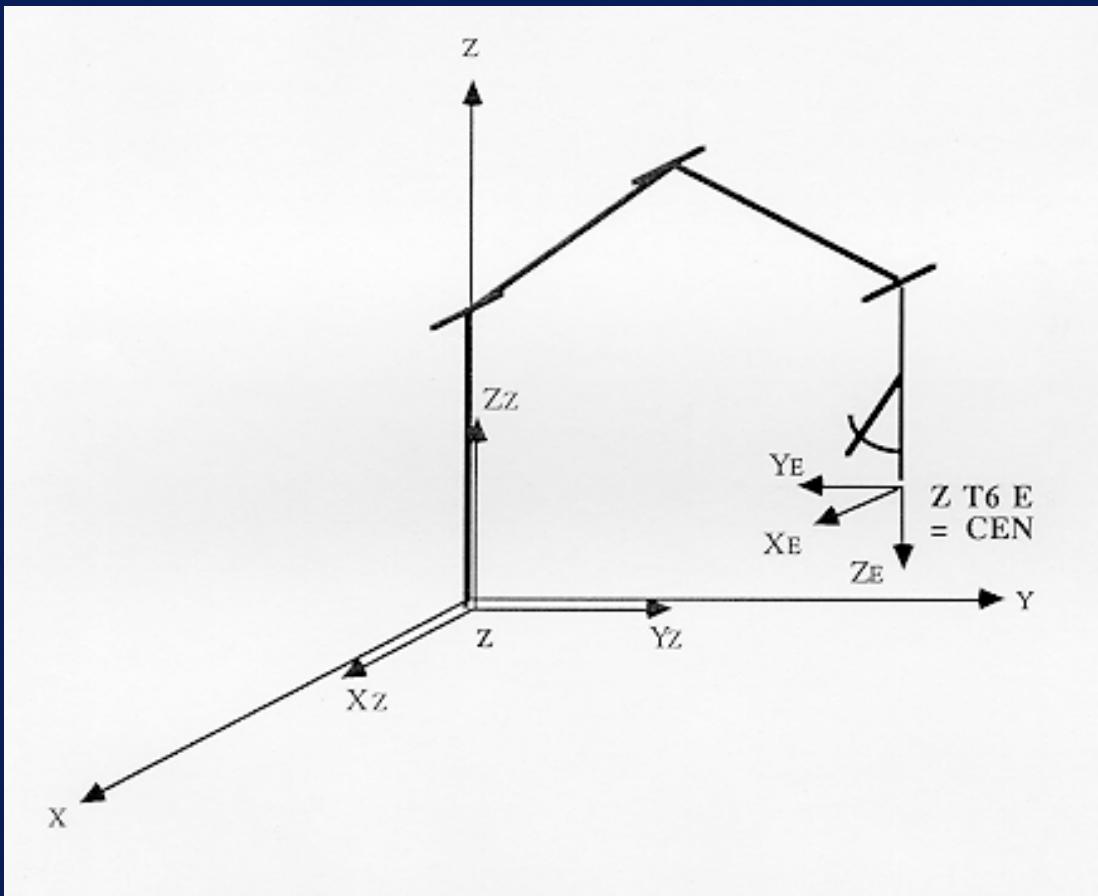


- *OOV - the end-effector out of the camera's field of view*

*CEN : the position of the end-effector centred over the tray, defined with respect to the base co-ordinate reference frame.*

- This position is defined such that the end-effector is directed vertically downwards over the centre of the tray, as shown in Figure 8.10 (*Slide 70*).

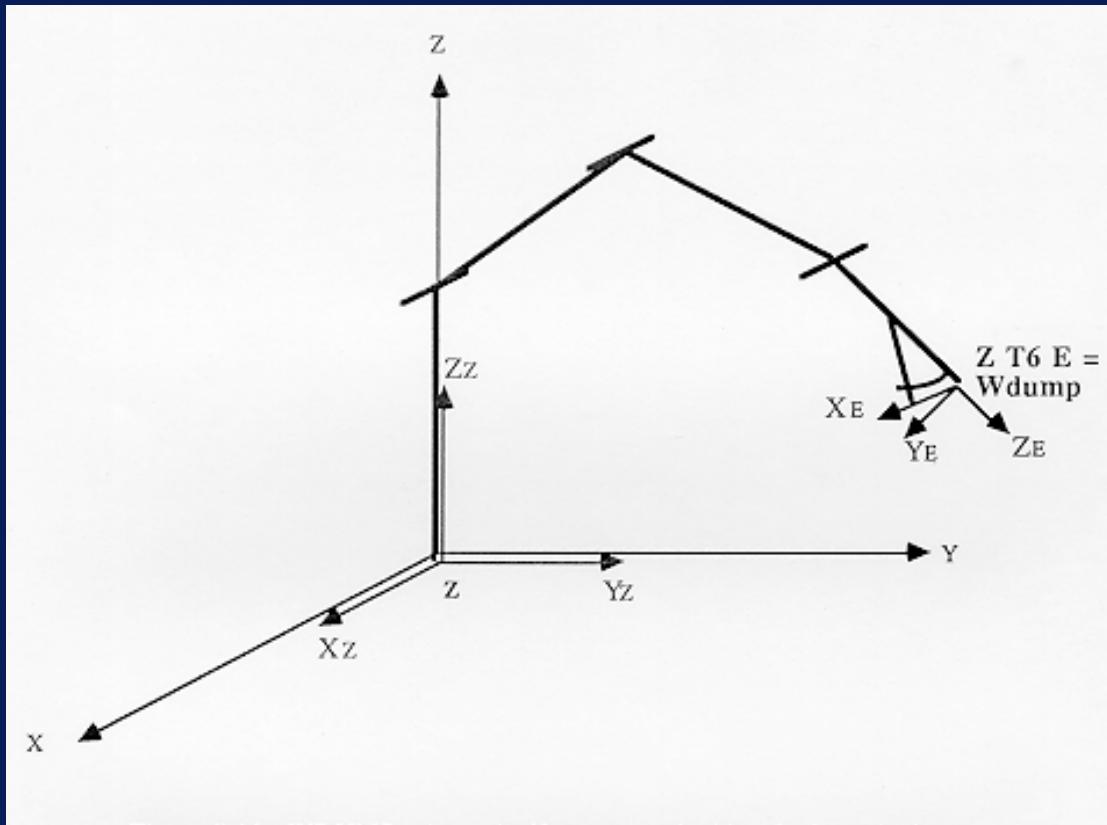
- Thus,  $CEN$  is defined by a translation (of the frame origin) to the point given by the co-ordinates (0, 360, 150) followed by a rotation of -180° about the station X axis :
- $CEN = Trans(0, 360, 150) Rot(X, -180)$



- *CEN - the end-effector centred  
over the trays of wires*

*WDUMP : the position of the end-effector over the bin of crimped wires, defined with respect to the base co-ordinate reference frame.*

- The position is defined such that the end-effector is directed -45° to the horizontal over the centre of a bin as shown in Figure 8.11 (*Slide 72*).



- *WDUMP - the end-effector over the bin of crimped wires*

- Thus, *WDUMP* is defined by a translation of the frame origin to the point given by the co-ordinates (0, 500, 160) followed by a rotation of -135° about the station X axis.

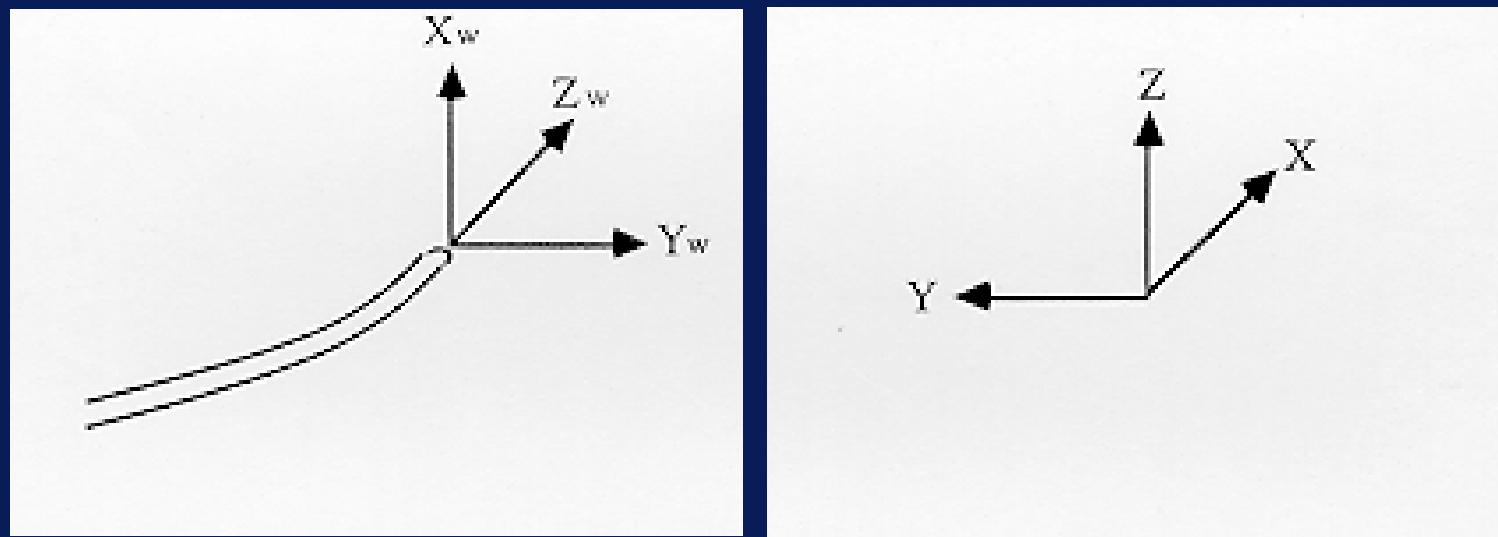
*W : the position of the wire-end, defined with respect to the base co-ordinate reference frame.*

- The origin of the wire frame  $W$  is defined to be at the end of the wire, with its Z axis aligned with the wire's axis of symmetry.

- The  $X$  axis is defined to be normal to the tray on which the wires lie, directed vertically upwards.
- The  $Y$  axis makes up a right-hand system.
- Since we are assuming that the wires are lying flat on the tray, both the  $X$  and  $Y$  axes lie in the plane of the tray.

- Furthermore, we will assume that the tray lies in the  $X$ - $Y$  plane of the base reference frame.
- Thus,
  - $W = \text{Trans}(x, y, 0) \text{Rot}(z, \theta) \text{Rot}(y, -90)$

- It is the responsibility of the vision system to analyse the image of the wires and to generate the components of this frame automatically, specifically by computing  $x$ ,  $y$  and  $\theta$ .
- $W$  is illustrated in Figure 8.12 (*Slide 78*).



- *W - the position of the wire-end*

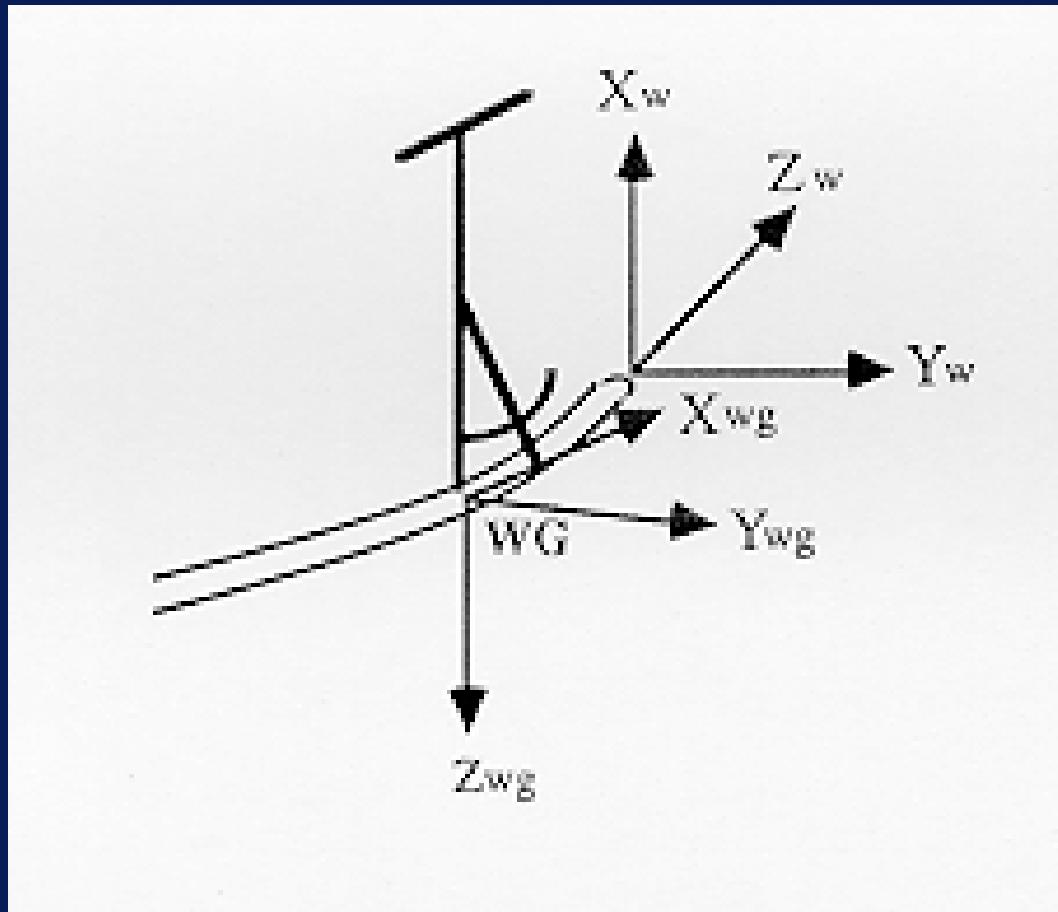
*WG : the position of the end-effector holding the wire, defined with respect to the wire-end.*

- The origin of the wire gripper frame  $WG$  is defined to be located a short distance from the origin of  $W$ , on the wire's axis of symmetry.
- The Z axis is defined to be normal to the plane of the tray, directed downwards.

- The  $Y$  axis is defined to be normal to the axis of symmetry of the wire, in the plane of the tray.
- The  $Y$  axis makes up a right-hand system.
- $WG$  is illustrated in Figure 8.13 (*Slide 83*).

- It is important to note that we define the  $WG$  frame in this manner since this is how the end-effector  $E$  will be oriented when grasping the wire, *i.e.*, with the  $Z$  axis pointing vertically downward and the  $Y$  axis at right angles to the wire.

- As with  $W$ , the vision system must return a homogenous transformation defining this frame; we cannot assume that  $WG$  will be a fixed offset from  $W$  since we are assuming that the curvature of the wire near the end will vary from wire to wire.



- *WG - the wire grasp position*

*WA : the position of the end-effector approaching the grasp position, defined with respect to the wire-grasp position.*

- This is defined to be a position directly above the wire grasp point.
- As such, it simply involves a translation in the negative direction of the Z axis of the *WG* frame.

- Since it is wished to approach the wire along a known path, many approach positions are used in which the translation distances get successively smaller.
- This motion, then, approximates continuous path control.

- Thus :

$$WA = Trans(0, 0, -(z\_approach)),$$

- where

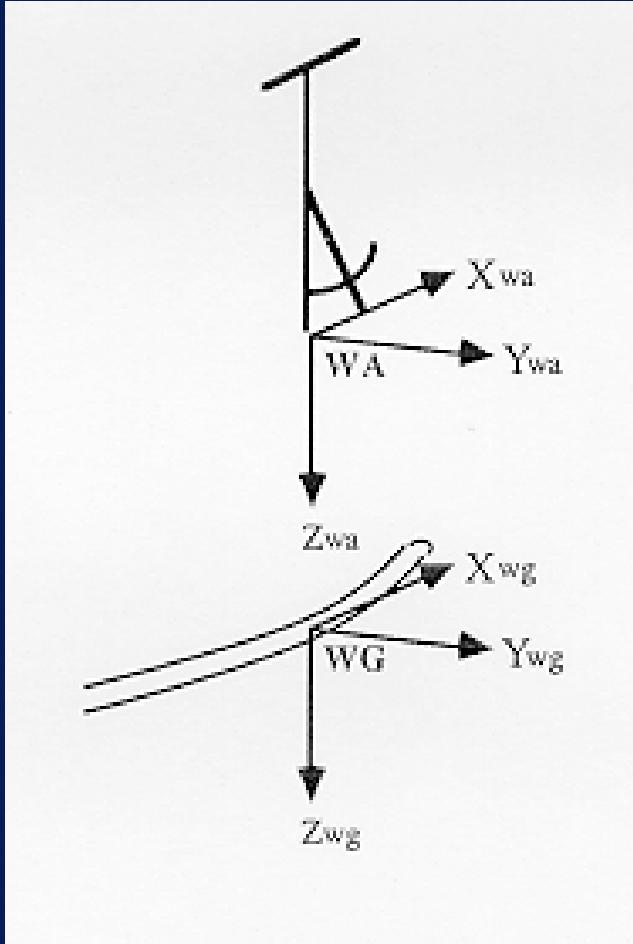
$$z\_approach = z\_approach\_initial$$

$$z\_approach\_initial - delta$$

$$z\_approach\_initial - 2 * delta,$$

:

0



- *WA - the position of the end-effector*
  - *approaching the grasp position*

*WD : the position of the end-effector departing the grasp position (having grasped the wire), defined with respect to the original wire-grasp position.*

- In a similar manner to  $WA$ ,  $WD$  is defined as a translation in the negative direction of the  $Z$  axis of the  $WG$  frame, except that in this case the translation distance becomes successively greater.

- Hence :

$$WD = Trans(0, 0, -(z\_depart)),$$

- where

$$z\_depart = 0,$$

$$delta,$$

$$2 * delta,$$

:

$$z\_depart\_final$$

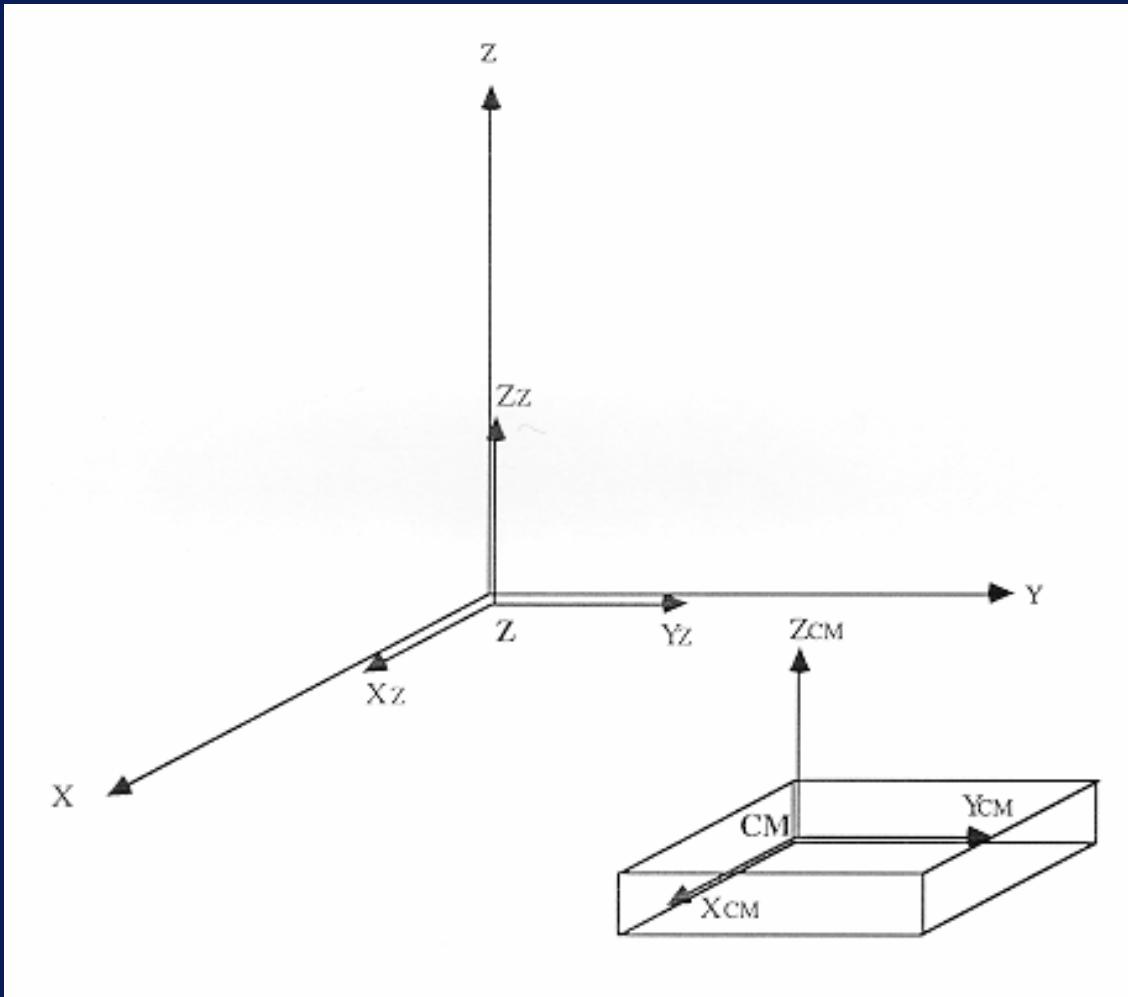
*CM : the position of the crimping machine,  
defined with respect to the base co-  
ordinate reference system.*

- The frame  $CM$ , representing the crimping machine, is defined to be embedded in a corner of the machine, as shown in Figure 8.15 (*Slide 93*).

- Thus :

$$CM = Trans(150, 300, 0)$$

- Note that the co-ordinates of the origin of  $CM$ , (150, 300, 0), are determined empirically.



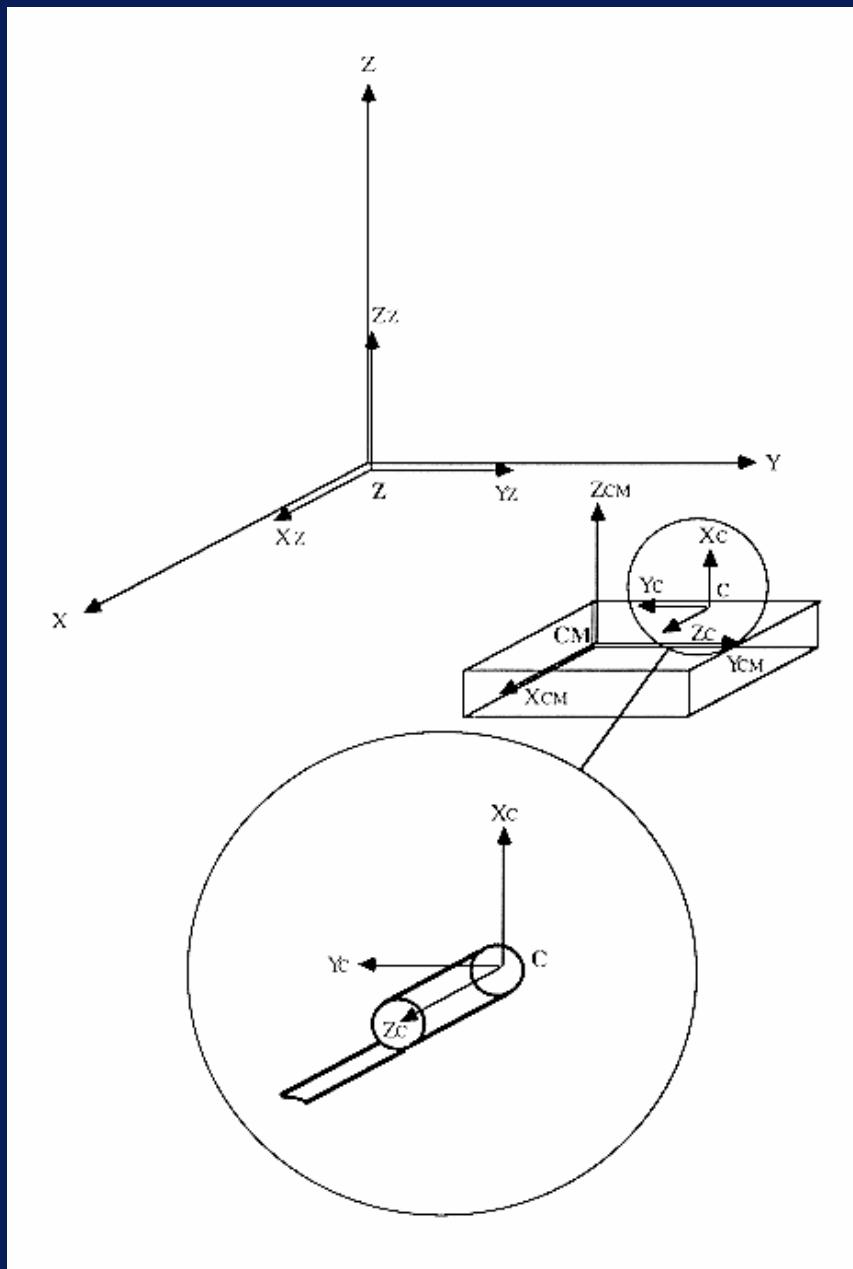
- *CM - the position of the Crimping Machine*

*C : the position of the crimp (ready to be attached), defined with respect to the crimping machine.*

- The origin of  $C$ , representing the crimp, is defined to be at the front of the crimp, of the radial axis; the  $Z$  axis is defined to be coincident with the radial axis (directed in toward the crimp), the  $X$  axis is defined to be directed vertically upward, and the  $Y$  axis makes a right-hand system.

- Thus :
  - $C = Trans(40, 40, 65) Rot(Y, 90) Rot(Y, 180)$

- *C - the position of the crimp*



*CA : the position of the wire-end approaching the crimp, defined with respect to the crimp.*

- CA is a frame embedded in the end of the wire, in exactly the same manner as W, except that it is positioned in front of, *i.e.* approaching, the crimp.

- Thus, as shown in Figure 8.17 (*Slide 102*), CA simply involves a translation of some approach distance in the negative direction of the Z axis of C.

- Since, in a similar manner to WA, we want to approach the crimp along a known path, many approach positions are used such that the translation distance get successively smaller.

- Thus :

$$CA = Trans(0, 0, -(z\_approach)),$$

- where

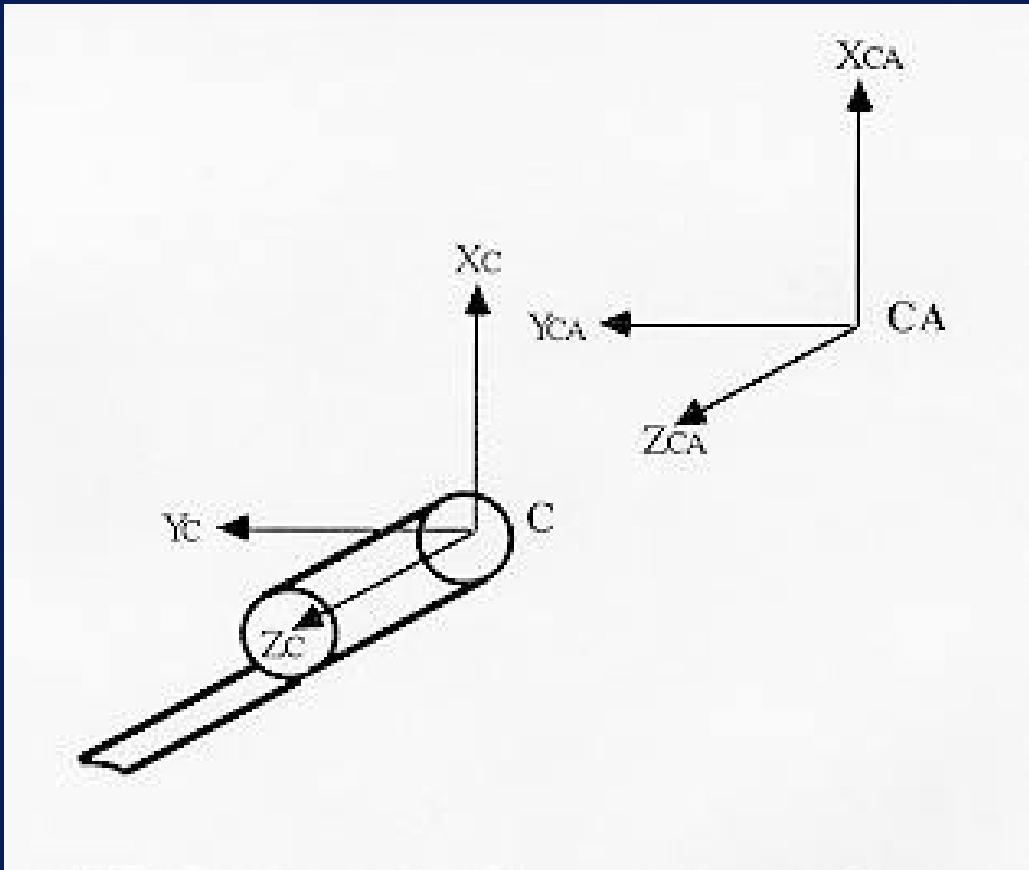
$$z\_approach = z\_approach\_initial,$$

$$z\_approach\_initial - delta,$$

$$z\_approach\_initial - 2 * delta$$

:

0



- CA - *the position of the wire end*
  - *approaching the crimp*

*CC : the position of the wire-end in contact with the crimp, defined with respect to the crimp.*

- Since the frames embedded in the end of the wire and the frame embedded in the crimp align when the wire is in contact with the crimp, this transform is simply the identity transform.

- Had either of these two frames been defined differently,  $CC$  would have been used to define the relationship between the end of the wire and the crimp, which would be, in effect, a series of rotations.

*CI : the position of the wire-end in the crimp,  
defined with respect to the crimp.*

- CI is a frame embedded in the end of the wire, in exactly the same manner as CA, except that it represents the wire inserted in the crimp.

- Thus,  $CI$  simply involves a translation of some insertion distance in the positive direction of the Z axis of  $C$ .
- In a similar manner to  $CA$ , many insertion positions are used such that the translation distances get successively greater.

- Thus :

$$CI = Trans(0, 0, z\_insert)$$

- where

$$z\_insert = 0,$$

$$delta,$$

$$2 * delta,$$

$$\vdots$$

$$z\_insert\_final$$

*CD : the position of the wire-end departed from the crimping machine (the crimp having been attached), defined with respect to the crimp.*

- In a similar manner to CA, CD is defined as a translation in the negative direction of the Z axis of the C frame, except, that is, this case the translation distance becomes successively greater.

- Hence :

$$CD = Trans(0, 0, -(z\_depart)),$$

- where

$$z\_depart = 0,$$

$$delta,$$

$$2 * delta,$$

$$\vdots$$

$$z\_depart\_final$$

# A Simple Robot Programming Language

- To illustrate the use of the frame expression, consider the first move in the wire-crimping task.
- This is represented by a move corresponding to the frame  $T6$ , given by the expression :
  - $T6 = Z^{-1} * OOV * E^{-1}$

- where

$Z = \text{Identity Transform}$

$OOV = \text{Trans} (150, 300, 150) \text{ Rot} (x, -180)$

$E = \text{Trans} (0, -15, 209)$

- Assuming the frames  ${}^T6$ ,  ${}^Z$ ,  ${}^{OOV}$  and  ${}^E$  have been declared, this is written in *RCL* as :

${}^Z := Trans(0, 0, 0);$

${}^{OOV} := Trans(150, 300, 150) * RotX(-180);$

${}^E := Trans(0, -15, 209);$

${}^T6 := Inv({}^Z) * OOV * Inv({}^E);$

- ... and a move to this position is effected by the statement

*Move (^T6);*

# An RCL Implementation of the Wire-crimping Task

```
/* RCL Wire Crimping Program For 6R/600 Manipulator */

/* Frame Declarations */
FRAME ^wire,
 ^wiregrasp,
 ^wireapproach,
 ^wireddepart,
 ^crimp,
 ^crimpapproach,
 ^crimpdepart,
 ^crimpcontact,
 ^crimpinsert,
 ^crimpmachine,
 ^wiredump,
```

```
^centre,
^outofview,
^Z,
^T6,
^effector;

/* the position of the manipulator is coincident */
/* with the base co-ordinate reference frame */

^Z := TRANS(0,0,0);
```

```
/* the end-effector is at the tip of the */
/* wire gripper; a position defined with */
/* respect to the end of the manipulator */

ey := -15;
ez := 195;
^effector := TRANS(0,ey,ez);

/* the position of the end-effector */
/* out of the field-of-view of the camera */

oovx := 150;
oovy := 300;
oovz := 150;
^outofview := TRANS(oovx,oovy,oovz) * RPY(0,0,-180);
```

```
/* the position of the end-effector over */
/* the bin of crimped wires */

wdumpx := 0;
wdumpy := 550;
wdumpz := 160;
^wiredump := TRANS(wdumpx,wdumpy,wdumpz) * RPY(0,0,-135);

/* the position of the end-effector */
/* centred over the tray of wires */

centrex := 0;
centreY := 360;
centreZ := 150;
^centre := TRANS(centrex,centreY,centreZ) *
 RPY(0,0,-180);
```

```
/* the position of the crimping machine */

cmx := 150;
cmy := 300;
cmz := 0;
^crimpmachine := TRANS(cmx,cmy,cmz);

/* the position of the crimp, ready to be */
/* attached; a position defined with */
/* respect to the crimping machine */

cx := 55;
cy := 55;
cz := 85;
^crimp := TRANS(cx,cy,cz) * ROTY(90) * ROTZ(180);
```

```
/* the position of the wire-end in contact with the
crimp

^crimpcontact := TRANS(0,0,0);

/* the position of the wire-end inserted in the crimp
*/

insertionlength := 5;
^crimpinsert := TRANS(0,0,insertionlength);

/* load the robot model of the Smart Arms */
/* 6R/600 manipulator from file */

LOAD_ROB;
```

```
/* load the components of the camera model from file */

LOAD_CAM;

/* incremental distances for point-to-point
approximation
/* to continuous path movement

delta := 3;

/* time delay between gross manipulator */
/* point-to-point movements */

lag := 20;
```

*REPEAT*

```
 /* move out of the field-of-view of the camera */

 ^T6 := INV(^Z) * ^outofview * INV(^effector);
MOVE(^T6);
DELAY(lag);

 /* determine the position and orientation of the */
 /* wire-end and wire grasping point using vision */

WIRE(^wire, ^wiregrasp);
```

```
/* if no error in the robot vision routine */
 /* proceed with the task */
IF errcode = 0
THEN
 /* move to a position above the centre of the tray
 ^T6 := INV(^Z) * ^centre * INV(^effector);
MOVE(^T6);
DELAY(lag);
RELEASE;
```

```

/* when grasping, the end-effector is defined
to be between the jaws of the gripper */

ey := -5;
^effector := TRANS(0,ey,ez);

/* move to an approach point above the grasp
point */

approachdistance := 30;
^wireapproach := TRANS(0,0,-approachdistance);
^T6 := INV(^Z) * ^wire * ^wiregrasp *
 ^wireapproach * INV(^effector);
MOVE(^T6);
DELAY(lag);

```

```

/* move to the grasp point through */
/* successive approach points */

approachdistance := approachdistance - delta;

REPEAT
 ^wireapproach := TRANS(0,0,-
 approachdistance);
 ^T6 := INV(^Z) * ^wire * ^wiregrasp *
 ^wireapproach *
 INV(^effector);
 MOVE(^T6);
 approachdistance := approachdistance -
delta;
UNTIL approachdistance <= 0;

```

```
/* move to the final grasp point and grasp the
 wire */

^T6 := INV(^Z) * ^wire * ^wiregrasp *
 INV(^effector);

MOVE(^T6);
GRASP;

/* move to the depart position through */
/* successive depart points */

departdistance := delta;
```

```

REPEAT
 ^wireddepart := TRANS(0,0,-departdistance);
 ^T6 := INV(^Z) * ^wire * ^wiregrasp *
 ^wireddepart * INV(^effector);
 MOVE(^T6);
 departdistance := departdistance + delta;
UNTIL departdistance > 30;

approachdistance := 40;

/* the end-effector is defined to be at the */
/* inside of the upper jaw of the gripper */
/* now that the wire has been grasped */
ey := -15;
^effector := TRANS(0,ey,ez);
^crimpapproach := TRANS(0,0,-approachdistance);

```

```

/* move to an approach position */
/* in front of the crimp */

^T6 := INV(^Z) * ^crimpmachine * ^crimp *
 ^crimpapproach * ^wiregrasp *
INV(^effector);

MOVE(^T6);
DELAY(lag);

/* bring the wire into contact with the crimp */
/* by moving through successive approach points */
approachdistance := approachdistance - delta;

```

```

REPEAT

 ^crimpapproach := TRANS(0,0,-
 approachdistance);

 ^T6 := INV(^Z) * ^crimpmachine * ^crimp *
 ^crimpapproach * ^wiregrasp *
 INV(^effector);

 MOVE(^T6);

 approachdistance := approachdistance - delta;

UNTIL approachdistance <= 0;

/* final contact position */
^T6 := INV(^Z) * ^crimpmachine * ^crimp *
 ^crimpcontact * ^wiregrasp *
 INV(^effector);

MOVE(^T6);

```

```
/* insert wire in crimp */

^T6 := INV(^Z) * ^crimpmachine * ^crimp *
 ^crimpinsert * ^wiregrasp *
INV(^effector);

MOVE(^T6);

/* actuate the crimping machine */
/* * this is a virtual action * */

DELAY(lag);

/* withdraw with the crimped wire */
/* through successive depart positions */
departdistance := delta;
```

```

REPEAT
 ^crimpdepart := TRANS(0,0,-departdistance);
 ^T6 := INV(^Z) * ^crimpmachine * ^crimp *
 ^crimpdepart * ^wiregrasp *
 INV(^effector);
 MOVE(^T6);
 departdistance := departdistance + delta;
UNTIL departdistance > 35;

/* move to a position above the collection bin
*/
^T6 := INV(^Z) * ^wiredump * INV(^effector);
MOVE(^T6);
DELAY(lag);
RELEASE;

```

```
/* return to the position above the */
/* centre of the tray */

^T6 := INV(^Z) * ^centre * INV(^effector);
MOVE(^T6);
DELAY(lag);

ENDIF;

/* this is repeated until there are no more wires */
/* to be crimped; WIRE returns error code 20 */

UNTIL errcode = 20;
```

# Two Vision Algorithms for Identifying Ends of Wires

# A Binary Vision Algorithm

- If we assume that the wires are well-scattered and lie no more than one or two deep, then all requisite information may be gleaned from the silhouette of the wire and, hence, binary vision techniques can be used.

- The original image is acquired at the conventional 512\*512 pixel resolution with eight bit grey-scale resolution.
- To ensure fast processing and analysis, we first reduce the resolution to 128\*128 pixels resulting in a reduction of the complexity of subsequent operations by a factor of sixteen.

- This reduction is important as the computational complexity of thinning operations is significant.

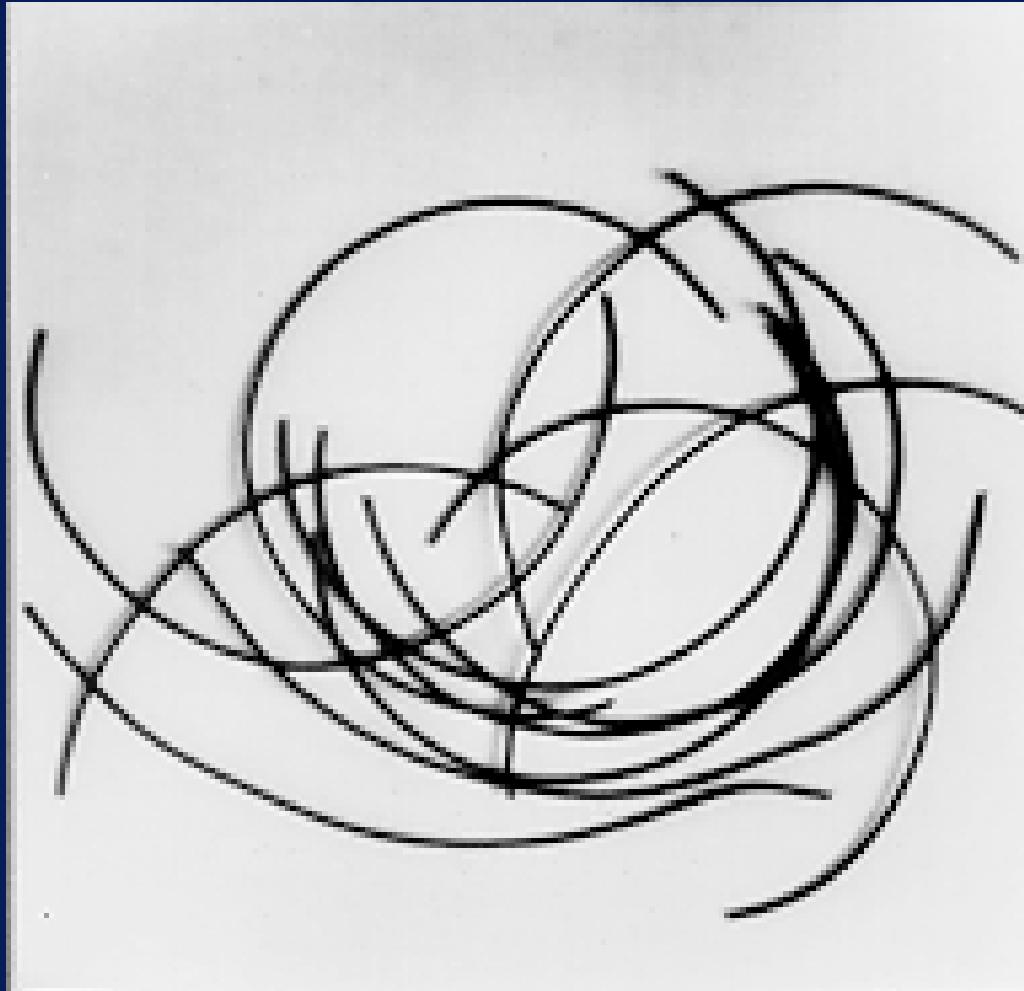
- There are essentially two ways in which this reduced resolution image may be generated :
  - by sub-sampling the original image every fourth column and every fourth line
  - by evaluating the average of pixel values in a  $4*4$  window.

- As we saw in Chapter 4, it is desirable to reduce the image noise and, since this can be accomplished by local averaging, the reduced resolution image in this implementation is generated by evaluating the local average of a 4\*4 (non-overlapping) region in the 512\*512 image.

- This also minimises the degradation in image quality (referred to as *aliasing*) which would result from sub-sampling.

- The image is then segmented by thresholding in the manner discussed in Chapter 4; the threshold value is automatically selected using the approach associated with the Marr-Hildreth Edge Detector.

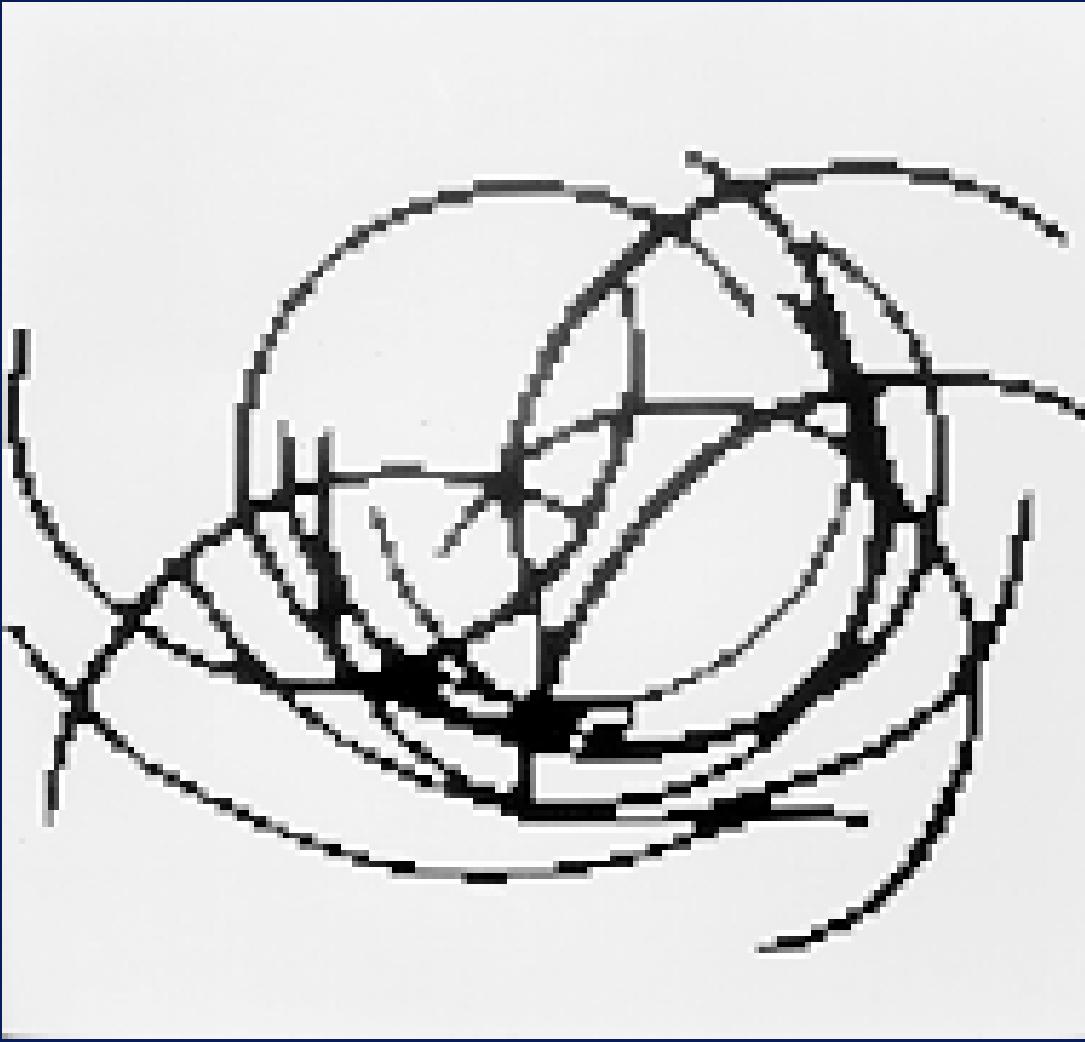
- Figure 8.20 (*Slide 10*) shows the binary image generated by thresholding the original grey-scale image at the automatically determined threshold.



- *512\*512 Image*



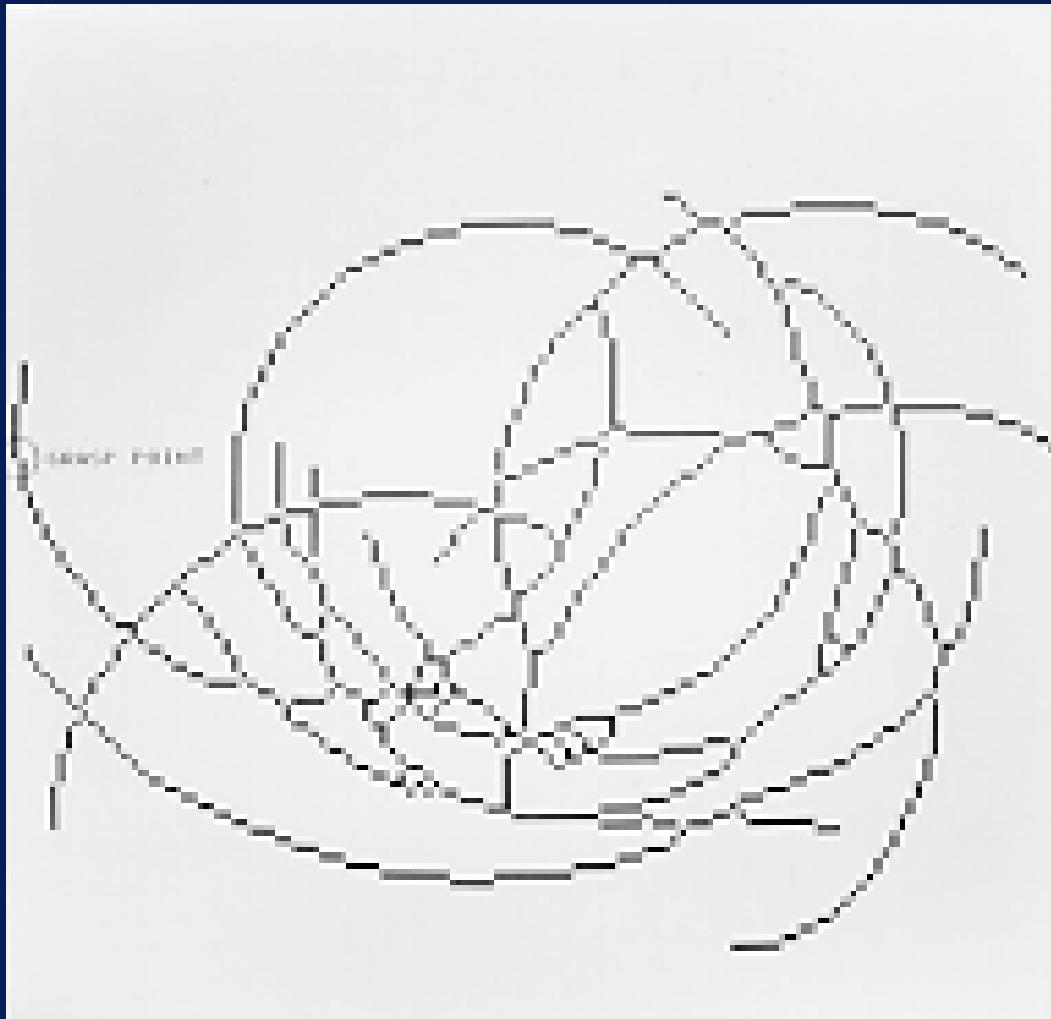
- *128\*128 Image*



- *Binary Image*

- Once the binary image has been generated, the next step is to model the wires in some simple manner.
- The skeleton is a suitable representation for electrical wires, objects which display obvious axial symmetry.

- In this instance, we use the thinning technique described in Section 4.2.3; Figure 8.21 (*Slide 15*) illustrates the application of this thinning algorithm to the binary image shown in Figure 8.20 (*Slide 12*).



- *Thinned Image*

- Having processed the image, we now proceed to its analysis.

- There are essentially two features that need to be extracted from the image :
  - the position of a point at which the robot end-effector should grasp the wire and the orientation of this point on the wire;
  - the position and orientation of the wire-end in relation to the point at which the wire is to be grasped.

- The orientations are required because unless the wire is gripped at right angles to the tangent at the grasp point, the wire will rotate in compliance with the finger grasping force.

- The orientation of the end-point is important when inserting the wire in the crimping-press as the wire is introduced along a path coincident with the tangent to the wire at the end point.

- Based on the skeleton model of the wires, a wire segment may be defined as
  - subsection of a wire bounded at each end by either a wire-crossing or by an arc-end (wire segment end).

- Thus, a wire segment with two valid-end points, at least one of which is an arc-end, and with a length greater than some predefined system tolerance, contains a feasible grasp point.
- This is a point some suitable fixed distance (15mm) from the wire end.

- Once the positions of both the grasp-point and the end-point are known, the orientation or tangential angles of these two points are estimated.
  - The tangent to the wire at the grasp-point is assumed to be parallel to the line joining two skeletal points equally displaced by two pixels on either side of the grasp point.

- The tangent to the wire end is assumed to be parallel to a line joining the end point and a skeletal point three pixels from the end.

# A Grey-Scale Vision Algorithm

- If the organisation of the wires becomes more complex than assumed in the previous section, with many layers of wires occluding both themselves and the background, the required information can no longer be extracted with binary imaging techniques.

- The grey-scale vision system described in this section addresses these issues and facilitates analysis of poor contrast images.
- It is organised as two levels, comprising a peripheral level and a supervisory level.

- All shape identification and analysis is based on boundary descriptors built dynamically by the peripheral level.
- The supervisory level is responsible for overall scheduling of activity, shape description and shape matching.

# The Peripheral Level

- The Peripheral Level corresponds to conventional low-level visual processing, specifically edge detection and the generation of edge and grey-scale information at several resolutions, and segmentation using boundary detection.

- The Prewitt gradient-based edge operator described in Chapter 5 is used as it provides reasonable edges with minimal computational overhead, especially in comparison to other edge operators.

- The edge detector operates on both 256\*256 and 64\*64 resolution images.
- High resolution edge detection is used for image segmentation and low resolution edge detection is used by an area-of-interest operator.

- The ability of any edge detector to segment an image depends on the size of the objects in the image with respect to the spatial resolution of the imaging system.

- The system must be capable of explicitly representing the features (edges) that define the objects, in this case, electrical wires.
- When dealing with long cylinder-like objects, the constraining object dimension is the cylinder diameter.

- At least three pixels are required to unambiguously represent the wire (across the diameter) :
  - one for each edge
  - one for the wire body.

- Using wires of diameter 1.0mm imposes a minimum spatial resolution of 2 pixels/mm or a resolution of 256\*256 for a field of view of 128\*128 mm
- Using a spatial resolution of 1 pixel/mm will tend to smear the object (given that we are reducing the resolution by local averaging and not by sub-sampling).

- Edge detection tests at this resolution showed that such smearing does not adversely affect the boundary/feature extraction performance if the wire is isolated (*i.e.* the background is clearly visible) but in regions of high occlusion where there are many wires in close proximity the edge or boundary quality does degrade significantly.

- Tests using a spatial resolution of 0.5 pixels/mm indicated that a detector's ability to segment the image reliably is severely impaired in most situations.

- There are several approaches which may be taken to boundary building; this system uses a dynamic contour following algorithm and is the same one described in detail in Chapter 5.

- As the algorithm traces around the boundary, it builds a ***Boundary Chain Code (BCC)*** representation of the contour;  
See Chapter 7.
- The complete BCC represents the segmented object boundary and is then passed to the supervisory level for analysis.



- *Boundary Following*

- The disadvantage of the contour following technique is that, because the algorithm operates exclusively on a local basis using no *a priori* information, the resulting segmentation may not always be reliable and the resulting contour may not correspond to the actual object boundary.

- In particular, the presence of shading and shadows tends to confuse the algorithm.

# The Supervisory Level

- The Supervisory Phase is concerned with
  - overall scheduling of activity within the vision system
  - the transformation and analysis of the boundaries passed to it by the peripheral level.

- An interest operator was used which identifies a sequence of sub-areas within the image, ordered in descending levels of interest.

- This operator is based on the analysis of the edge activity in a reduced resolution image;
- allows the system to avoid cluttered areas with many (occluding) wires and concentrate on points of low scene population which are more likely to contain isolated and accessible wires.

- the area of interest is one sixteenth of the size of the original image and is based on a 4\*4 division of a 64\*64 pixel resolution image;
- extract a contour, representing the boundary of a group of wires
- in a specific area of interest in the image;

- analyse this boundary to determine whether or not it contains a boundary segment describing a wire-end.

- What is required of the supervisory processes is to ascertain which part of the contour, if any, corresponds to the wire-end template and to subsequently determine the position and orientation of both the end of the wire and a suitable grasp-point.

- As we noted in Chapter 7, the use of BCC-based shape descriptors to identify shapes is not reliable and, instead, the wire-end is identified by heuristic analysis, formulated as follows.

- A boundary segment characterising a wire-end is defined to be
  - a short segment (20 units in length) in which the boundary direction at one end differs by  $180^\circ$  from the direction at the other end
  - and in which the distance between the end points is less than or equal to 5 units.

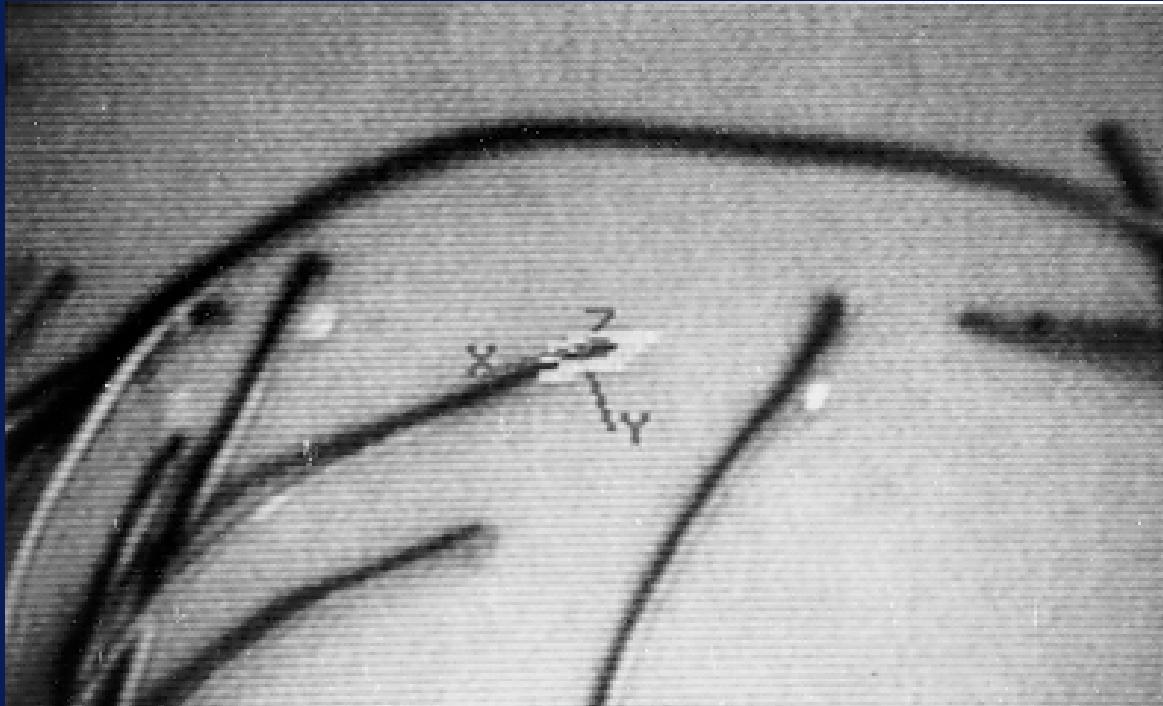
- In addition, the wire-end should be isolated, *i.e.* there should be no neighbouring wires which might foul the robot end-effector when grasping the wire.

- This condition is identified by checking that the edge magnitude in the low resolution image in a direction normal to the boundary direction is less than the usual threshold used by the edge detection process.

# The Vision/Manipulator Interface

- Once the wire-end shape has been identified, it is necessary to determine the components of the homogenous transformations representing the two frames,  $W$  and  $WG$ , which denote the position and orientation of the wire-end and the position and orientation of the grasp position with respect to the wire-end.

- In the task specification discussed above, we defined the origin of the wire frame  $W$  to be at the end of the wire.



- *Identification of a wire-end*
  - *(with W frame attached)*

- Z axis aligned with wire's axis of symmetry, directed away from the end.
- $X$  axis of  $W$  was defined to be normal to the tray on which the wires lie (and, hence, is normal to image plane) directed vertically upwards.
- The  $Y$  axis makes up a right-hand system.

- The origin of the wire gripper frame  $WG$  was defined to be located on the  $Z$  axis of the  $W$  frame, in the negative  $Z$  direction, and located a short distance from the origin of  $W$ .
- The  $Z$  axis of  $WG$  is defined to be normal to the plane of the tray, directed downwards.

- The  $Y$  axis is defined to be normal to the axis symmetry of the wire, in plane of the tray. The  $X$  axis makes up a right hand system.

- To determine the components of the frame  $W$ , we only need to identify the position of the end of the wire and orientation of the axis of symmetry of the wire at its end, which gives us the direction of the Z axis.

- The  $Y$  axis is at right angles to it and the  $X$  axis has already been defined.

- Similarly, we only need to identify the orientation of the axis of symmetry of the wire at the grasp point to determine  $WG$ ;
  - this gives us the direction of the  $X$  axis
  - the  $Y$  axis is at right angles to it and
  - the  $Z$  axis has already been defined.

- The main problem at this stage is that any orientation and position will be computed in the image frame of reference, *i.e.* using pixel coordinates.
- This is not satisfactory since the robot task specification is formulated in the real world frame of reference.

- Obviously, the relationship between these two reference frames must be established.
- Once it is, we can transform the relevant image position (the end of the wire and other points on its axis) to the real world frame of reference and then compute the required orientations.

# The Camera Model and the Inverse Perspective Transformation

- For any given optical configuration, there are two aspects to the relationship :
  - the *camera model*, which maps a 3D world point to its corresponding 2D image point
  - the *inverse perspective transformation*, which is used to identify the 3D world point(s) corresponding to a particular 2D image point.

- Since the imaging process is a projection (from a 3D world to a 2D image), the inverse process, *i.e.* the inverse perspective transformation, cannot uniquely determine a single world point for a given image point

- the inverse perspective transformation thus maps a 2D image point into a line (an infinite set of points) in the 3D world,
- however, it does so in a useful and well-constrained manner.

- For the following, we will assume that the camera model (and, hence, the inverse perspective transformation) is linear; this treatment closely follows that of [Ballard and Brown 1982].

- Details of non-linear models can be found in the references to camera models in the bibliography at the end of the chapter.

# The Camera Model

- Let the image point in question be given by the co-ordinates which, in homogenous co-ordinates is written .  
$$\begin{bmatrix} u \\ v \\ t \end{bmatrix}$$
- Thus,  $U = \frac{u}{t}$
- and  $V = \frac{v}{t}$

- Let the desired camera model, a transformation which maps the 3D world point to the corresponding 2D image point, be  $C$ .

- Thus,

$$C \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ t \end{bmatrix}$$

- Hence  $C$  must be a 3\*4 (homogenous) transformation :

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix}$$

- and

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ t \end{bmatrix}$$

- Expanding this matrix equation, we get :

$$C_{11}x + C_{12}y + C_{13}z + C_{14} = u \quad (1)$$

$$C_{21}x + C_{22}y + C_{23}z + C_{24} = v \quad (2)$$

$$C_{31}x + C_{32}y + C_{33}z + C_{34} = t \quad (3)$$

- but  $u = Ut$  and  $v = Vt$
- so  $u - Ut = 0 \quad (4)$
- $v - Vt = 0 \quad (5)$

- Substituting (1) and (3) for  $u$  and  $t$ , respectively, in (4) and substituting (2) and (3) for  $v$  and  $t$ , respectively, in (5) :

$$C_{11}x + C_{12}y + C_{13}z + C_{14} - UC_{31}x - UC_{32}y - UC_{33}z - UC_{34} = 0 \quad (6)$$

$$C_{21}x + C_{22}y + C_{23}z + C_{24} - VC_{31}x - VC_{32}y - VC_{33}z - VC_{34} = 0 \quad (7)$$

- If we establish this association (*i.e.* if we measure the values of  $x$ ,  $y$ ,  $z$ ,  $U$  and  $V$ ), we will have two equations in which the only unknowns are the twelve camera model coefficients (and which we require).

- Since a single observation gives rise to two equations, six observations will produce twelve simultaneous equations which we can solve for the required camera coefficients  $C_{ij}$ .

- Remember that these two equations arose from the association of a particular world point  $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$  with a particular and corresponding image point  $\begin{bmatrix} u \\ v \\ t \end{bmatrix}$ .

- Before we proceed, however, we need to note that the overall scaling of  $C$  is irrelevant due to the homogenous formulation and, thus, the value of  $C_{34}$  may be set arbitrarily to 1 and we can re-write (6) and (7), completing the equations so that terms for each coefficient of  $C$  is included, as follows :

$$C_{11}x + C_{12}y + C_{13}z + C_{14} + C_{21}0 + C_{22}0 + C_{23}0 + C_{24}0 - UC_{31}x - UC_{32}y - UC_{33}z = U$$
$$C_{11}0 + C_{12}0 + C_{13}0 + C_{14}0 - C_{21}x + C_{22}y + C_{23}z + C_{24} - VC_{31}x - VC_{32}y - VC_{33}z = V$$

- This reduces the number of unknowns to eleven.

- For six observations, we now have twelve equations and eleven unknowns: *i.e.* the system of equations is over-determined.
- Re-formulating the twelve equations in matrix form, we can obtain a least-square-error solution to the system using the pseudo-inverse method which we described in Chapter 4.

• Let  $X = \begin{bmatrix} x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^1x^1 & -U^1y^1 & -U^1z^1 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^1x^1 & -V^1y^1 & -V^1z^1 \\ x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^2x^1 & -U^2y^2 & -U^2z^2 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^2x^2 & -V^2y^2 & -V^2z^2 \\ x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^2x^2 & -U^3y^3 & -U^3z^3 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^3x^3 & -V^3y^3 & -V^3z^3 \\ x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^4x^4 & -U^4y^4 & -U^4z^4 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^4x^4 & -V^4y^4 & -V^4z^4 \\ x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^5x^5 & -U^5y^5 & -U^5z^5 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^5x^5 & -V^5y^5 & -V^5z^5 \\ x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^6x^6 & -U^6y^6 & -U^6z^6 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^6x^6 & -V^6y^6 & -V^6z^6 \end{bmatrix}$

$$c = \begin{bmatrix} C_{11} \\ C_{12} \\ C_{13} \\ C_{14} \\ C_{21} \\ C_{22} \\ C_{23} \\ C_{24} \\ C_{31} \\ C_{32} \\ C_{33} \end{bmatrix} \quad y = \begin{bmatrix} U^1 \\ V^1 \\ U^2 \\ V^2 \\ U^3 \\ V^3 \\ U^4 \\ V^4 \\ U^5 \\ V^5 \\ U^6 \\ V^6 \end{bmatrix}$$

- where the trailing superscript denotes the observation number.

†In general, it is better to significantly over-determine the system of equations by generating a larger set of observations than the minimal six.

- Then :  $c = (X^T X)^{-1} X^T y$
- $= X^T y$
- We assumed above that we make six observations to establish the relationship between six sets of image co-ordinates and six sets of real world co-ordinates. †

- This is, in fact, the central issue in the derivation of the camera model, that is, the identification of a set of corresponding *control points*.

- There are several approaches.
  - we could present the imaging system with a calibration grid
  - empirically measure the positions of the grid intersections
  - identifying the corresponding points in the image, either interactively or automatically.

- The empirical measurement of these real-world co-ordinates will be prone to error and this error will be manifested in the resultant camera model.

- It is better practice to get the robot itself to calibrate the system
  - by fitting it with an end-effector with an accurately located calibration mark (*e.g.* a cross-hair or a surveyor's mark)
  - by programming it to place the mark at a variety of positions in the field of view of the camera system.

- *The main benefit of this approach is that the two components of the manipulation environment, the robot and the vision system, both of which are reasoning about co-ordinates in the 3D world, are effectively coupled and, if the vision system “sees” something at a particular location, that is where the robot manipulator will go.*

# The Inverse Perspective Transformation.

- Once the camera model  $C$  has been determined, we are now in a position to determine an expression for the co-ordinates of a point in the real world in terms of the co-ordinates of its imaged position.

- Recalling equations (1) - (5) :

$$C_{11}x + C_{12}y + C_{13}z + C_{14} = u = Ut$$

$$C_{21}x + C_{22}y + C_{23}z + C_{24} = v = Vt$$

$$C_{31}x + C_{32}y + C_{33}z + C_{34} = t$$

- Substituting the expression for  $t$  into the first two equations gives :

$$U(C_{31}x + C_{32}y + C_{33}z + C_{34}C_{11}x) = C_{11}x + C_{12}y + C_{13}z + C_{14}$$

$$V(C_{31}x + C_{32}y + C_{33}z + C_{34}C_{11}x) = C_{21}x + C_{22}y + C_{23}z + C_{24}$$

- Hence :

$$(C_{11} - UC_{31})x + (C_{12} - UC_{32})y + (C_{13} - UC_{33})z + (C_{14} - UC_{34}) = 0$$

$$(C_{21} - VC_{31})x + (C_{22} - VC_{32})y + (C_{23} - VC_{33})z + (C_{24} - VC_{34}) = 0$$

- Letting :  
 $a_1 = C_{11} - UC_{31}$
- $b_1 = C_{12} - UC_{32}$
- $c_1 = C_{13} - UC_{33}$
- $d_1 = C_{14} - UC_{34}$

- and

$$a_2 = C_{21} - VC_{31}$$

$$b_2 = C_{22} - VC_{32}$$

$$c_2 = C_{23} - VC_{33}$$

$$d_2 = C_{24} - VC_{34}$$

- we have :

$$a_1x + b_1y + c_1z + d_1 = 0$$

$$a_2x + b_2y + c_2z + d_2 = 0$$

- These two equations are, in effect, equations of two planes; the intersection of these planes determines a line comprising the set of real-world points which project onto the image point  $\begin{bmatrix} U \\ V \end{bmatrix}$ .

- Solving these plane equations simultaneously (in terms of  $z$ ) :

$$x = \frac{z(b_1c_2 - b_2c_1) + (b_1d_2 - b_2d_1)}{(a_1b_2 - a_2b_1)}$$

$$y = \frac{z(a_2c_1 - a_1c_2) + (a_2d_1 - a_1d_2)}{(a_1b_2 - a_2b_1)}$$

- Thus, for any given  $z_0$ ,  $U$  and  $V$ , we may determine the corresponding  $x_0$  and  $y_0$ ,
- *i.e.* the real-world co-ordinates.

# Recovery of the Third Dimension

- The camera model and the inverse perspective transformation which we have just discussed allow us to compute the  $x$  and  $y$  real-world co-ordinates corresponding to a given position in the image.

- However, we must assume that the  $z$  coordinate, *i.e.* the distance from the camera, is known.
- For the wire crimping application in which the wires lie on a table at a given and constant height (*i.e.* at a given  $z_0$ ), this is quite adequate.

- In general, however, we will not know the coordinate of the object in the third dimension and we must recover it somehow.

- The purpose of this section is to show how we can compute  $z_0$ 
  - if we have a second image of the scene, taken from another viewpoint,
  - if we know the image co-ordinates of the point of interest (e.g. the wire end) in this image also.

- In this instance, we have two camera models and, hence, two inverse perspective transformations.
- Instead of solving two plane equations simultaneously, we solve four plane equations.

- In particular, we have :

$$a_1x + b_1y + c_1z + d_1 = 0$$

$$a_2x + b_2y + c_2z + d_2 = 0$$

$$p_1x + q_1y + r_1z + s_1 = 0$$

$$p_2x + q_2y + r_2z + s_2 = 0$$

- and  $C1_{ij}$  and  $C2_{ij}$  are the coefficients of the camera model for the first and second images, respectively.

- Similarly,  $U_1, V_1$  and  $U_2, V_2$  are the coordinates of the point of interest in the first and second images, respectively.
- Since we now have four equations and three unknowns, the system is over-determined and we compute a least-square-error solution using the pseudo-inverse technique discussed in Chapter 4.

- where  $a_1 = C1_{11} - U1C1_{31}$        $a_2 = C1_{21} - V1C1_{31}$   
 $b_1 = C1_{12} - U1C1_{32}$        $b_2 = C1_{22} - V1C1_{32}$   
 $c_1 = C1_{13} - U1C1_{33}$        $c_2 = C1_{23} - V1C1_{33}$   
 $d_1 = C1_{14} - U1C1_{34}$        $d_2 = C1_{24} - V1C1_{34}$   
 $p_1 = C2_{11} - U2C2_{31}$        $p_2 = C2_{21} - V2C2_{31}$   
 $q_1 = C2_{12} - U2C2_{32}$        $q_2 = C2_{22} - V2C2_{32}$   
 $r_1 = C2_{13} - U2C2_{33}$        $r_2 = C2_{23} - V2C2_{33}$   
 $s_1 = C2_{14} - U2C2_{34}$        $s_2 = C2_{24} - V2C2_{34}$

- It should be noted that the key here is not so much the mathematics which allows us to compute  $x_0$ ,  $y_0$  and  $z_0$  but, rather, the image analysis by which we identify the corresponding point of interest in the two images.

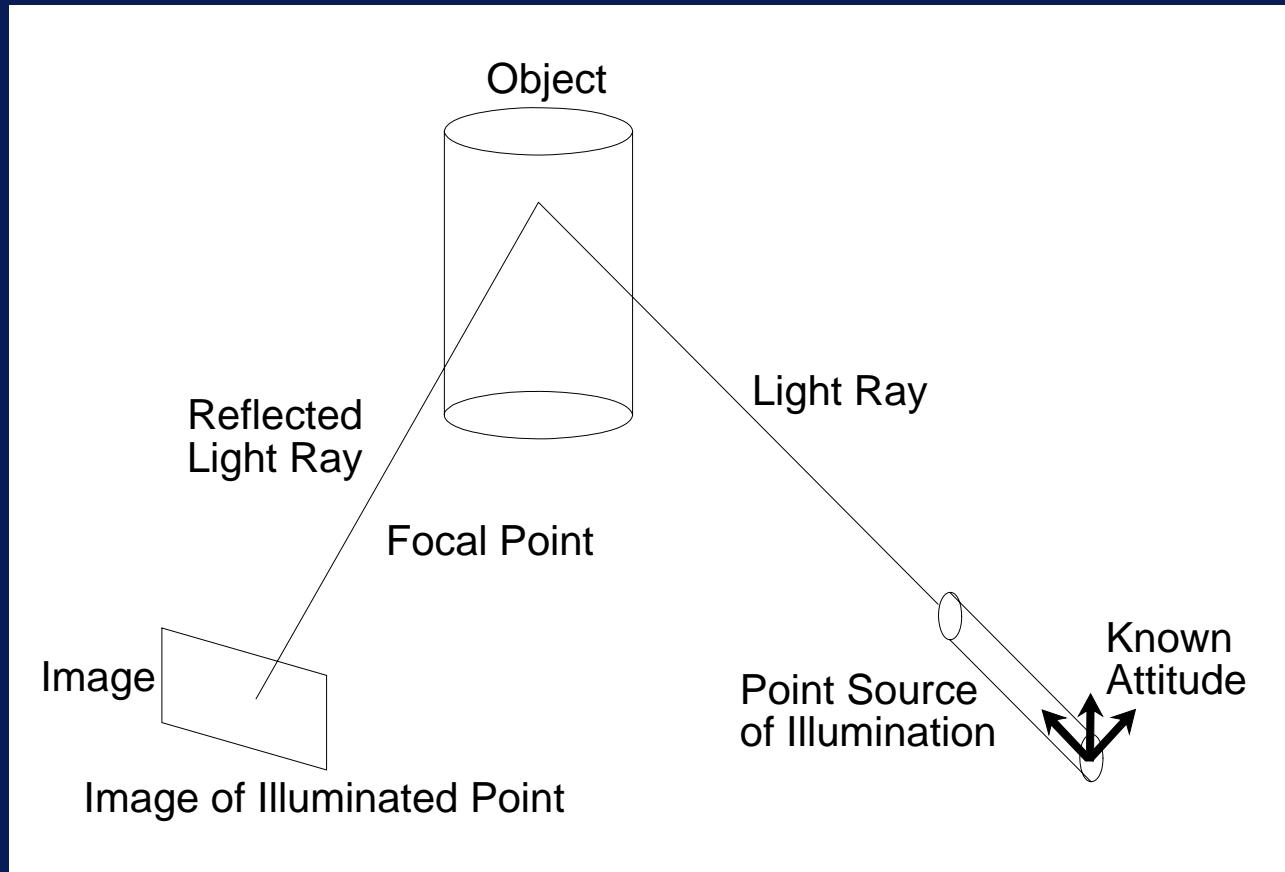
- It is this correspondence problem which lies at the heart of most of the difficulties in recovery of depth information.
- To complete the chapter, the next section describes a simple, popular and useful technique for analysing images and computing depth information.

# 3D Vision using Structured Light

- *Active ranging using structured light is one of the most popular of ranging techniques in industrial vision.*

- The essential idea is to illuminate the object in such a way so that :
  - a) we know the position and direction of the source of illumination;
  - b) the point being illuminated is easily identifiable (*e.g.* we illuminate a very small part of the surface of the object with a dot of light);

- c) we know the position of the sensor (camera) and can compute the direction to the illuminated part of the object surface.



- *Active Triangulation*

- Thus, we can draw two lines
  - one along the ray of illumination from the light source to the object surface,
  - the other from the position of the sensed illumination on the image through the focal point to the object surface.

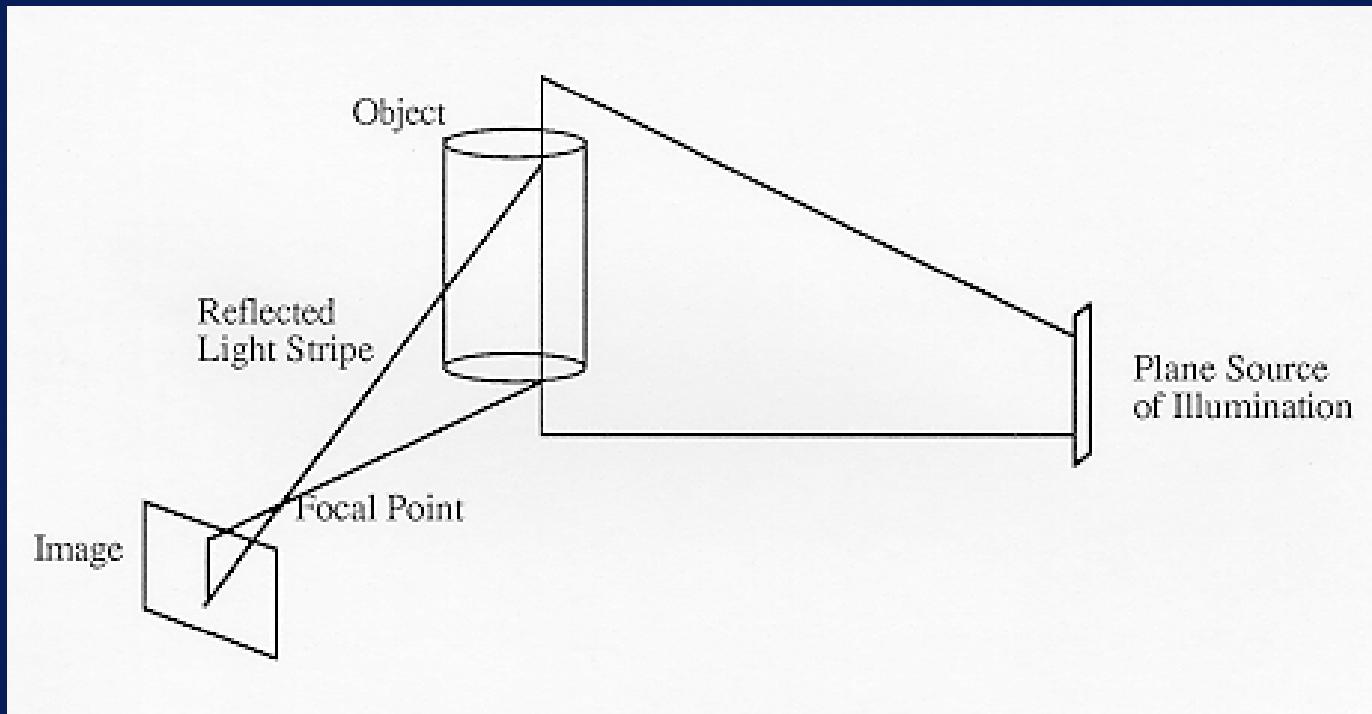
- The object surface is at the intersection of these two lines (see Figure 8.24) (*Slide 49*) and to compute the 3D position of this point on the surface of the object, we just have to compute the point of intersection of these two lines.

- The basis of the approach is that it solves, in a simple but contrived way, the correspondence problem to which we alluded in the previous section.
- The solution is not without problems, however.

- Since this approach will yield the range to only one small point on the object's surface, we either have to :
  - scan the surface with the dot of light,
  - computing the range at each point
  - or illuminate more than one point at the same time.

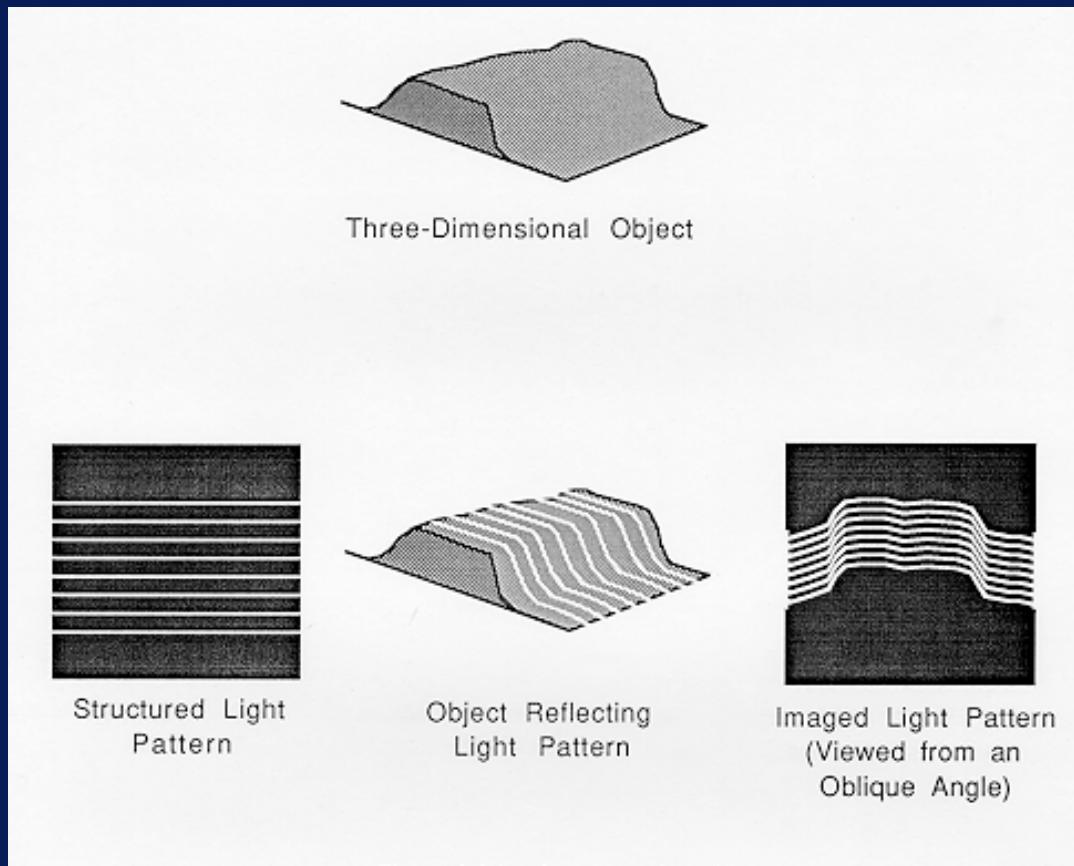
- The former approach is not normally employed, except in specialised laser-scanning applications, since one would have a significant overhead in image acquisition.

- The latter approach popularly utilises stripes of light (and is hence referred to as *Light Striping*) or grids of light to illuminate the object.



- *Light Striping*

- In these cases, the derivation of range involves the computation of the point of intersection of the plane of light and the line from an image point on the sensed line through the focal point (see Figures 8.25 and 8.26) (*Slides 56 and 58*).



- *Structured Light*

- If we calibrate the vision system and determine the camera model then, for any point in the image, we can derive the equation of a single line in the real world.
- To identify the co-ordinates of the single point which reflected light causing the imaged point, we need an additional constraint.

- One such constraint might be the knowledge that the real-world point lies on some plane (which is not coincident with the line of sight).
- For example, in section 8.6 where we derived the inverse perspective transformation, we assumed that the point lay on a plane given by  $z = z_0$ .

- In the case of light stripping, use the same idea and illuminate the object with a single plane of light and, if we know the equation of this plane, then the identification of the 3D world point co-ordinates simply requires the computation of the intersection of the line of sight (given by the inverse perspective transformation) and this plane.

- In order to determine the equation of the light plane, one can locate several points on it, identify their 3D co-ordinates and fit a plane through these points.
- One simple way of identifying points in the plane of light is to place blocks of different known heights on the work-surface in the path of the light beam.

- Knowing the real  $z$  co-ordinate
  - the real world  $x$  and  $y$  co-ordinates of points on the resulting stripe are then computed
  - by imaging the stripe and applying the inverse perspective transformation of the camera to the measured points.

- Having identified a number ( $M$ , say) of points on the plane at several different heights, one can use these  $x$ ,  $y$  and  $z$  values to generate a set of simultaneous plane equations :

$$a_1x_i + a_2y_i + a_3z_i + a_4 = 0, \quad i = 1..M$$

- ... and solve them using the pseudo-inverse method. Unfortunately, this equation has a degenerate solution in which all the coefficients are zero.

- To avoid this possibility, we can reformulate (1) [from Bolles et al. 1981] by dividing across by  $a_3$  and letting :

$$\frac{a_1}{a_3} = b_1$$

$$\frac{a_2}{a_3} = b_2$$

$$\frac{a_4}{a_3} = b_3$$

- Thus :

$$b_1x_i + b_2y_i + b_3z_i + b_4 = 0$$

- and hence :

$$b_1x_i + b_2y_i + b_3 = -z_i$$

- A least square error solution to this set of equations, written in matrix form as :

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ \dots & \dots & \dots \\ x_M & y_M & 1 \end{bmatrix} * \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} -z_1 \\ -z_2 \\ -z_3 \\ \dots \\ -z_M \end{bmatrix} \quad M > 3$$

- ... can now be generated using the pseudo-inverse method.
- The only restriction in this case is that the plane of light cannot be normal to the  $z$  axis since an equation of this form cannot be used to represent such a plane (*i.e.* cannot be constant)

- The equation of the plane of light is thus

$$b_1 x_i + b_2 y_i + z_i + b_3 = 0$$

- and the 3D position of a point in an imaged light stripe can be found by solving the set of simultaneous equations given by the two plane equations provided by the inverse perspective transformation :

$$x(C_{11} - UC_{31}) + y(C_{12} - UC_{32}) + z(C_{13} - UC_{33}) = UC_{34} - C_{14}$$

$$x(C_{21} - VC_{31}) + y(C_{22} - VC_{32}) + z(C_{23} - VC_{33}) = VC_{34} - C_{24}$$

- and the light plane :

$$b_1x + b_2y + z = -b_3$$

- The plane of light can be generated either using a laser scanner or it can be generated by projecting a collimated light source through a slit.

- The advantage of using a laser is that it can illuminate the object in the presence of ambient lighting, *e.g.* by using an infra-red laser and an infra-red sensitive sensor (CCD sensor).

- The slit projection approach will typically require a somewhat darkened environment or an extremely bright light source.

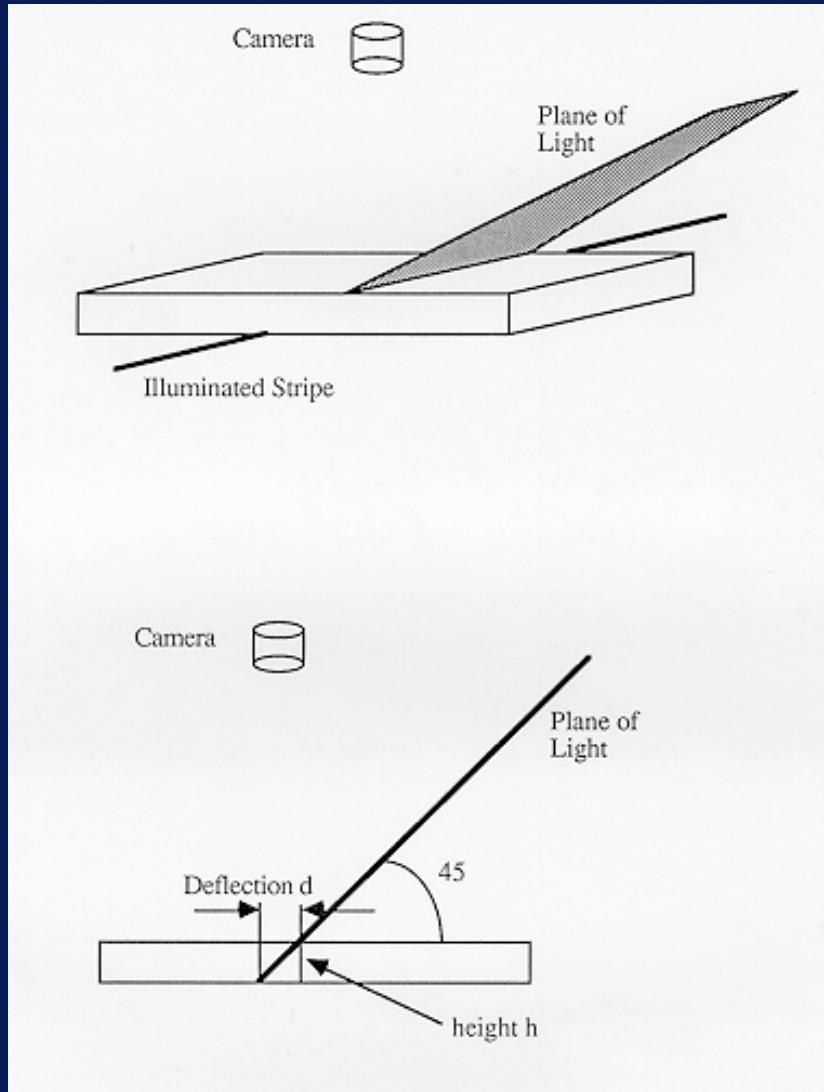
- Furthermore, this approach suffers from a problem common to all so-called triangulation systems of this type : that only surface points which are visible from both illumination source and sensor can be used to yield range measurements.

- Hence, hidden parts in concavities will cause some problems.

- As a final note, it is worth remarking that this structured light approach is quite general in the sense that it allows you to generate all three real-world co-ordinates for points on an imaged light stripe.

- If you are only interested in deriving the height of the object rather than its range, then you can adopt a simpler approach.

- Consider a plane of light which is incident to the work surface at an angle of  $45^\circ$  (see Figure 8.27) (*Slide 80*).
- An object on the work-surface in the path of the beam will cause the illuminated stripe to be deflected by an amount which is proportional to the height of the block.



- *Height Measurement using Light Stripes*

- An object on the work-surface in the path of the beam will cause the illuminated stripe to be deflected by an amount which is proportional to the height of the block.
- In fact, for the example shown, the deflection will be equivalent to the height of the block (in an image frame of reference).

- Thus, to measure the height you merely have to calibrate the system by computing the relationship between a deflection  $d$  and a height  $h$  in the real-world (using a block of known height) and subsequently measure deflections.

# Introduction to Image Understanding

- Representations and information processing: from images to object models.
- *Image Understanding must take into consideration the considerable loss of information which arises when a 3D world is imaged and represented by 2D digital images.*

- There is, however, more to image understanding than just the recovery of depth information.

- If we are going to be able to identify the structure of the environment, we need to do more than develop a 3D range map since this is still an image and the information which we need is still *implicit*.

- We require an *explicit* representation of the structure of the world we are imaging.

- Hence, we still need the processes of segmentation and object recognition that we dealt with in the previous chapters.
  - The data we are dealing with is much more complex than before;
  - the simplistic image segmentation, template matching or feature extraction and classification paradigms are wholly inadequate.

- To be able to proceed from raw 2D images of the world to explicit 3D structural representation we must
  - adopt a much more sophisticated stance,
  - acknowledge that no single process or representation is going to be generally adequate.

- We require intermediate representations to bridge the gap between raw images and the abstracted structural model.
  - These representations should make different kinds of knowledge explicit,
  - should expose various kinds of constraint upon subsequent interpretations of the scene.

- It is the progressive integration of these representations
- and their mutual constraint
- to facilitate an unambiguous interpretation of the scene that most characterises Image Understanding.

- In summary then, we can characterise Image Understanding as :
  - a sequence of processes
  - concerned with successively extracting visual information from one representation (beginning with digital images)...

- organising it
- and making it explicit in the representation to be used by other processes.

- *Vision is computationally modular and sequential.*

# Organisation of Visual Processes

- At present, it is not clear how information in one representation should influence the acquisition and generation of information in another representation.

- Some possibilities include :
  - A bottom-up flow of data in which information is made explicit without recourse to a priori knowledge. *Thus, we form our structural representation purely on the basis of the data implicit in the original images.*

- Heterarchical Constraint Propagation.
  - » This is similar to the bottom-up approach
  - » but we now have the additional constraint that cues, *i.e.* a given information representation, at any one level of the hierarchical organisation of representations can mutually interact to delimit and reduce the possible forms of interpretation at that level.

- Perhaps one of the simplest examples is the integration of depth values generated by two independent processes such as binocular stereo and the parallax caused by a moving camera.
- A top-down, *model driven*, information flow whereby early vision is guided by firm expectations of what is to be seen.

- It should be noted that this model-based vision is quite different from the knowledge-based approach which was fashionable in artificial intelligence a decade ago :

- in which the effort was to design control processes that could utilise appropriate knowledge at the appropriate time
- with inadequate attention being paid to the representation used, the processes using them, and, indeed, the reasons for utilising those representations.

- David Marr, at MIT, exerted a major influence on the development of this new computational approach to vision.

- Marr modelled the vision process as:
  - an information processing task
  - in which the visual information undergoes different hierarchical transformations at and between levels,
  - generating representations which successively make more three-dimensional features explicit.

- There are three distinct levels at which the problem must be understood :
  - a computational level,
  - a representational and algorithmic level,
  - and an implementation level.

- The computational level is concerned with the “what” and the “why” of the problem: what is to be computed, why it is to be computed and what is the best strategy for computing it?

- The representational and algorithmic level addresses the “how” of the problem: how is this theory to be implemented?

- This necessitates a representation for the input and the output.
  - The third level, hardware implementation, addresses the problem of how to physically realise these representations and algorithms.

- Thus, this approach removes one from the image environment and addresses the more fundamental problems of the task, formulating a theory of computation and then putting this theory into effect by applying the available visual abilities.

- *Attempting to understand perception solely from the neurological standpoint is as fruitless as trying to understand bird-flight from the study of feathers; one must first understand aerodynamics, only then does the structure of the feathers make sense.*

# Visual Representations

- *The Raw Primal Sketch*
- Beginning with a grey-level image, Marr proposed the generation of a Raw Primal Sketch, which consists of primitives of
  - edges, terminations, blobs and bars
- at different scales.

- Each primitive has certain associated properties :
  - orientation
  - width
  - length
  - position
  - strength

- The computation of the raw primal sketch requires both :
  - the measurement of intensity gradients of different scale and
  - the accurate measurement of location of these changes.

- This causes a problem, however, since no single measuring device can be optimal simultaneously at all scales.

- For example, a watchmaker will use a micrometer for very fine measuring work and a callipers for coarse work; both of which are useless to a surveyor measuring up a building site.

- Thus, we need an edge detector
  - which can operate at different scales and
  - is well-localised in the frequency domain, that is, the spatial frequencies (the rate of variation of image intensity with distance) to which it is sensitive are confined to given ranges.

- Since edges are also localised in space (*i.e.* in the image), this means that the measuring device must also be spatially localised.

- Obviously, the requirements of spatial localisation and confinement in spatial frequency are at variance :
  - devices for measuring variation over large distances must themselves be large.

- Marr and Hildreth proposed a compromise.

- They suggested that one should use smoothing filters to select information in the grey-level intensity image at different scales, but choose a type of smoothing that optimises these two opposing demands (of spatial frequency and positional localisation).

- The Gaussian distribution is the only appropriate smoothing distribution with these properties.

- Marr suggested that the raw primal sketch should be generated using the Marr-Hildreth theory of edge detection (which we discussed in Chapter 5)
  - by convolving the image with the Laplacian-of-Gaussian functions,
  - each Gaussian having a different standard deviation (hence the facility for analysis at different scales).

- Points at which the resultant image go from positive to negative, *i.e.* zero-crossings, are isolated; see Figures 9.1 to 9.4
- These points correspond to instances of sharp intensity change in the original image.



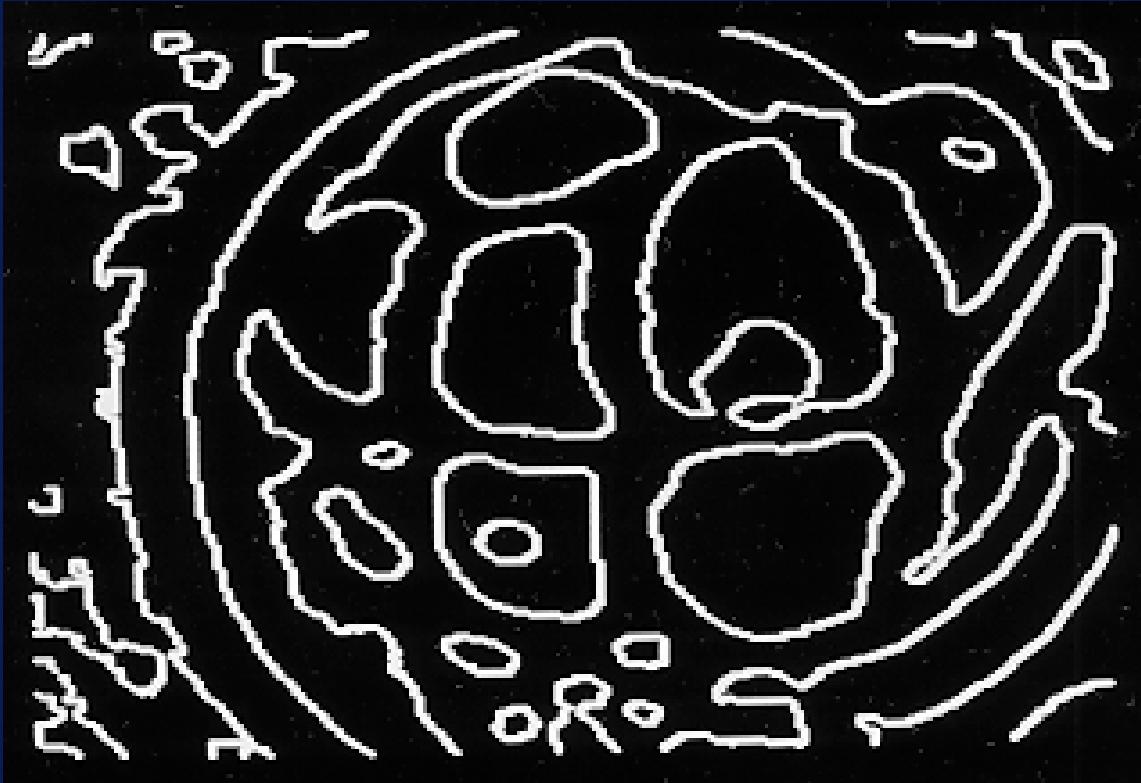
- *A grey-scale image*



- *Zero-crossings derived by convolving the image with a Laplacian of Gaussian filter in which the standard deviation of the Gaussian is 3.0 pixels.*
  - **All Zero-Crossings**



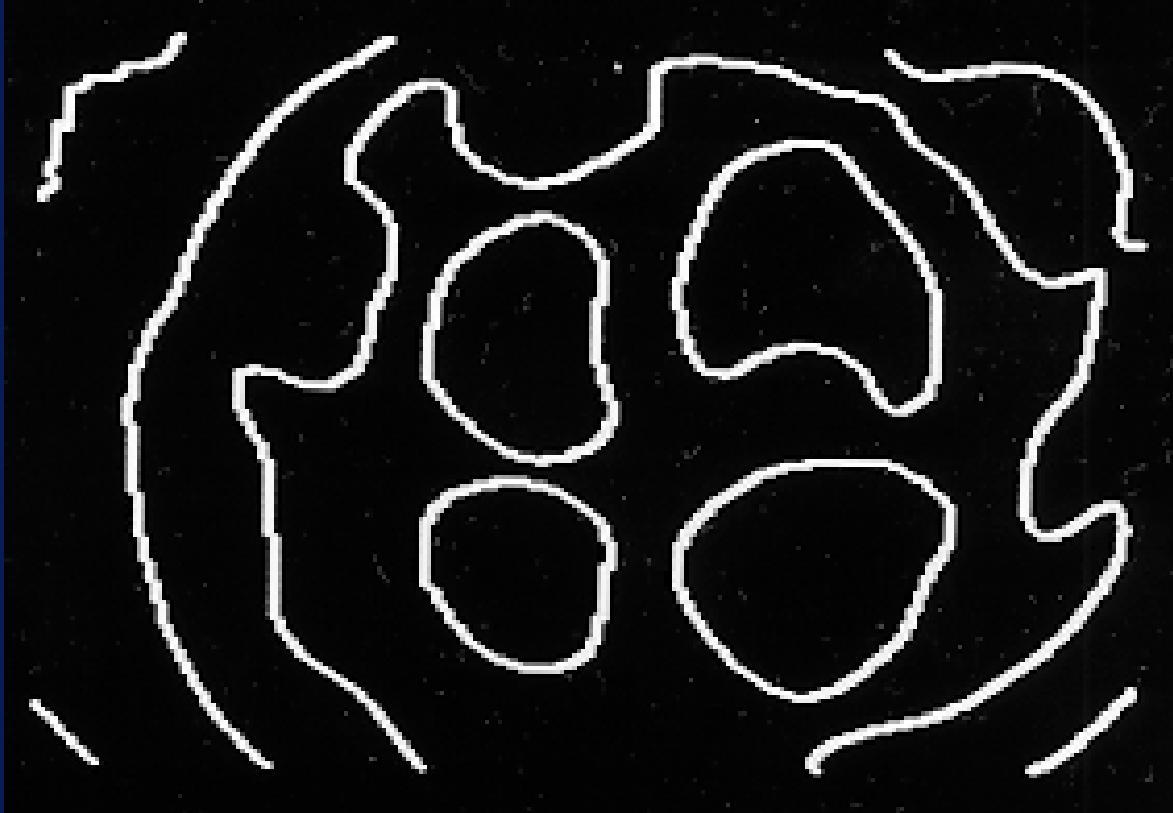
- *After Removal of Noisy Zero-Crossings*



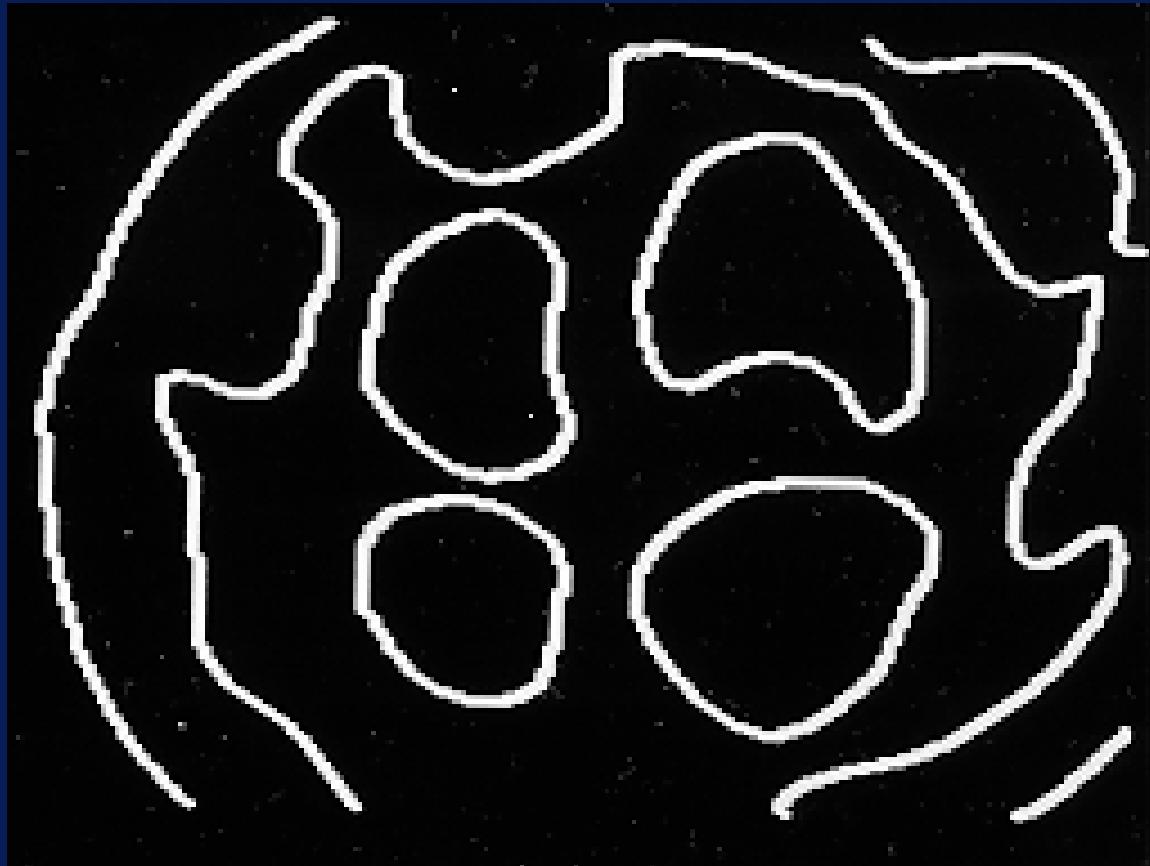
- *Zero-crossings derived by convolving the image with a Laplacian of Gaussian filter in which the standard deviation of the Gaussian is 6.0 pixels.*
  - **All Zero-Crossings**



- *After Removal of Noisy Zero-Crossings*



- *Zero-crossings derived by convolving the image with a Laplacian of Gaussian filter in which the standard deviation of the Gaussian is 9.0 pixels.*
  - **All Zero-Crossings**



- *After Removal of Noisy Zero-Crossings*

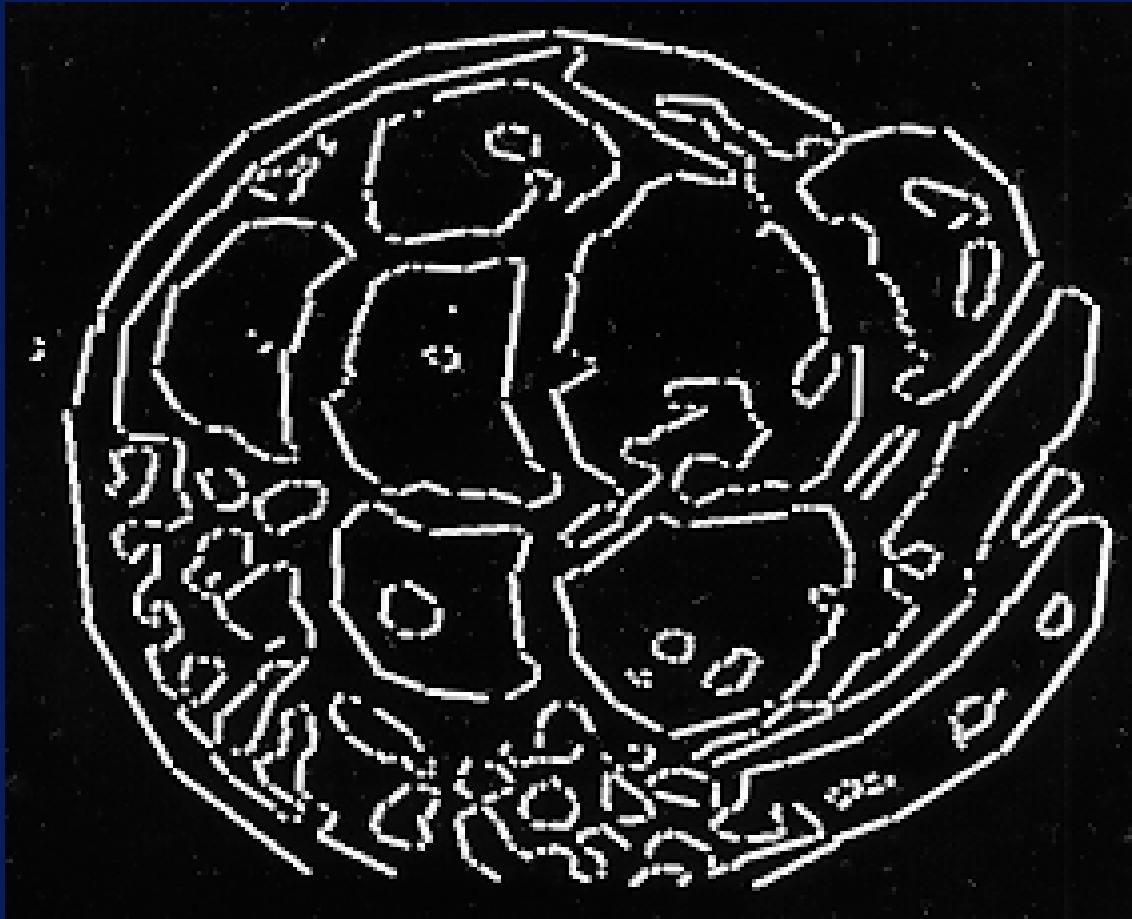
- By analysing the coincidence of zero-crossings in particular positions at different scales, one can infer evidence of physical reality, *i.e.* of a real physical edge.

- It is these spatially-coincident and hence (hopefully) real physically-meaningful zero-crossings that are then represented by the primitives of the raw primal sketch.

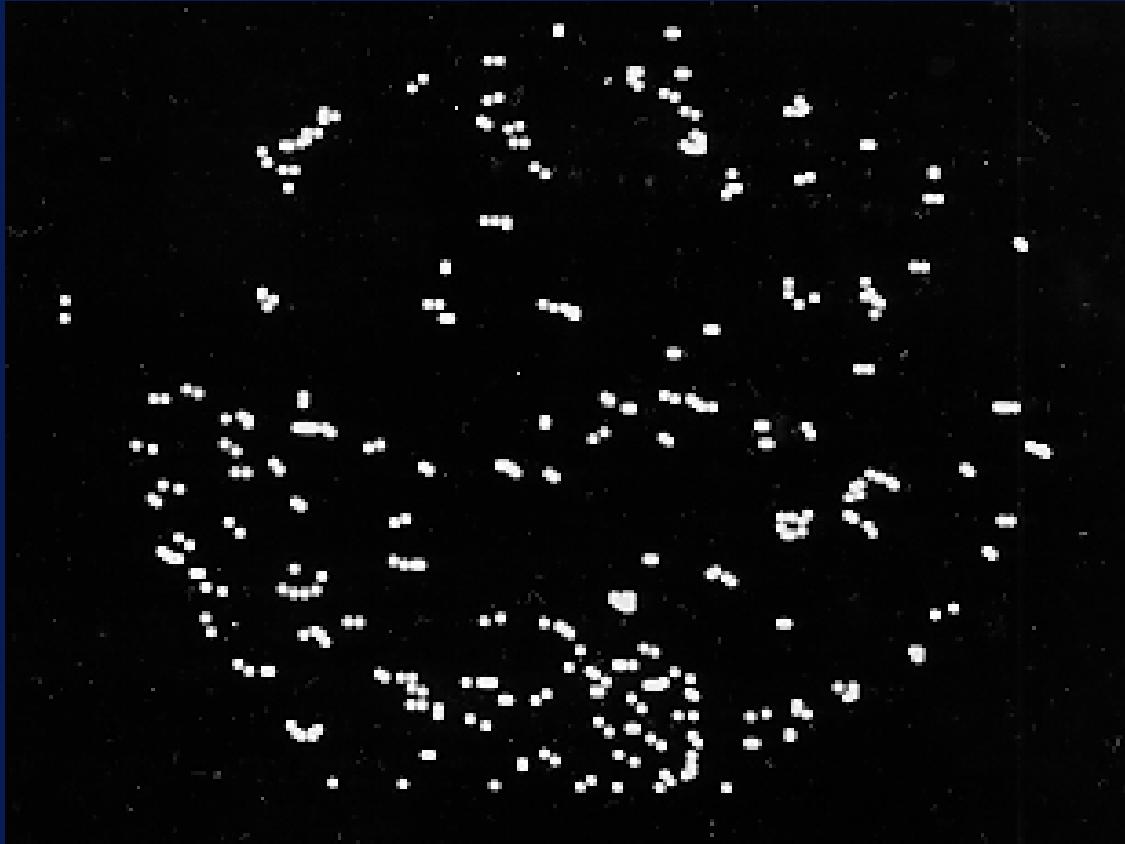
- *Edge* primitives are, effectively, local line segment approximations of the zero-crossing contours (see Figure 9.5)
- (*Slide 51*).
- Curves comprise a sequence of edges, delimited at either end by the *termination* primitives (see Figure 9.6) (*Slide 52*).

- Instances of local parallelism of these edges are represented by *bars* (see Figure 9.7) (*Slide 53*).

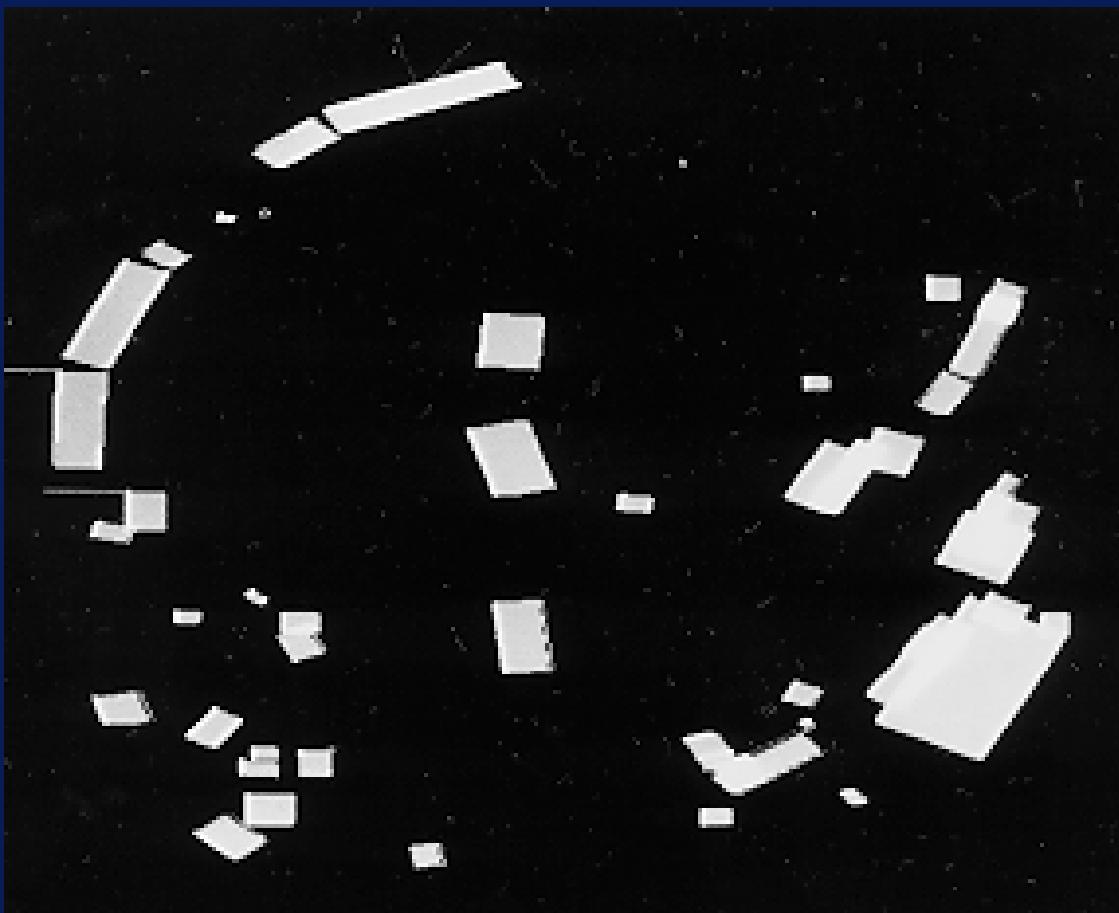
- Both represent the zero-crossing contours which are not present, *i.e.* which have no spatial coincidence, in all of the zero-crossing images derived at the different scales, *i.e.* using different standard deviations for the Gaussian Smoothing (see Figure 9.8) (*Slide 54*).



- *The Raw Primal Sketch: edges.*



- *The Raw Primal Sketch: terminations.*



- *The Raw Primal Sketch : Bars*



- *The Raw Primal Sketch : Blobs*

# The Full Primal Sketch

- The information made explicit in the raw primal sketch is still local and spatially restricted, *i.e.* it does not convey any global information about shape in an explicit manner.

- We now group these primitives so that the groups correspond to physically meaningful objects
  - in this sense, the grouping process is exactly what we mean by segmentation, which we described in Chapter 5.

- Since we are now dealing with more complex data, the segmentation processes must be more sophisticated.

- Many are based on Gestalt “figural grouping principles”, named after the Gestalt school of psychology formed in the early part of this century.

- For example, primitives can be grouped according to three criteria :
  - continuity
  - proximity
  - similarity

- In the first case, for instance, lines bounded by terminations which are co-linear would be grouped.
- Primitives which are spatially proximate are also candidates for the formation of groups while so too are primitives which belong to the same class, *i.e.*, which are similar.

- These criteria are not independent and the final grouping will be based on the relative weights attached to each criterion.

- The horizontal row would be grouped on the basis of spatial proximity and continuity;
  - the vertical columns would be grouped on the basis of similarity and, to an extent, continuity
  - the single diagonal group is formed on the basis of continuity of absence of tokens.

- This may seem a little strange: grouping entities which do not exist.
- However, it becomes a little more sensible when we note that, apart from the criteria of grouping, there are also definite grouping principles.

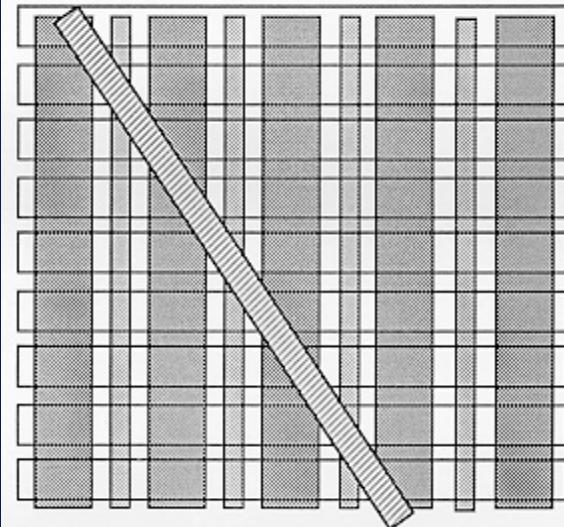
- First Principle - *Explicit Naming*

“grouped entities are to be treated as single entities and given an explicit name”
- *i.e.* grouped entities can be grouped, hence, grouping is a recursive operation.

- The groups we identified are not necessarily achieved in one pass of the grouping algorithm; typically, it would take two:
  - one to form the groups identified in Figure 9.10 (*Slide 68*) ...

- a second pass to group the new horizontal groups into the horizontal groups shown in Figure 9.9 (*Slide 67*)

+ . + + . + + . + + . + +  
+ + + + . + + . + + . + +  
+ + . + . + + . + + . + +  
+ + . + . + + . + + . + +  
+ + . + + + + . + + . + +  
+ + . + + . + . + + . + +  
+ + . + + . + . + + . + +  
+ + . + + . + + + + . + . + +



- *Grouping Processes*



- *Recursive Grouping*

- Now, however, we have new tokens comprising the termination points of these intermediate horizontal tokens
  - it is the grouping of these terminations, on the basis of continuity, that form the diagonal group.

- There is a second grouping principle, the principle of least commitment, which is pragmatic in nature.

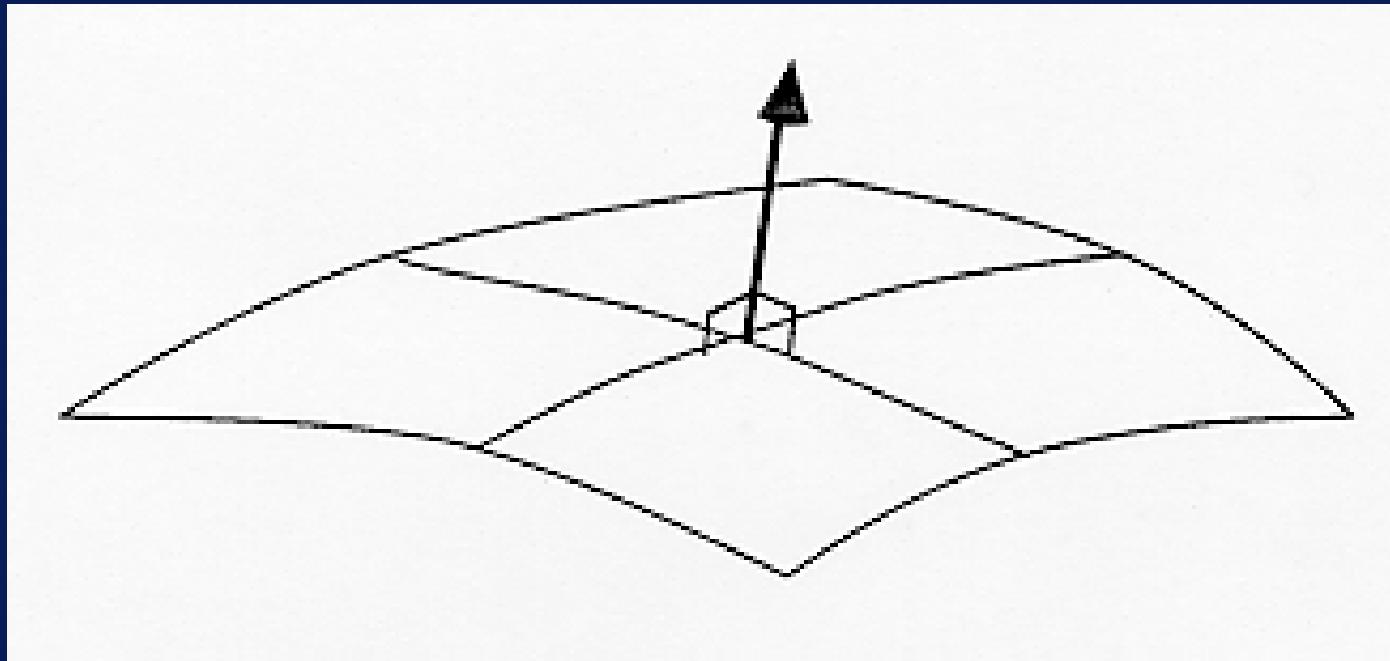
- It addresses the problem that, since grouping can be applied recursively to tokens, groups can be formed which do not successfully segment the physical objects and it may be necessary to backtrack and undo a grouping process.

- The principle of least commitment states that these “mistakes” are expensive and, therefore, grouping should be conservatively applied.

- In summary, the outcome of such grouping, the full primal sketch, makes explicit region boundaries, object contours, primitive shapes.
- It is a segmented representation and it exploits processes which are knowledge-free.

# The 2½D Sketch

- The next level of representation is the 2½D sketch (two-and-a-half dimensional) sketch.
- This is derived both from the primal sketch and from the grey-level image by using many visual cues, including stereopsis, apparent motion, shading, shape and texture.



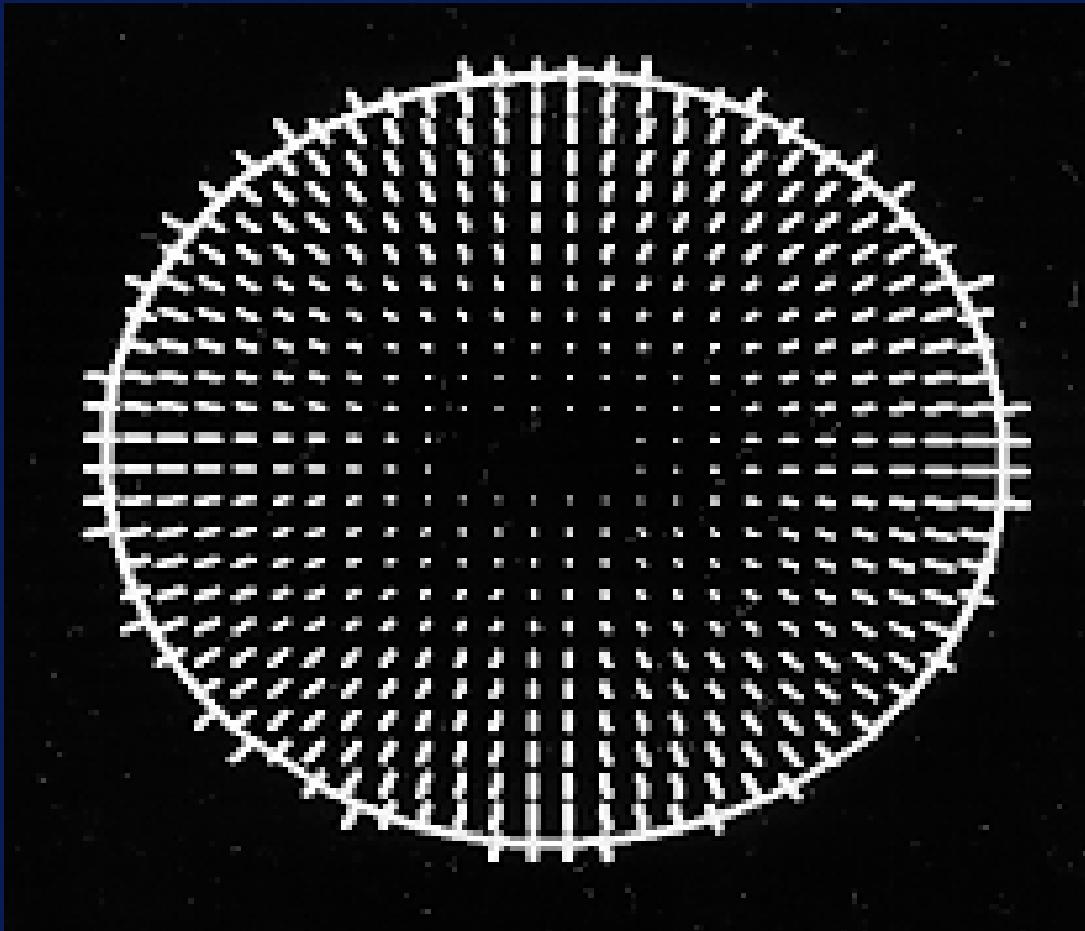
- *The Surface Normal Vector*

- The 2½D sketch is a rich view-centred representation of the scene containing :
  - primitives of spatial organisation and surface discontinuity and
  - the local surface orientation at each point and an estimate of the distance from the viewer.

- There are three degrees of freedom in the general specification of orientation of a vector.
- For example, we used the roll, pitch and yaw angles to specify the orientation of the robot end effector.

- In this case, however, we only require two angles since the vector is symmetric and the roll angle is redundant.

- Thus, the 2½D sketch can be thought of as a 2D array of 3-valued entities
  - representing the distance from the camera to the surface
  - the two angles specifying the surface normal vectors.



- *The 2½D Sketch*

- It is important to realise that this array-based, or *iconic*, representation is not the complete 2½D sketch; since it is also based on the full primal sketch, it integrates the information about grouping and segmentation.

- Thus, it is a *viewer-centred* 3D model, in that all distances are defined with respect to the camera co-ordinate system, with integrated surface-based object representations.

# The 3D Model

- The final stage of this information processing organisation of visual processes lies in the analysis of the 2½D sketch and the production of an explicit 3D representation.

- There are two issues which must be addressed :
  1. The conversion from a viewer-centred representation to an object-centred representation. This is, in effect, the transformation between a camera co-ordinate system and the real-world co-ordinate system and is exactly the process we discussed in Chapter 8 in the section on the Camera Model.

2. The type of 3D representation we choose to model our objects. There are three types of 3D representation based on volumetric, skeletal and surface primitives.

# Volumetric Representations

- Volumetric representations work on the basis of spatial occupancy, delineating the segments of a 3D workspace which are, or are not, occupied by an object.

- The simplest representation utilises the concept of a *Voxel* Image (voxel derives from the phrase volumetric element) which is a 3D extension of a conventional 2D binary image.

- It is typically a uniformly-sampled 3D array of cells
- each one belonging either to an object or to the free space surrounding the object.

- Because it is a three dimensional data-structure, voxel images require a significant amount of memory and, hence, they tend to be a coarse representation.

- For example, a  $1\text{m}^3$  work-space comprising  $1\text{cm}^3$  voxels will require approximately 1 Mbyte of memory (assuming that a voxel is represented by one byte; although eight voxels could be packed into a single byte, you do not then have simple direct access to each voxel.)

- The oct-tree is another volumetric representation
  - the volumetric primitives are not uniform in size;
  - you can represent the spatial occupancy of a work-space to arbitrary resolution in a fairly efficient manner.

- The oct-tree is a 3D extension of the quad-tree which we discussed in Chapter 5.

- The occupancy of a 3D image is achieved by
  - identifying large homogenous areas in the image
  - representing them as single nodes in a tree, the position in the tree governing the size of the region.

# Skeletal Representations

- The generalised cylinder, also referred to as the *generalised cone*, is among the most common skeletal 3D object representations.

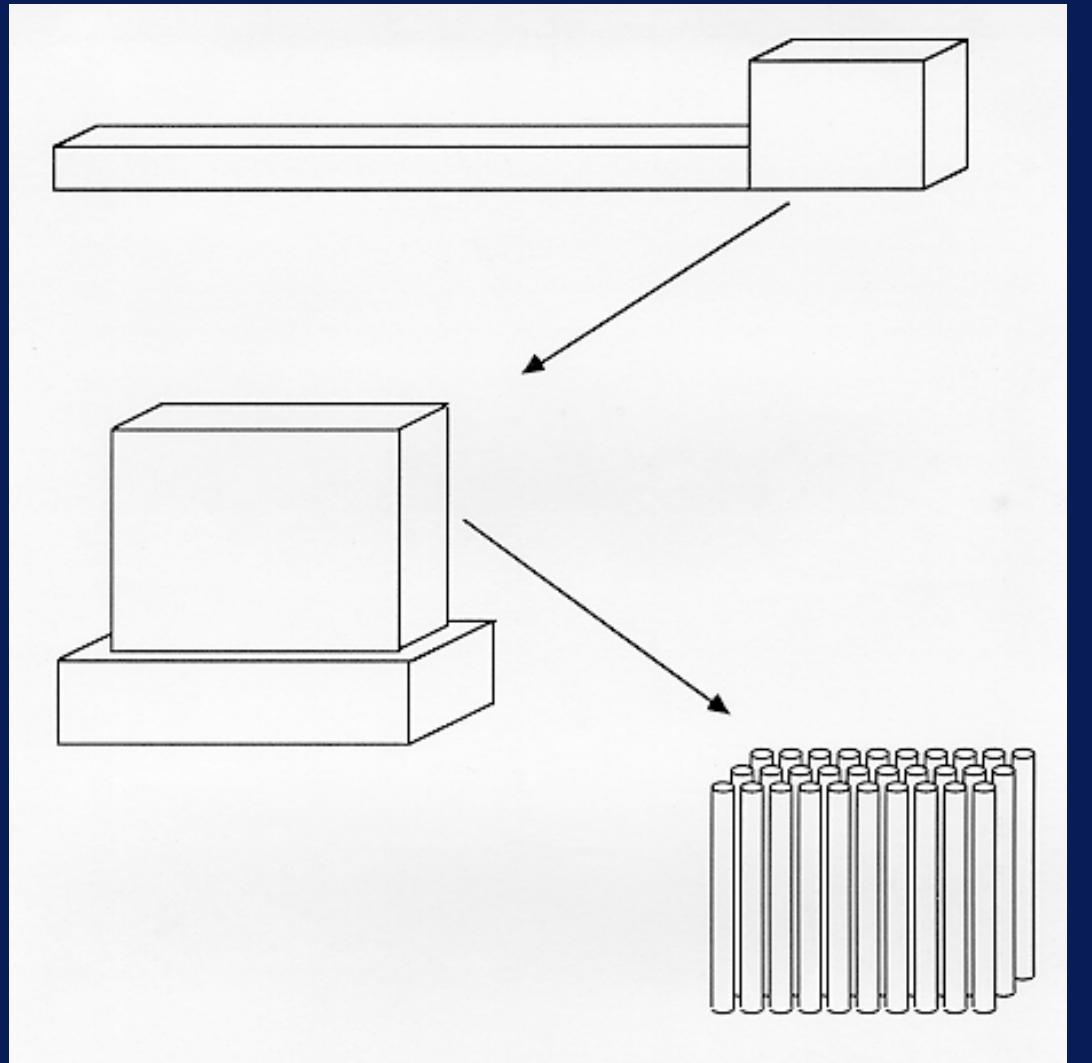
- the cross-section may vary in size, getting larger or smaller
- but the shape remains the same,
- the axis may trace out any arbitrary three-dimensional curvilinear path.

- Thus, a single generalised cylinder can represent
  - a cone (circular cross-section; linear decrease in diameter; linear axis)
  - a sphere (circular cross-section; sinusoidal variation in diameter; linear axis)
  - a washer (rectangular cross-section; constant area; circular axis).

- However, a general 3D model comprises several generalised cones and is organised in a modular manner with each component subsequently comprising its own generalised cylinder based model.

- Thus, the 3D model is a hierarchy of generalised cylinders : see Figure 9.13

- *Generalised Cylinder Representation*
- *of a Tooth-Brush*



# Surface Representations

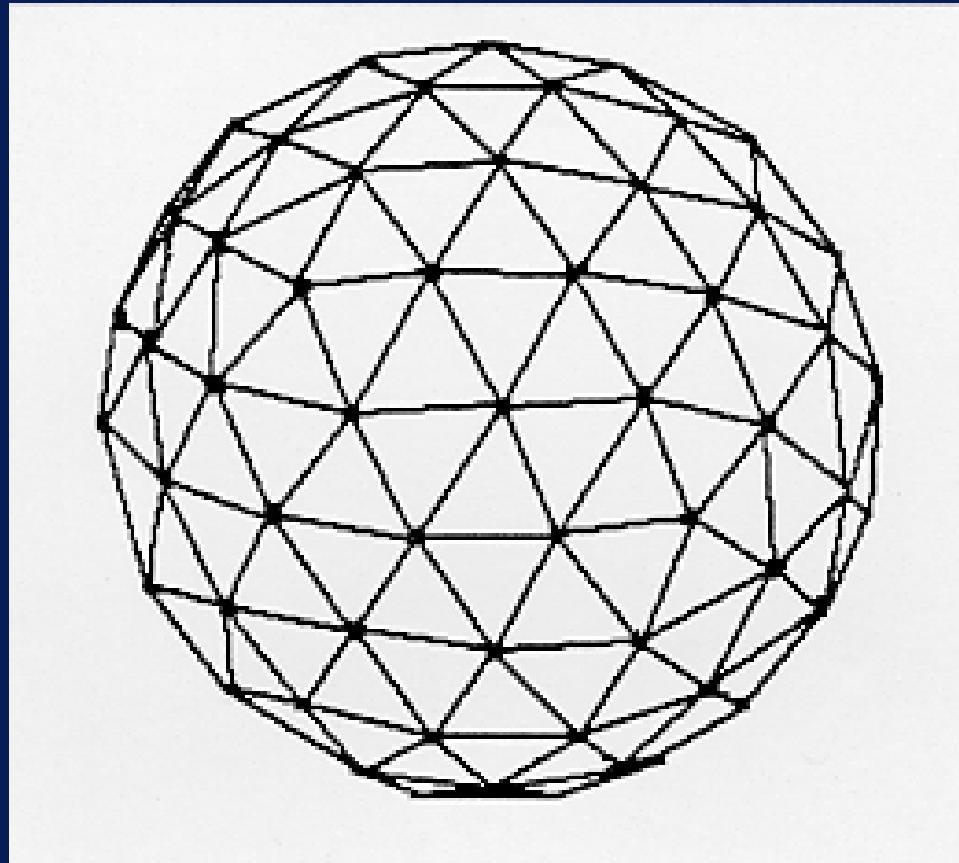
- We have a choice to make and to decide which type of surface primitives (or surface patches) we will allow:
  - planar patches
  - curved patches

- Although there is no universal agreement about which is the best, the planar patch approach is quite popular and yields polyhedral approximations of the object being modelled.

- This is quite an appropriate representation for man-made objects which tend predominantly to comprise planar surfaces.

- It is not, however, a panacea for 3D representational problems and it would appear that many of the subtleties of 3D shape description cannot be addressed with simplistic first-order planar representations.

- Nevertheless, it does have its uses and, even for naturally curved objects, it can provide quite a good approximation to the true shape, if an appropriate patch size is used.



- *A Sphere comprised of*
- *triangular planar patches*

# Recognition Rules

- The voxel image represents the 3D shape in an implicit manner (in the same way as a 2D binary image represents 2D shapes) and recognition can be effected by 3D template matching.

- However, all of the problems associated with this technique (see Chapter 6) still apply and, indeed, they are exacerbated by the increase in the dimensionality of the model.

- Object recognition using oct-trees essentially requires a graph (or tree) matching algorithm
  - since the oct-tree will vary with the orientation of the object, especially at the lower levels
  - the matching algorithm must be able to detect structural similarities in subsets of the tree.

- Similarly, a tree matching technique can be used to facilitate matching models based on generalised cylinders since ...

- the representation is a hierarchical structure of nodes comprising three parametric entities :
  - » the axis
  - » the cross section
  - » and the variation of the cross section.

- Finally, it is difficult to accomplish object recognition using planar surface models for anything except the simplest of regular objects.

- The size and position of each path and
- the total number of patches
- will vary considerably with the orientation of the original object and the completeness of the data.

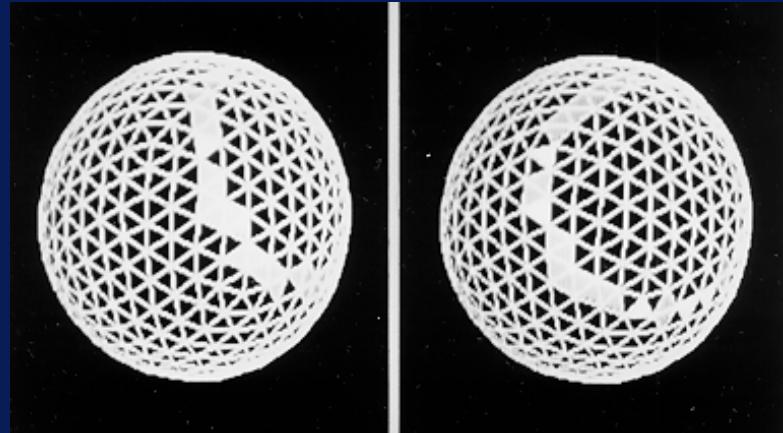
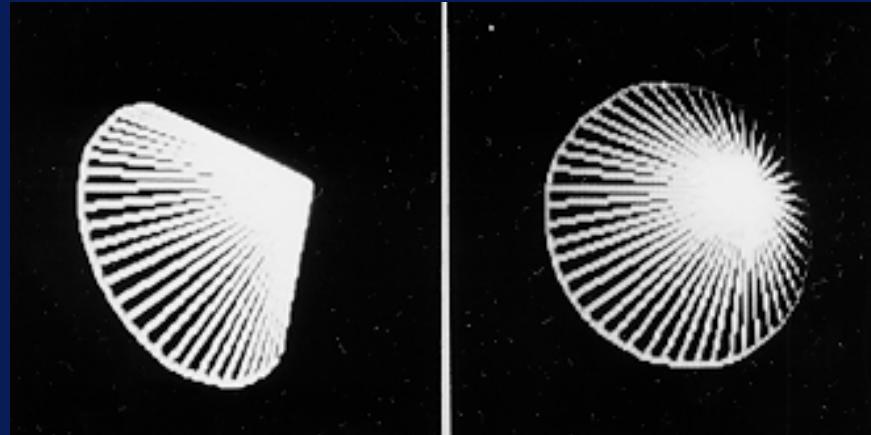
- Instead, object recognition can be effected by backtracking somewhat and regenerating an iconic (array-based) version of the 2½D sketch.
- The recognition process is accomplished by matching this 2D representation with a template.

- It assumes, however, that the template and object poses have first been registered;
  - otherwise, several (indeed a very large number) of object templates will have to be used in the template matching process
  - each one generated by projecting the 3D model onto the 2½D sketch at different orientations.

# The Extended Gaussian Image

- The 2½D sketch provides us with the local orientation at each point on the surface of the object.

- These surface normals can be represented on a unit sphere called a *Gauss Map*.
  - Attach a unit mass to each end point,
  - we observe a distribution of mass over the Gaussian Sphere;
  - this distribution is called the *Extended Gaussian Image (EGI)* of the object.



- *An artificial object and its*
- *Extended Gaussian Image (EGI)*

- The EGI is effectively a 2D histogram of surface normal orientations, distributed on a sphere.
- The attitude, or pose, of an object is greatly constrained by the global distribution of the EGI mass over the visible Gaussian hemisphere.

- Constraints on the viewer direction can be derived from the position of the EGI mass centre, and from the directions of the EGI axis of inertia.

- To determine object pose, we simply match partial EGIs, derived from the observed scene, with template EGIs derived from our stored prototypical model.

# Visual Processes

- *Stereopsis*
- The use of two views of a scene to recover information about the distance of objects in the scene from the observer (the Cameras).

- We have discussed in detail how the inverse perspective transformation allows us to construct a line describing all of the points in the 3D world which could have been projected onto a given image point.

- If we have two images acquired at different positions in the world, *i.e.* a stereo pair
  - then for the two image co-ordinates which correspond to a single point in 3D space, we can construct two lines
  - the intersection of which identifies the 3D position of the point in question.

- Thus, there are two aspects to stereo imaging
  - :
- 1. The identification of corresponding points in the two stereo images.
- 2. The computation of the 3D co-ordinates of the world point which gives rise to these two corresponding image points.

- The main problem in stereo is to find the corresponding points in the left and right images.

- In characterising a stereo system, we must address :
  - the kind of visual entities on which the stereo system works;
  - the mechanism by which the system matches visual entities in one image with corresponding entities on the other.

- Typically, the possible visual entities from which we can choose include
  - at an iconic level, zero-crossing points, patches (small areas) in the intensity image and patches in filtered images
  - at a more symbolic level, line segment tokens, such as are made explicit in the raw primal sketch.

- The matching mechanism which establishes the correspondence will depend on the type of visual entities which we have chosen :

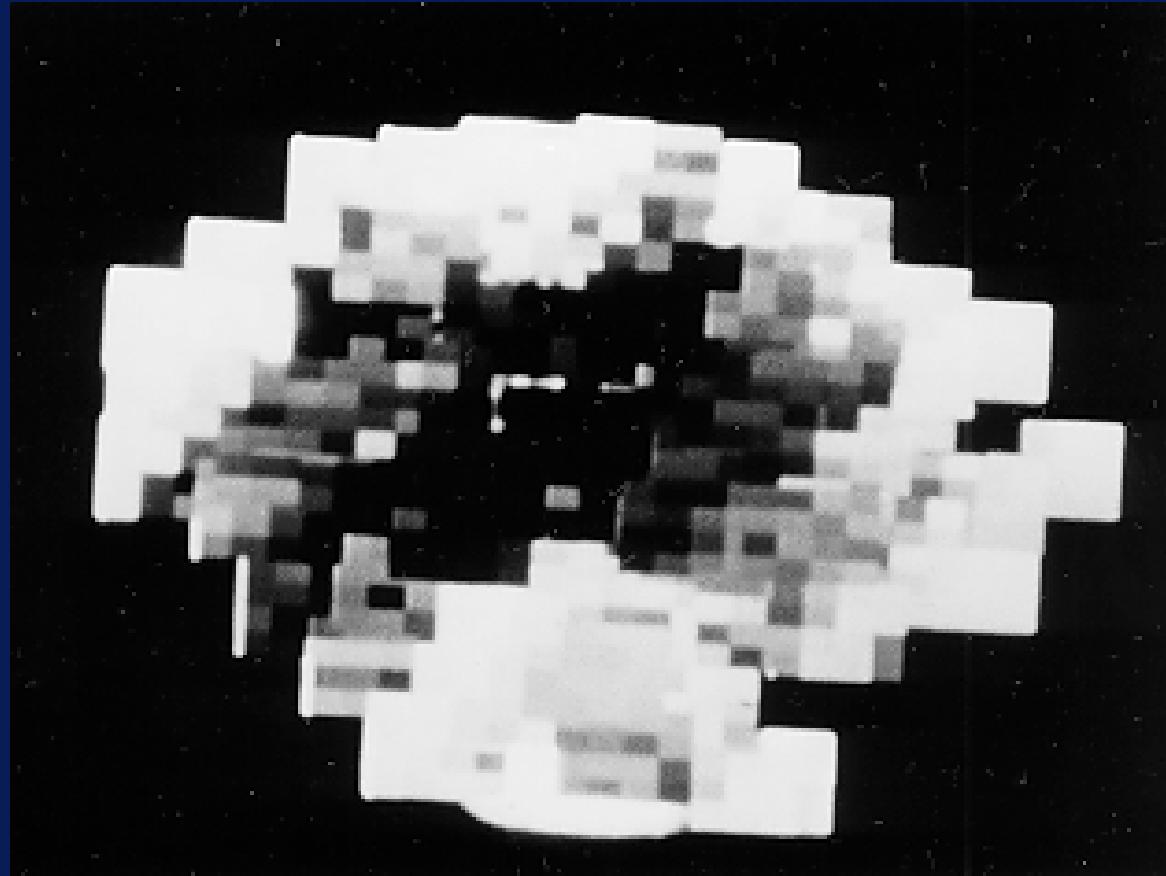
- iconic entities will normally exploit some template matching paradigm, such as normalised cross-correlation  
(see Chapter 6)
- token entities can be used with more heuristic search strategies.



• *Right Stereo Image*



• *Left Stereo Image*



- *Disparity of a Stereo Pair of Images*

\* This disparity representation, which is based on the zero-crossings of Laplacian of Gaussian filtered images, is merely a short-hand way of summarising the relative position of corresponding points in the two images. Given a zero-crossing point, we can compute its corresponding position in the second stereo image from its disparity, represented here as intensity. This assumes that we know the direction in which the corresponding point lies; for a stereo pair of cameras which have parallel focal axes, the direction is horizontal.

- As an example, the stereo disparity\* image shown in Figure 9.17 (*Slide 12*), in which disparity is inversely proportional to grey-level, is derived ...

- by convolving the stereo pair (see Figure 9.16) (*Slides 11 & 12*) with a Laplacian of Gaussian filter
- computing the zero-crossings
- correlating patches in the convolved image, centred on the zero-crossing points at discrete intervals along the zero-crossing contour.

# Camera Motion

- The analysis of object motion in sequences of digital images, or of apparent motion in the case of a moving observer, to provide us with information about the structure of the imaged scene is an extremely topical and important aspect of current image understanding research.

- The general object motion problem is difficult, since the motion we perceive can be due to either the rotation of the object, a translation of the object, or both.

- We confine our attention to the somewhat easier problem of camera motion
  - and to the study of apparent motion of objects in a scene arising from the changing vantage point of a moving camera.

- This restriction is not necessarily a limitation on the usefulness of the technique;

- the concept of a camera mounted on the end-effector of a robot, providing hand-eye co-ordination
- or on a moving autonomously guided vehicle (AGV), providing navigation information,
- is both appealing and plausible.

- From an intuitive point of view, camera motion is identical to the stereo process
  - in that we are identifying points in the image (*e.g.* characteristic features on an object)
  - then tracking them as they appear to move due to the changing position (and, perhaps, attitude) of the camera system.

- At the end of the sequence of images, we then have two sets of corresponding points, connected by *optic flow vectors*, in the first and last images of the sequence.

- Typically, we will have a sequence of vectors which track the trajectory of the point throughout the sequence of images.
- The depth, or distance, of the point in the world can then be computed in the manner discussed in Chapter 8 and in the previous section.

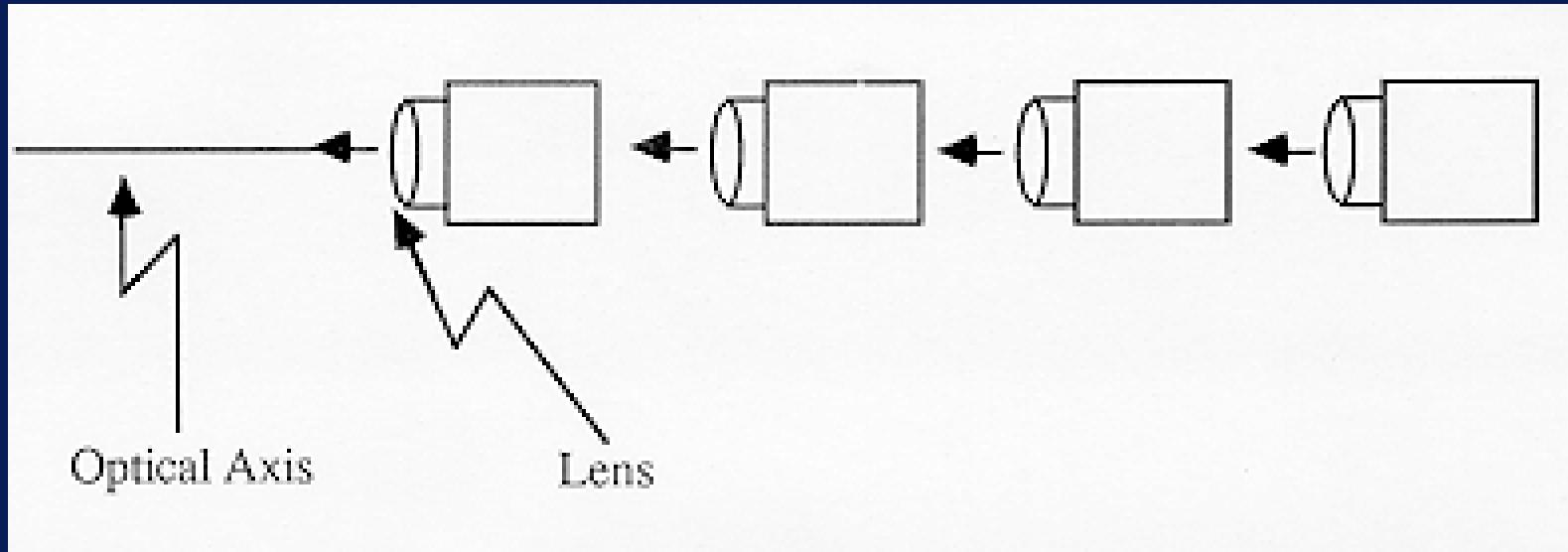
- However, there are a number of differences.

- *First :*
  - the tracking is achieved quite often, not by a correlation technique or by a token matching technique, but by differentiating the image sequence with respect to time to see how it changes from one image to the next.

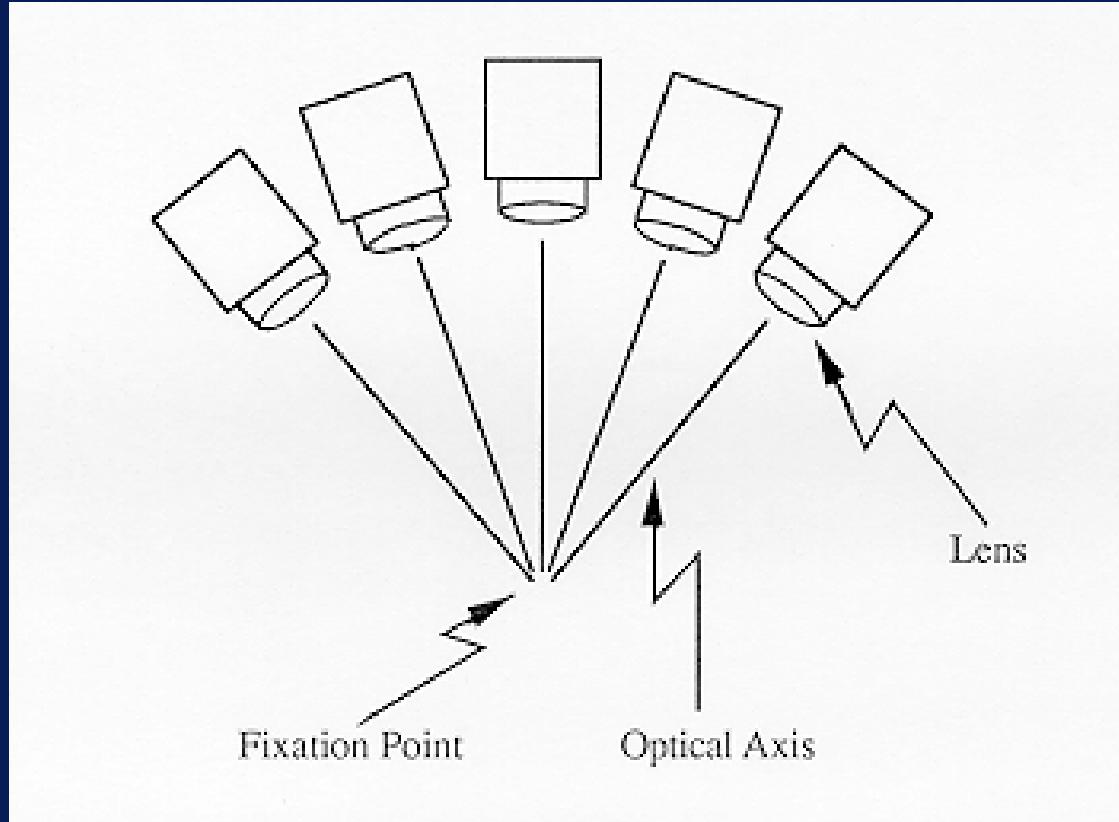
- there is often a subsequent matching process to ensure the accuracy of the computed image change, and sometimes it is not the grey-scale image which is differentiated but, rather, a filtered version of it.
- Nevertheless, the information about change is derived from a derivative of the image sequence.

- *Second :*
  - the “correspondence” between points is established incrementally, from image to image, over an extended sequence of images.

- We will confine our attention to two simple types of camera motion in order to illustrate the approach.
  - The first describes a trajectory along the optical axis of the camera (see Figure 9.18) (*Slide 28*)
  - In the second the camera is rotated about a fixation point.



- *Translational motion along the optic axis*



- *Rotational motion about a fixation point*

- The optic flow field resulting from this first type of ego-centric motion is very easy to compute
  - all flow vectors are directed radially outward from the focus of expansion (FOE)\*, *i.e.* the centre of the image.

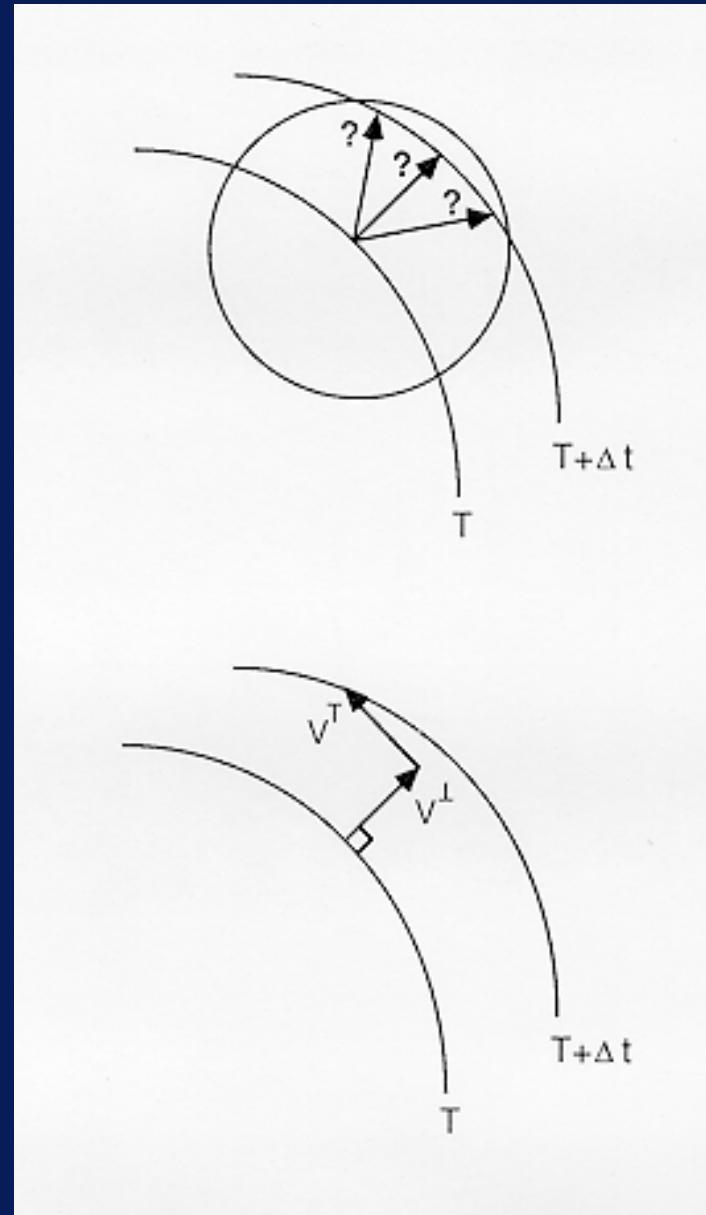
- For camera motion in rotation about a fixation point
  - the rotational component of optical flow can be determined directly from the known camera trajectory
  - the direction of the translational component is also constrained by the camera motion.

- knowing the direction of the flow vector, the magnitude of the visual motion is directly derived from a time-derivative of a sequence of images acquired at successive points along the camera trajectory.

- There is one main difficulty when attempting to compute the true optical flow, *i.e.* the visual motion, of a point in the image.

- It is generally referred to as the *aperture problem*.
  - Consider a contour in an image.
  - Let us say that we only have a small local window of visibility around the point of interest on the contour (see Figure 9.20) (*Slide 35*).

- *The Aperture Problem*



- If the position of the contour changes due to the camera motion, then we cannot say with any certainty in which direction it moved, based on the local information available in this small local window.

- The only thing we can compute with certainty is the *orthogonal component of the velocity vector*, i.e. the component which is normal to the local contour orientation.

- This vector component is referred to as  $v^\perp$ , and the second component is referred to as the tangential component  $v^T$ . Thus, the true velocity vector  $v$  is given by :
  - $v = v^\perp + v^T$

- If the luminance intensity does not change with time (*i.e.* there are no moving light sources in the environment) the component of the orthogonal velocity vector for each image point is given by :

$$v^\perp = \frac{-\partial I / \partial t}{|\nabla I|}$$

- where  $\partial$  indicates the partial derivative operator and  $|\nabla I|$  is the local intensity gradient.

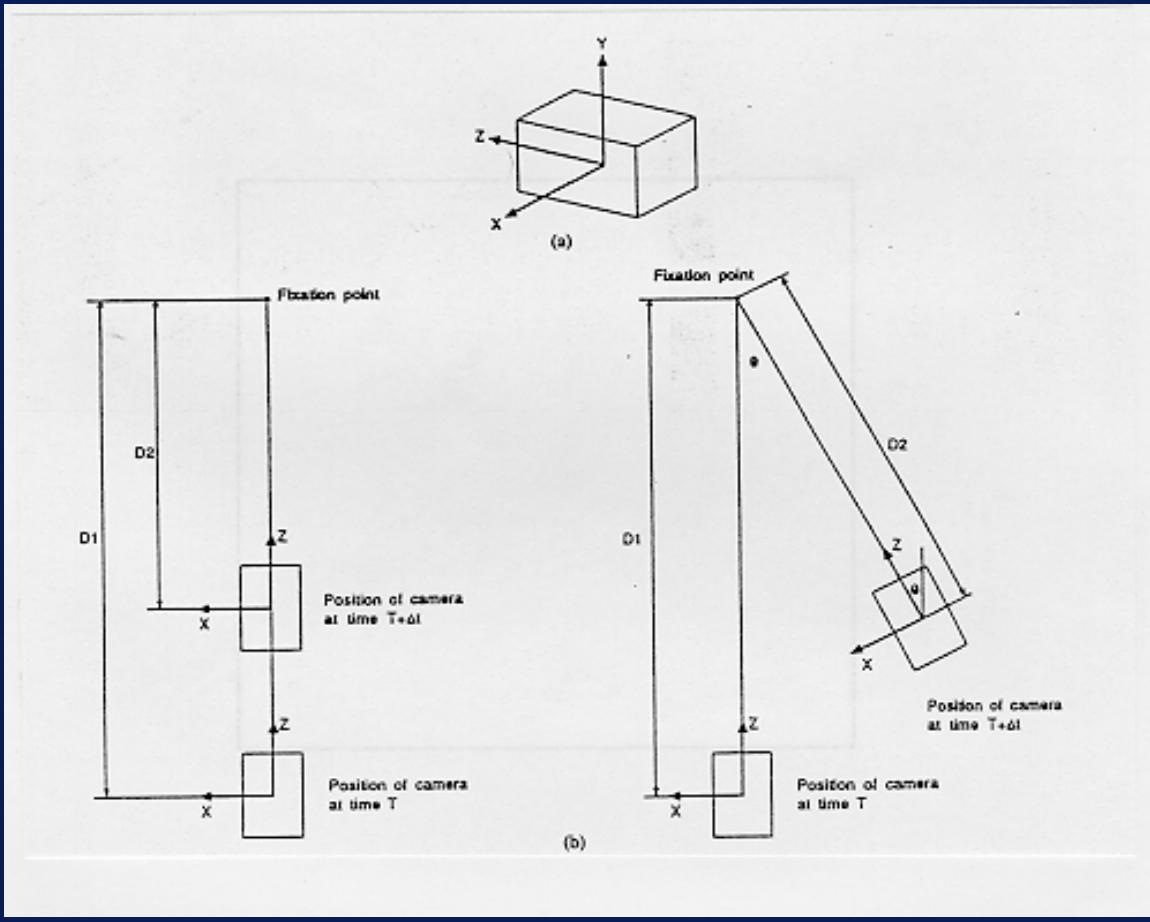
- In the technique described here :
  - we compute the time derivative of a  $\nabla^2 G$  filtered image instead of the raw intensity image
  - compute the optical flow at zero-crossing contours.

- This means that the amount of data to be processed is limited and, furthermore, the effects of noise are less pronounced.

- The computation of  $v^\perp$  (the orthogonal component of velocity) is based on a computation of the time derivative using image subtraction, in the same way as we saw that gradient-based edge detectors can be effected by local pixel differences, according to the relationship described above.

- The computation of the true velocity vector depends on the prior knowledge of the parameters of the camera motion :

- the position of the camera at time  $T$  and  $T+\Delta t$  from the fixation point;
- $\theta$ , the rotational angle of the camera around the  $Y$  axis
- $W_x$  and  $W_z$ , the components of the translational velocity of the camera along the  $X$  axis and the  $Z$  axis respectively.



- *Camera co-ordinate system and the parameters associated with camera motion*

- $W_x$  and  $W_z$  are defined with respect to the co-ordinate system of the camera at time  $T$  (see Figure 9.21) (*Slide 46*). Using basic trigonometric relations, we find :

$$W_x = \frac{D_2 \sin \theta}{\Delta t}$$

$$W_y = 0$$

$$W_z = \frac{D_1 - D_2 \cos \theta}{\Delta t}$$

- where  $D_1$  and  $D_2$  are the distances of the camera from the fixation point at time  $T$  and  $T + \Delta t$ , respectively.

- For the constrained camera motion shown in Figure 9.19 (*Slide 29*), the camera translational velocity is given by equations for  $W_x$ ,  $W_y$  and  $W_z$  (*Slide 47*) while the rotational velocity  $\omega$  is
- $(0, \theta/\Delta t, 0)$

- The image velocity components can then be written as :

$$v_t = \left( \frac{x(D_1 - D_2 \cos \theta) - FD_2 \sin \theta}{Z\Delta t}, \frac{y(D_1 - D_2 \cos \theta)}{Z\Delta t} \right)$$

$$v_r = \left( \frac{-x^2 - F^2 \theta}{F\Delta t}, \frac{-xy\theta}{F\Delta t} \right)$$

- In these two equations for  $v_t$  and  $v_r$ , the only unknown is  $Z$  (which is what we wish to determine).
- Thus, to determine  $v_t$  and  $v_r$ , and hence  $Z$ , we exploit the value of  $v^\perp$ , the orthogonal component of velocity, computed at an earlier stage.

- In the solution by geometrical construction,  $v$  is determined from the intersection of three straight lines derived from  $v_r$  (for which all terms are known),  $v^\perp$  (which was computed previously), and the position of the FOE.

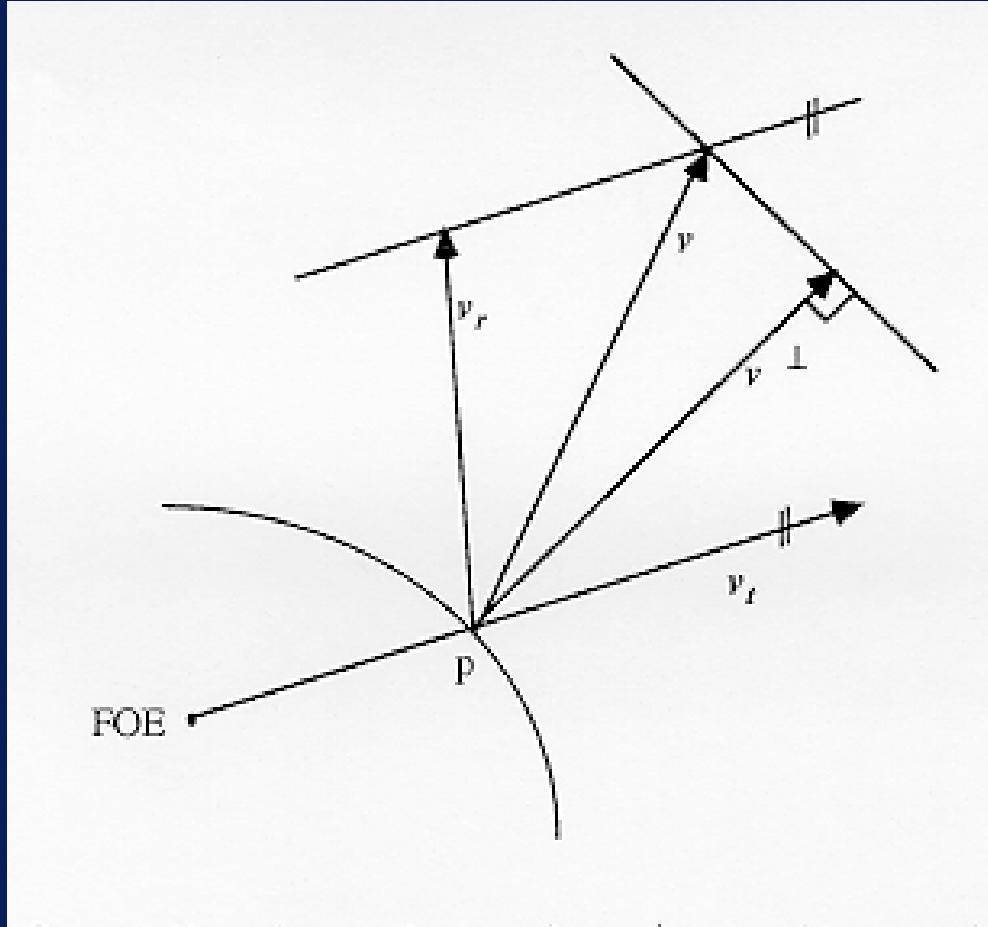
- First,  $v_r$  defines the first line of the construction.
- Second, the position of the FOE defines the direction of  $v_t$ , since  $v_t$  is parallel to the line joining the FOE and the point  $(x,y)$  in question.

- Thus, the second line is parallel to  $v_t$  and passes through the point given by  $v_r$ .

- The co-ordinates of the FOE are given by :

$$(x_{FOE}, y_{FOE}) = \left( \frac{FW_x}{W_z}, \frac{FW_y}{W_z} \right)$$

- where  $W_x$ ,  $W_y$  and  $W_z$  are the known velocities of the camera in the  $x$ ,  $y$  and  $z$  directions, respectively.



- *Computation of true velocity  $v$  from  $v^\perp$ ,  $v_t$  and  $v_r$  at a point  $P$  on a zero-crossing contour.*

- Finally, we note again that  $v$  is also given by the sum of the orthogonal component and the tangential component of velocity :

- This can be accomplished directly by solving the attendant system of equations or by a geometrical construction.
  - $v = v^\perp + v^T$

- Since these two vectors are orthogonal to one another, and since  $v^\perp$  is known, this relationship defines a third line through the point given by  $v^\perp$  and normal to the direction of  $v^\perp$ .

- Hence,  $v$  is given by the intersection of the second and the third lines: see Figure 9.22 (*Slide 56*).

- In the simpler case of translatory motion along the optic axis,  $\theta$  is equal to zero and the translational component of velocity reduces to :

$$v_t = \left( \frac{x(D_1 - D_2)}{Z}, \frac{y(D_1 - D_2)}{Z} \right)$$

- ... while the rotational component  $v_r$  is now zero.
- Computing  $v$  in this manner and, in particular  $v^\perp$  using image differences, errors can still be recorded in the final flow.

- A significant improvement can be achieved by performing a contour-to-contour matching between successive frames, along the direction of the flow vectors, tuning the length of the flow vectors to the correct size.

- The tracking procedure
  - searches in the direction of the flow vector until the next contour is found
  - then it searches in the direction of the new flow vector
  - and so forth until the whole image sequence is processed.

- The algorithm for computing the optical flow can be summarised as follows:

*Convolve the images with a Laplacian of Gaussian operator*

*Extract the zero-crossings*

*Compute the difference between the  $\nabla^2G$  of successive frames of the sequence*

*Compute the velocity component in the direction perpendicular to the orientation of the contour*

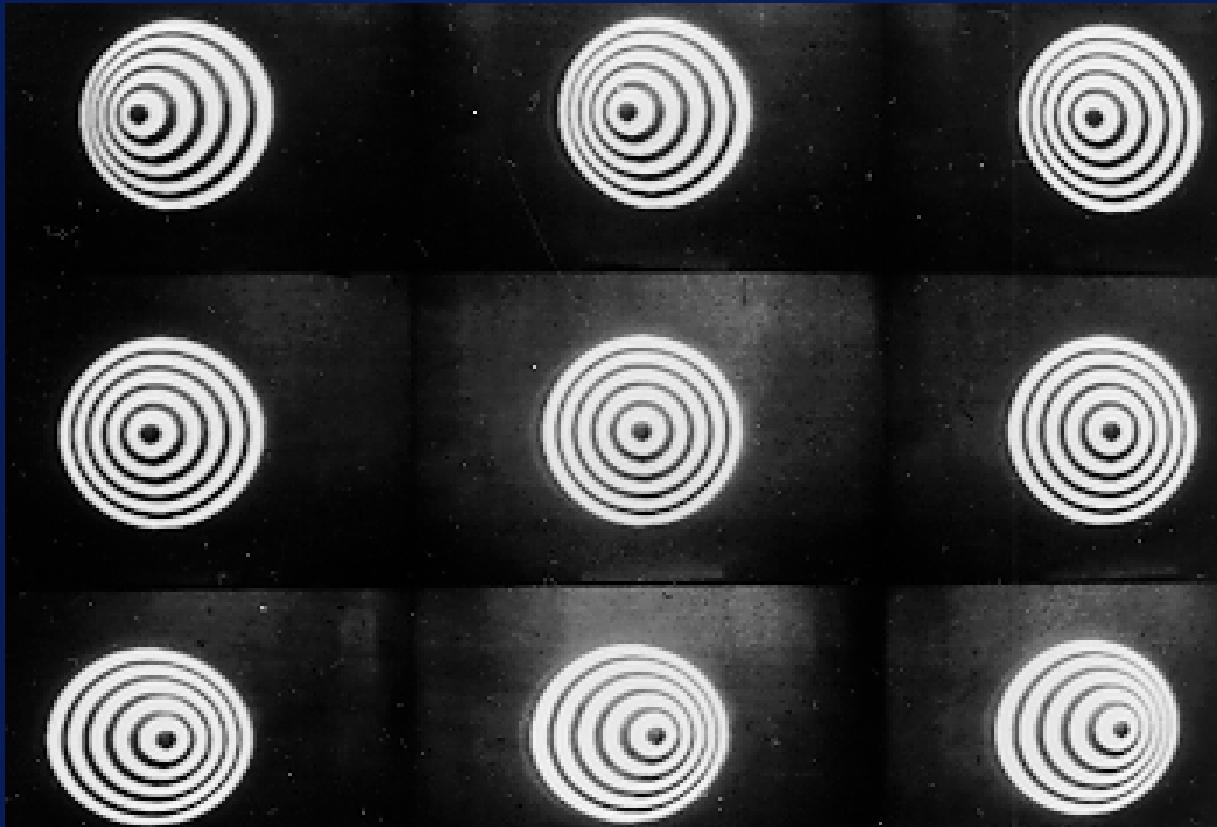
*Compute the velocity along the contour using the known motion parameters*

*Search for the zero-crossings of the second frame projected from the first frame in the direction of the velocity vector.*

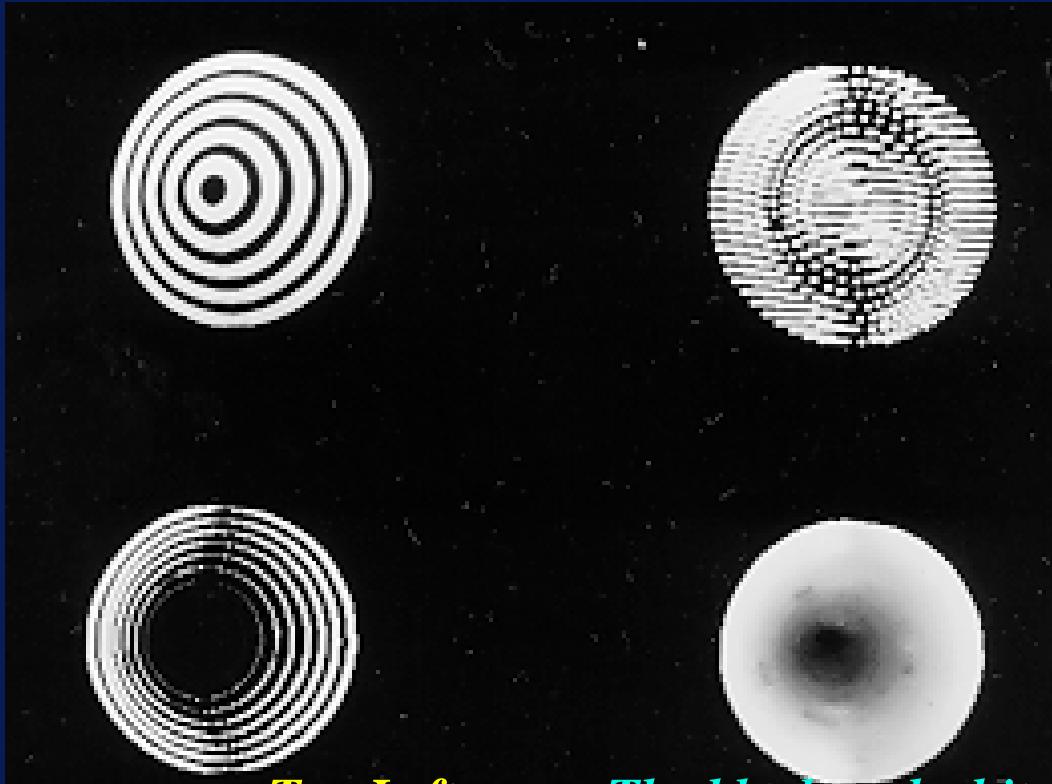
- The depth, for each contour point, is computed as before by applying the inverse perspective transformation, derived from camera models corresponding to the initial and final camera positions, to the two points given by the origin of the optical flow vector and the end of the optical flow vector.



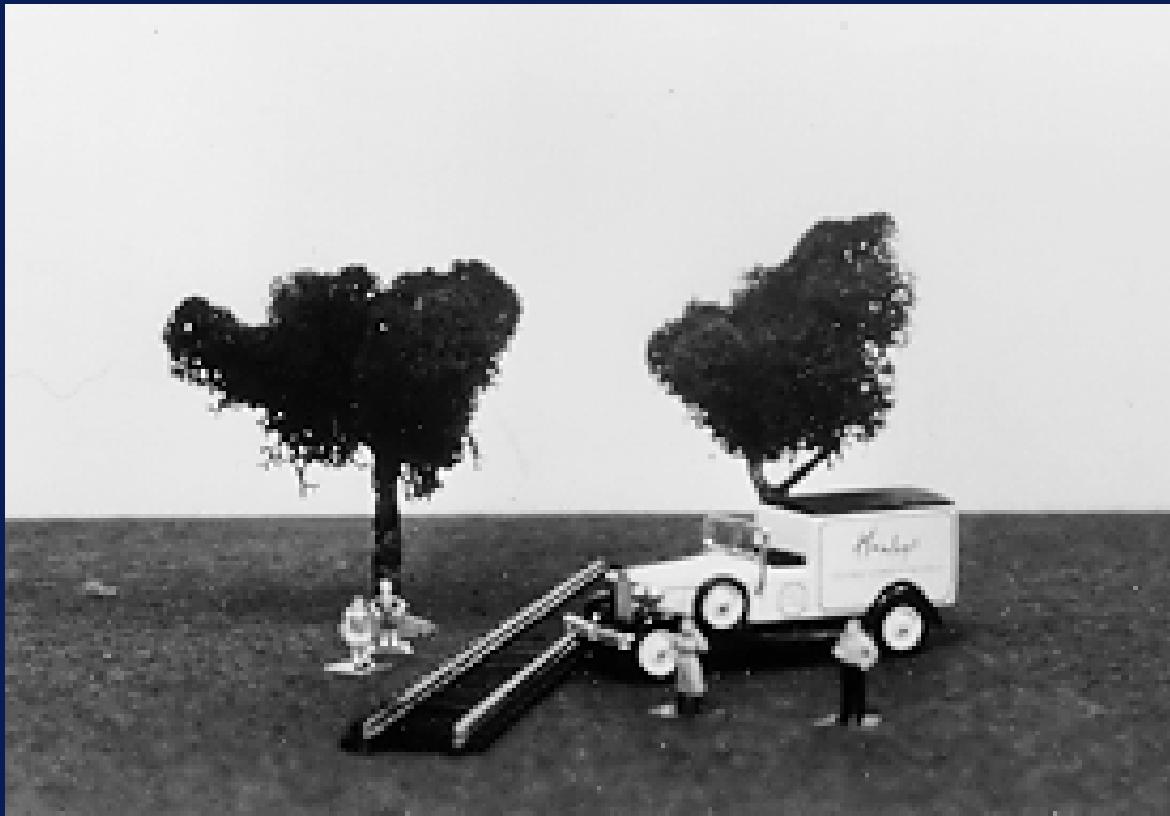
- *A black and white cone*



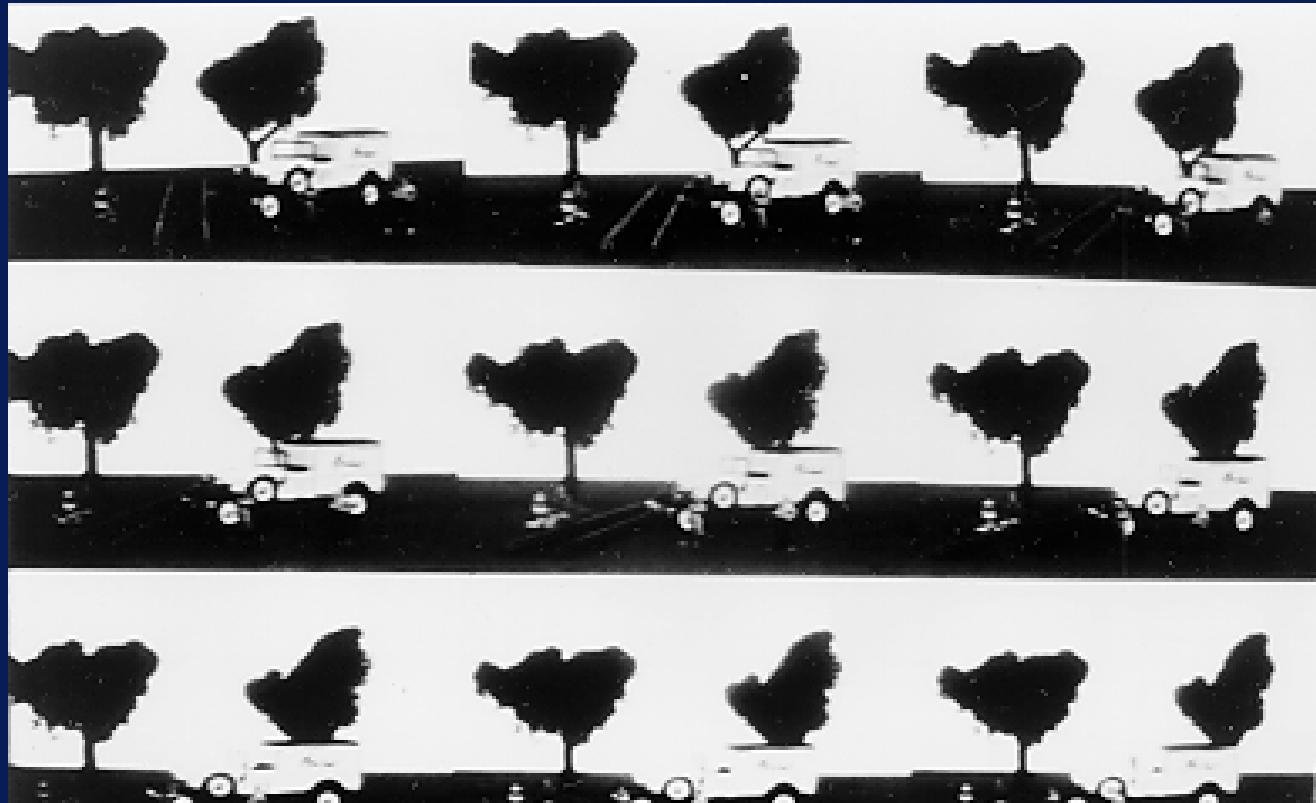
- *The nine views of the black and white cone*



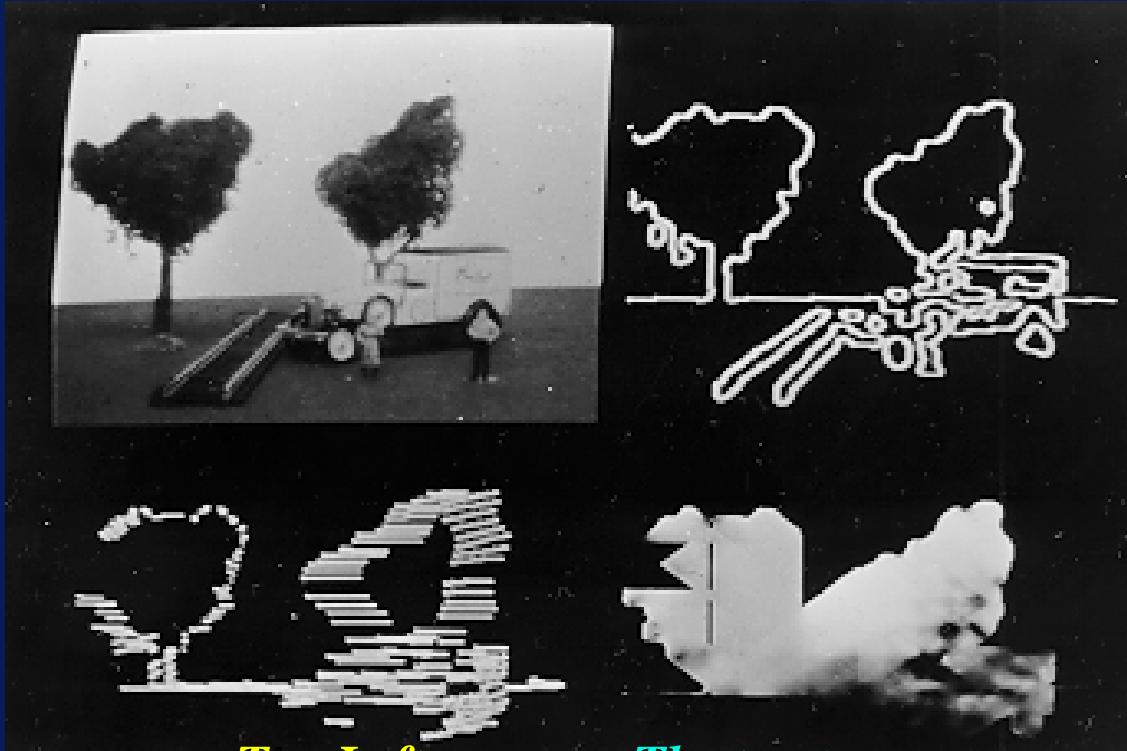
- **Top Left** : *The black and white cone*
- **Top Right** : *The optical flow vectors*
- **Bottom Left** : *Zero crossings with intensity proportional to distance from camera*
- **Bottom Right** : *Range image with intensity proportional to distance from camera*



• *A Toy-Town Scene*



- *The nine views of the toy-town scene*



- **Top Left** : *The toy-town scene*
- **Top Right** : *The optical flow vectors*
- **Bottom Left** : *Zero-crossings with intensity proportional to distance from camera*
- **Bottom Right** : *Range image with intensity proportional to distance from camera*

# Shading

- The construction of the 2½D sketch requires one further element: the computation of the local orientation of a point, *i.e.* the surface normal vector.

- The analysis of the shading of a surface, based on assumed models of the reflectivity of the surface material, is somewhat used to compute this information.

- The amount of light reflected from an object depends on (referring to figure 9.29) (*Slide 79*) :
  1. The surface material
  2. The emergent angle,  $e$ , between the surface normal and the viewer angle.
  3. The incident angle,  $i$ , between the surface normal and light source direction.

- There are several models of surface reflectance, the simplest of which is the Lambertian model.
- A Lambertian surface is a surface that looks equally bright from all view points, *i.e.* the brightness of a particular point does not change as the viewpoint changes.

- It is a perfect diffuser :
  - the observed brightness depends only on the direction to the light source, *i.e.* the incident angle  $i$ .

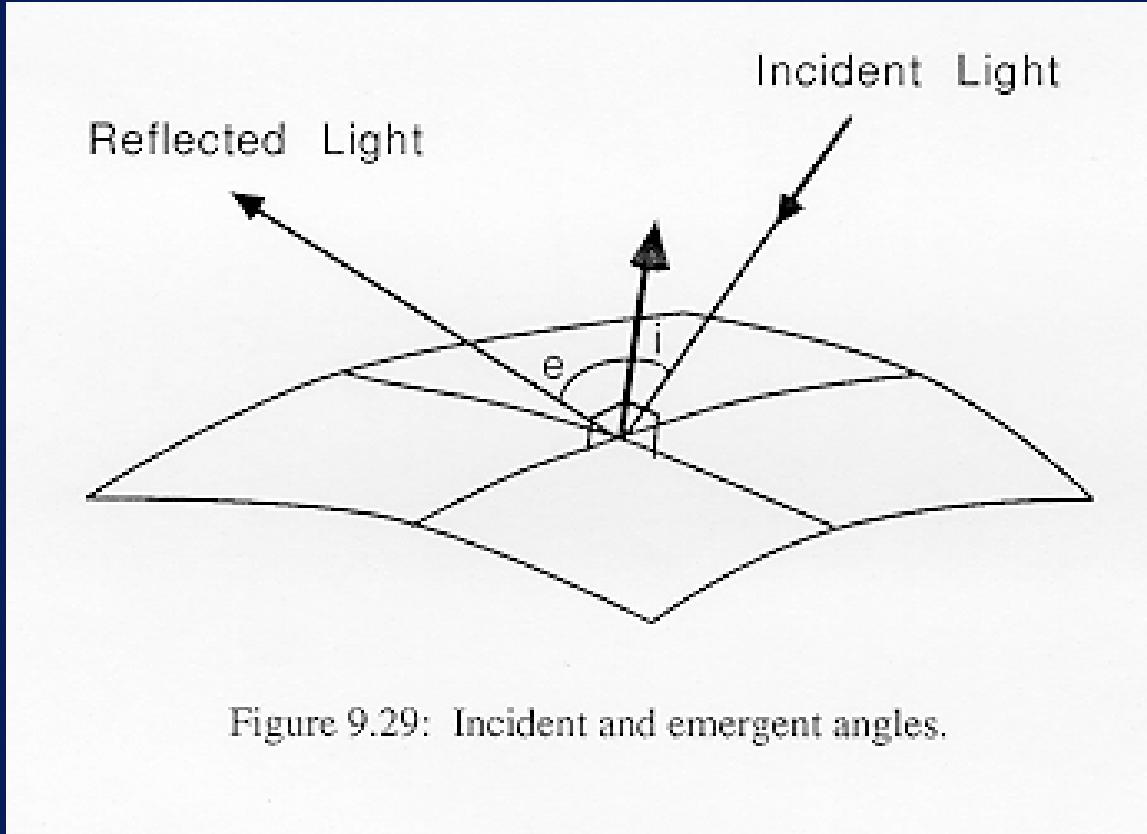


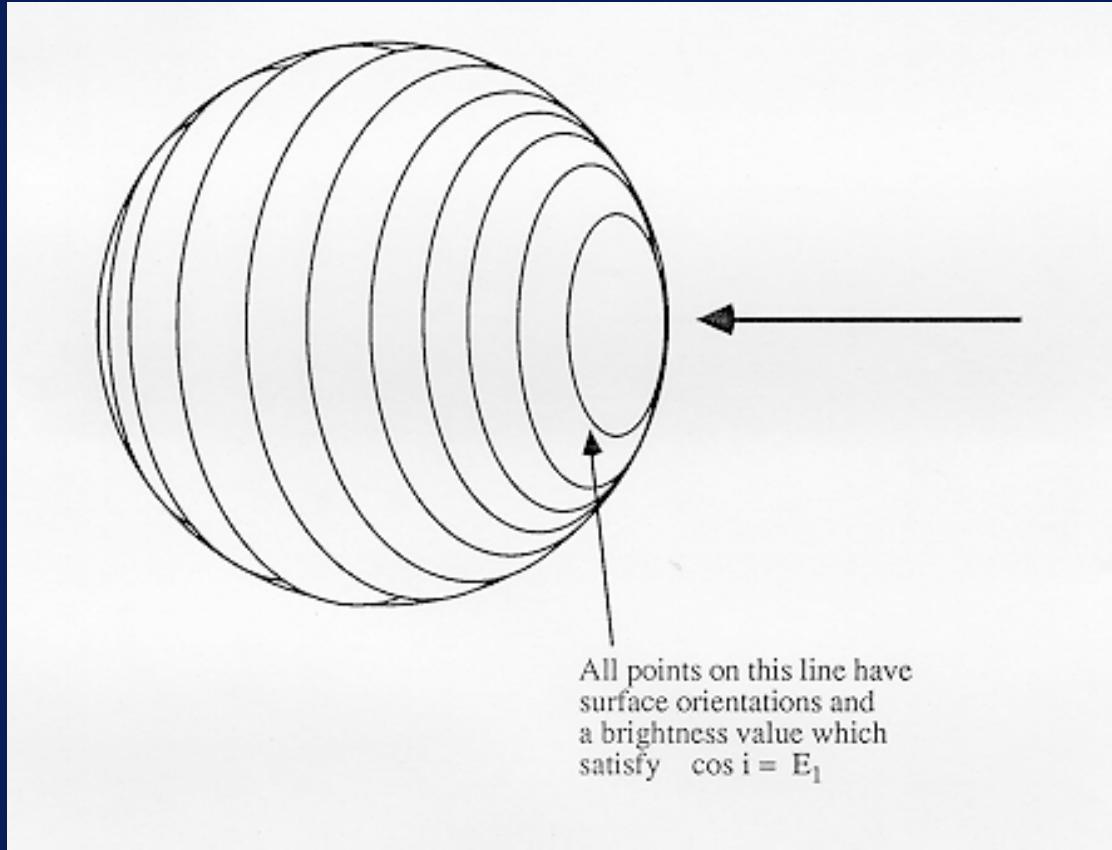
Figure 9.29: Incident and emergent angles.

- *Incident and Emergent Angles*

- Let  $E$  be the observed brightness, then for a Lambertian surface :
  - $E = \rho \cos I$
- where  $\rho$  is a constant called the surface albedo and is peculiar to the surface material under analysis.

- The reflectance properties of Lambertian surfaces are encapsulated by images of Lambertian spheres of radius 1 upon which we draw (for convenience) lines joining points of equal brightness, *i.e.* iso-brightness lines.

- There will, in general, be many points on the surface of the sphere which satisfy
- $E = \rho \cos I$  for a given brightness  $E_l$ ; See Figure 9.30 (*Slide 83*).



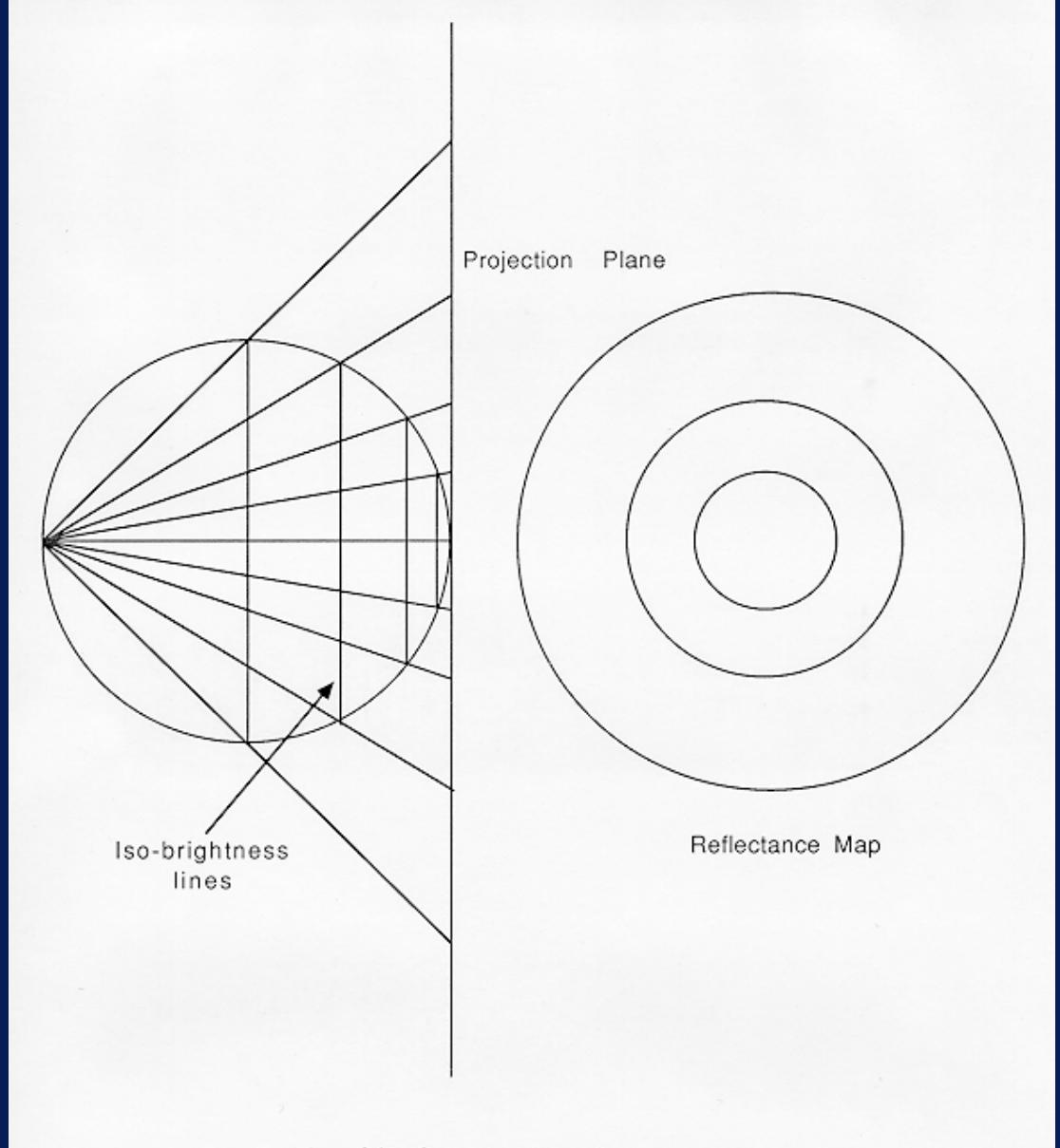
All points on this line have  
surface orientations and  
a brightness value which  
satisfy  $\cos i = E_l$

- *Lambertian sphere and iso-brightness lines*

- More often, a projection of these iso-brightness values onto a plane is used.
- This is called a *Reflectance Map*.

- The most commonly used projection is one onto a plane which is parallel to the viewers image plane and is a tangent to the sphere.
- This is effected by drawing a line from the point on the sphere opposite the viewer through the point to be corrected.

- *Generation of the Reflectance Map*



- Given a brightness value, we can determine a curve in the reflectance map (*i.e.* a circle). However we still cannot determine orientation.

- Thus, an extra constraint is required to uniquely determine the orientation of the imaged point from the brightness :
  - this is supplied by assumptions of surface smoothness (or continuity) that *the surface should not vary much from the surface direction at neighbouring parts.*

- Obviously we need some co-ordinates on the reflectance map to anchor the process and from which we can infer the orientation of neighbouring points.
- On occluding boundaries of objects without sharp edges, surface direction is perpendicular to the viewer's line of sight.

- All such directions project onto a circle of radius 2 on the reflectance map (see Figure 9.31) (*Slide 86*).

- we know immediately the surface orientation of every occluding contour point and, more importantly,
- the correspondence between a given occluding boundary point and the point to which it maps on the reflectance function.

- The required point of the reflectance map must correspond to the surface normal direction at the occluding boundary.

- Since the surface orientation changes smoothly, all points on the surface close to the occluding boundary must have an orientation which is not significantly different from that of the occluding boundary.

- The surface orientation of each point adjacent to the occluding boundary can now be computed
  - by measuring the intensity value and
  - reading off the corresponding orientation from the reflectance map *in a local area surrounding the map which corresponds to the current occluding boundary anchor point.*

- This scheme of local constraint is reiterated using these newly computed orientations as constraints, until the orientation of all points on the surface have been computed.

- There are a number of assumptions which must be made in order for the technique to work successfully,
  - the surface orientation must vary smoothly and
  - it must do so at the occluding boundary (the boundary of the object at which the surface disappears from sight).

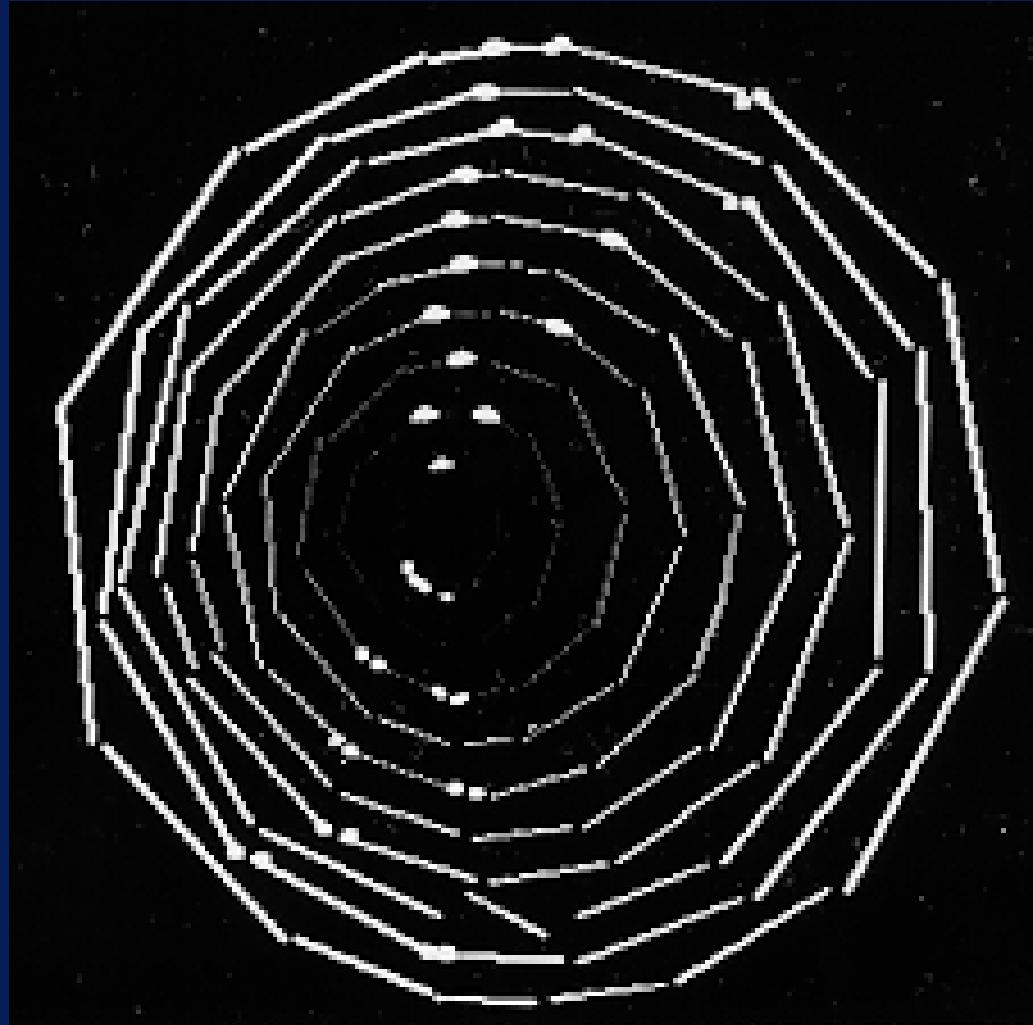
- Allied to this are the requirements :
  - that the reflective surface has a known albedo
  - that we can model its reflective properties, or alternatively, that we can calibrate for a given reflective material,
  - finally, that one must know the incident angle of light.

- This limits the usefulness of the technique for *general* image understanding.

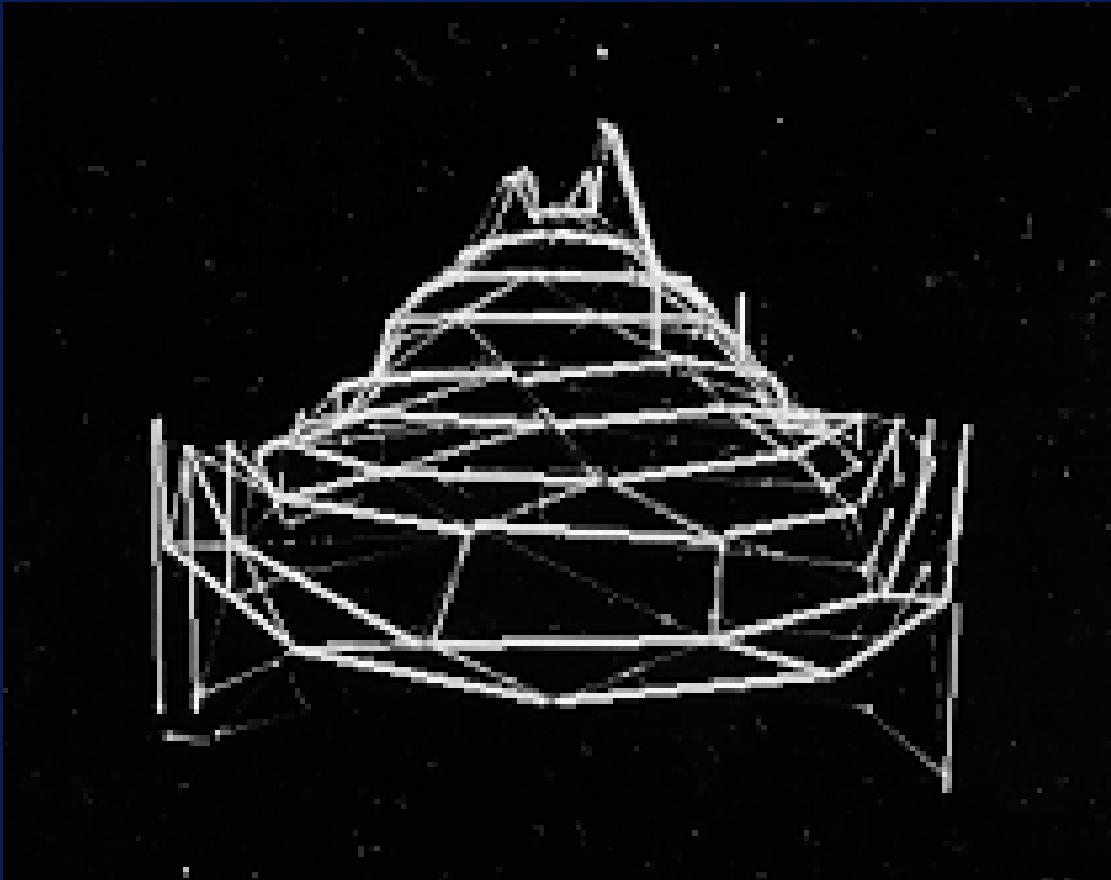
- There are other ways of estimating the local surface orientation.
- As an example of one coarse approach, consider the situation where we have a 3D raw primal sketch, *i.e.* a raw primal sketch in which we know the depth to each point on the edge segments ...

- if these raw primal sketch segments are sufficiently close
- we can compute the surface normal by interpolating between the edges
- generating a succession of planar patches
- effectively constructing a polyhedral model of the object (see section 9.3.4.3).

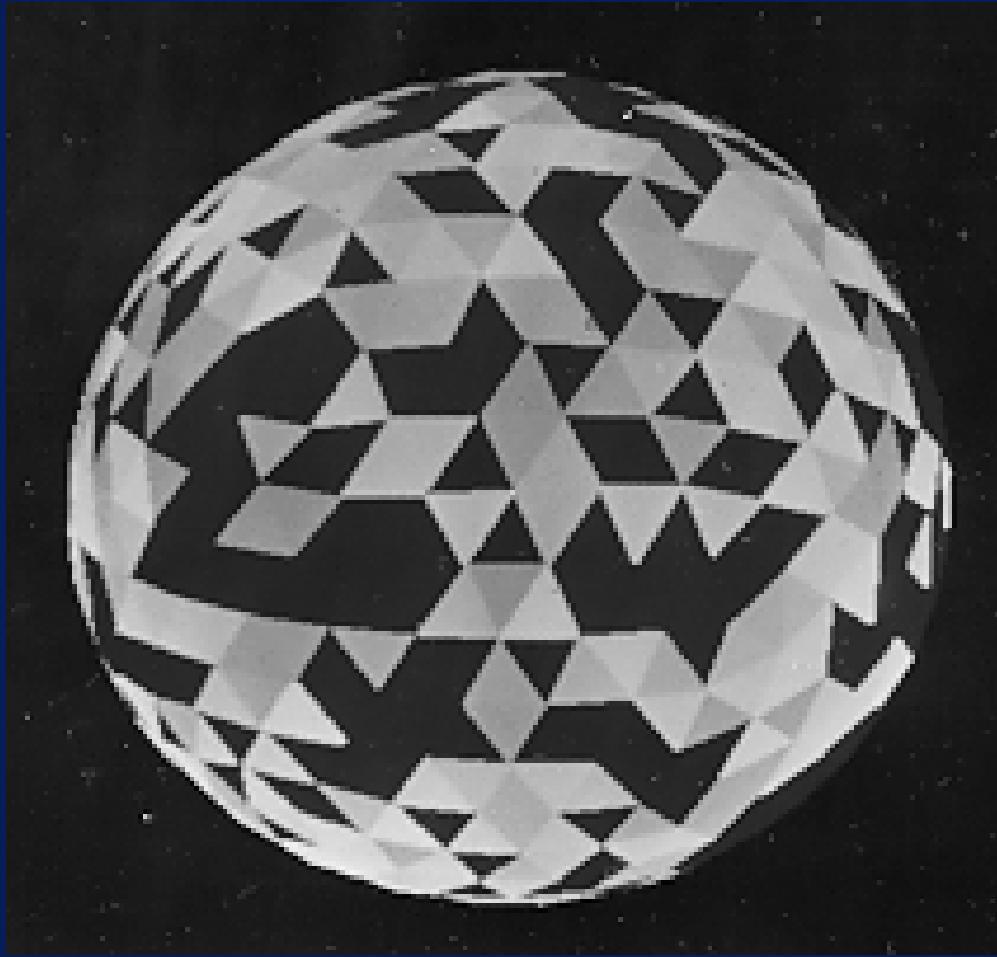
- The surface normal is easily computed by forming the vector cross product of two vectors in the plane of the patch (typically two non-parallel patch sides).



- *3D raw primal sketch of a striped cone.*



- *Reconstructed surface model*
  - *of the striped cone.*



- *Extended Gaussian image depicting the distribution of surface normals on the polyhedral model of the cone*

# Concluding Remarks

- You could be excused for thinking that Computer Vision is an end in itself, that is, that the task is complete once we arrive at our unambiguous explicit 3D representation of the world.
- *This is quite wrong.*

- Vision is no such thing :
  - it is merely part of a larger system which might best be characterised by a dual two-faced process of *making sense of / interacting with* the environment.

- *Without Action Perception is futile.*
- *Without Perception, Action is futile.*

- Both are complementary, but highly-related activities.
- Any intelligent action in which the system engages in the environment, *i.e.* anything it does, it does with an understanding of its action, and it gains this quite often by on-going visual perception.

- Image Understanding is as concerned with
  - cause and effect
  - with purpose
  - with action and reaction
  - as it is with structural organisation

- The issues of the temporal semantics of vision in contribution to and in participation with physically interactive systems, will not go away and must be addressed and understood someday.
- *Soon.*

# Performance Characteristics

# Performance Characteristics

- A measurement system consists of
  - ❖ A gauge
  - ❖ The people who use it
  - ❖ The techniques, procedures, software associated with making the measurement

# Performance Characteristics

- A good measurement system:
  - ❖ Total variability of the system must be small compared with:
    - The manufacturing process variability
    - The specification limits

# Performance Characteristics

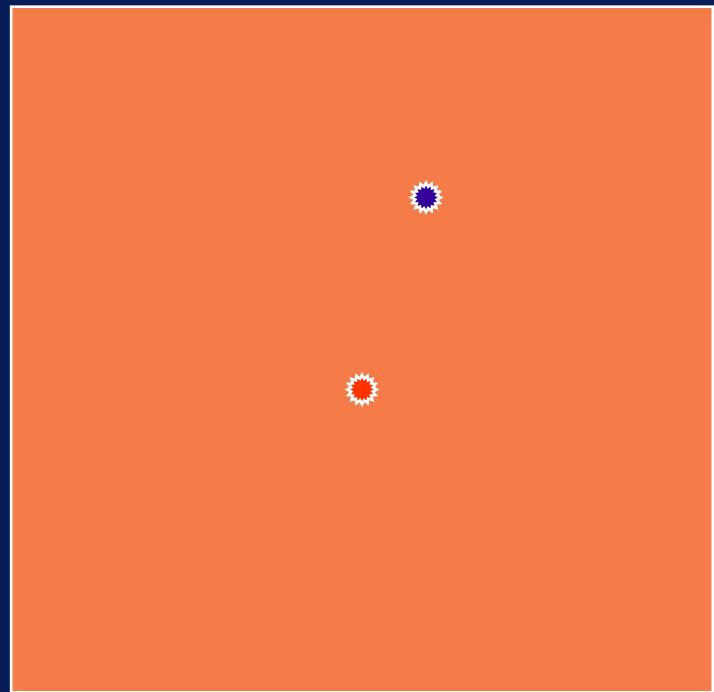
- A good measurement system:
  - ❖ Scale increments of the measuring device must not be larger than one-tenth of
    - The manufacturing process variability
    - The specification limits
- whichever is the smaller

# Performance Characteristics

- A measurement system capability must be assessed using appropriate statistical studies for
  - ❖ Accuracy
  - ❖ Repeatability
  - ❖ Reproducibility

# Performance Characteristics

- Accuracy

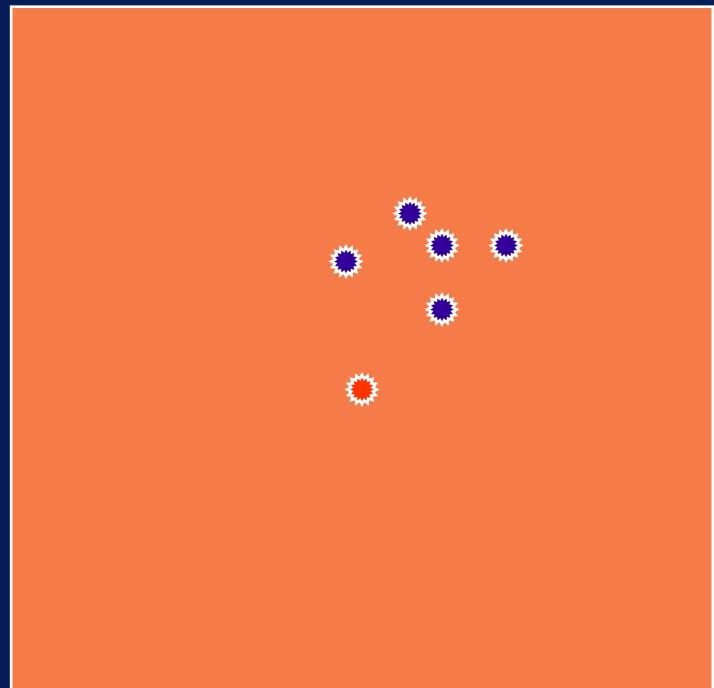


★ Actual Value

★ Measured Value

# Performance Characteristics

- Repeatability

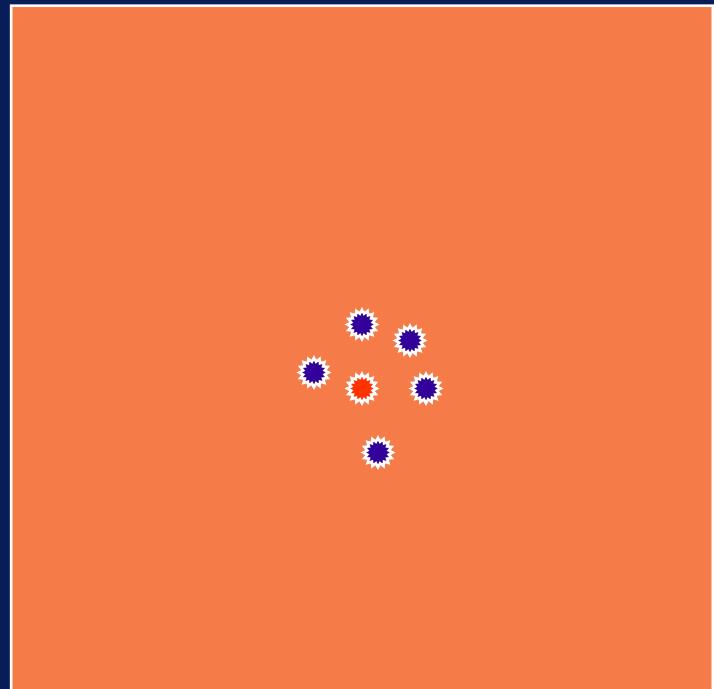


★ Actual Value

● Measured Value

# Performance Characteristics

- Accurate but not repeatable

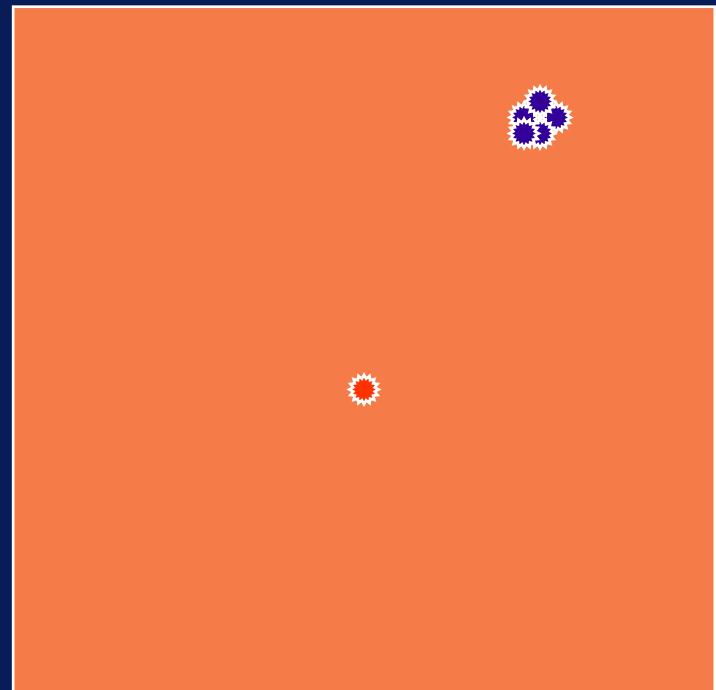


★ Actual Value

● Measured Value

# Performance Characteristics

- Repeatable but not accurate

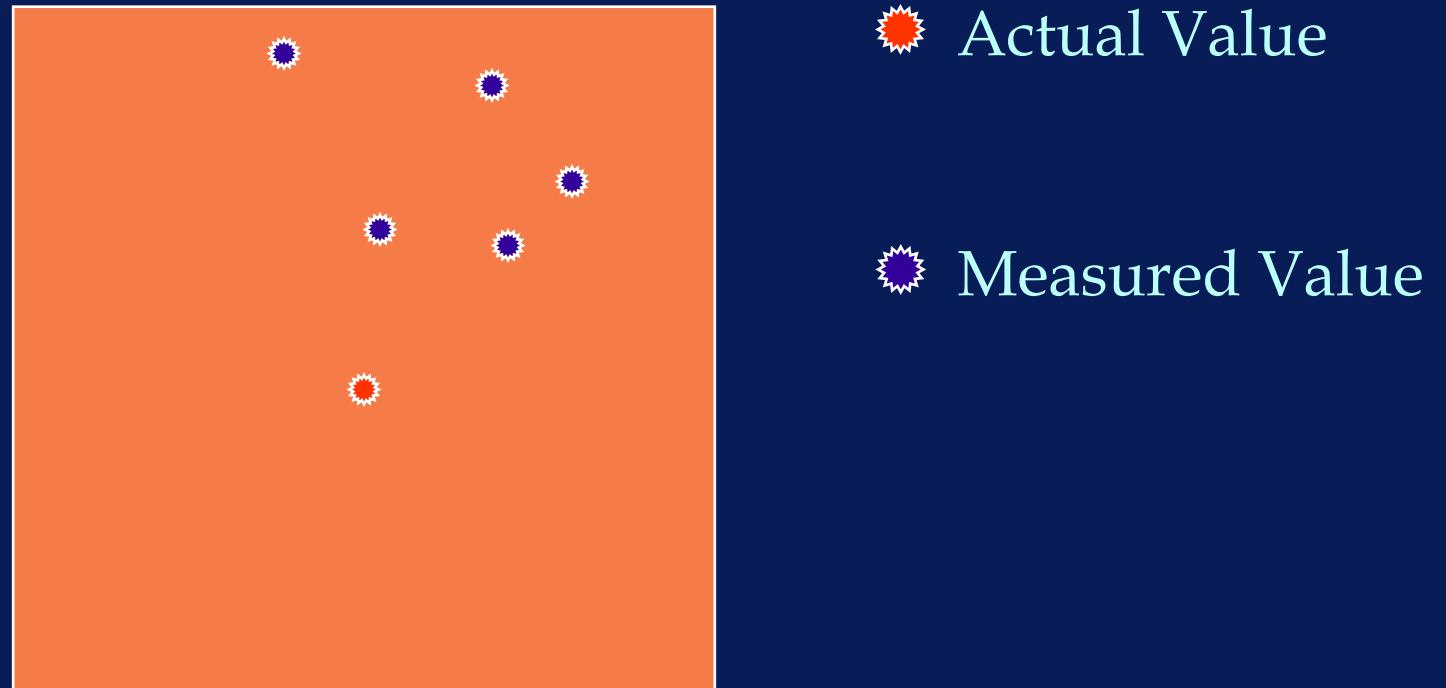


★ Actual Value

● Measured Value

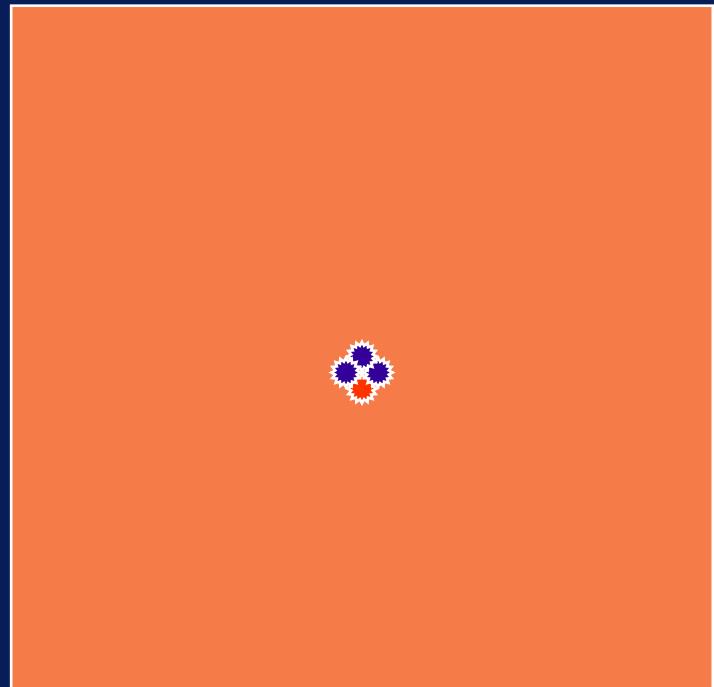
# Performance Characteristics

- Not accurate and not repeatable



# Performance Characteristics

- Accurate and repeatable



★ Actual Value

★ Measured Value

# Performance Characteristics

- Measurements will exhibit a statistical variation; typically a normal distribution
- ❖ Accurate measurements will have its mean centred on the actual value
- ❖ Repeatable measurements will have a small standard deviation

# Performance Characteristics

- Gauge Repeatability and Reproducibility (R&R) Studies
  - ❖ Investigate the ability of a gauge to repeat a measurement on the same part (repeatability)
  - ❖ The ability to give the same reading on the same part when used by different people (reproducibility)

# Performance Characteristics

- Gauge Repeatability and Reproducibility (R&R) Studies
  - ❖ Usually done using three operators
  - ❖ Making the same measurements
  - ❖ On the same set of 10 parts