1. INTRODUCTION

Elementary taxonomy

		Output	
		Image	**Description**
Input	**Image**	**Image processing**	**Image Analysis** **Image Understanding** **Computer Vision**
	Description	**Computer Graphics**	**Data Analysis** **Pattern Recognition**

Image processing

scope:	to form a digital image (from 3-dimensional scene to 2-dimensional image), to transform one 2D image into another.
aims:	digitisation, enhancement (for human viewing and also for subsequent computer processing), simple object extraction
examples:	remove warps or distortions, remove blur, smooth speckle or noise, improve contrast or other visual properties

Image analysis and image understanding

scope:	to obtain information from 2-dimensional image and about 2-dimensional image (rather than about 3-dimensional scene that it represents)
aim:	derive descriptions of the image contents
examples:	finding properties of the image or its parts (e.g. size, shape of image objects), finding key structures in the image (e.g. edges or regions of similar brightness),

Computer vision

scope:	to obtain information from an image (or a set of images) about 3-dimensional scene that the image(s) represents
aim:	derive meaningful and usable descriptions of the scene depicted in the image(s)
examples:	object sorting and picking by robots, object tracking, vehicle guidance, obtaining diagnostic information from medical images, face recognition, inferring object distance, inferring spatial relationships between objects, event description

Human visual system

Knowledge about the human visual system can be very useful to the designer of image analysis and computer vision systems. This knowledge is crucial for vision modelling in both computer vision and the cognitive and natural vision sciences.

When pictures are intended for viewing by humans, to carry out image processing appropriately we need to understand:

- how to assess subjective quality of an image
- how to ensure the required fidelity of a picture to an original scene

- the nature of the "weak spots" of the human visual system

When image is to be analysed for purposes of image description, we need the knowledge of the human perception so that:

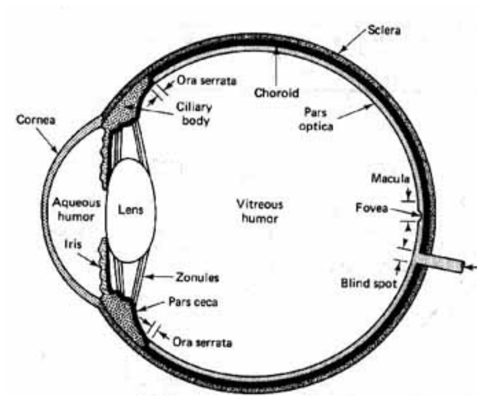
- parts extracted through image analysis correspond to those seen by humans
- parts can be described in terms used by humans

When developing computer vision systems for a particular purpose:

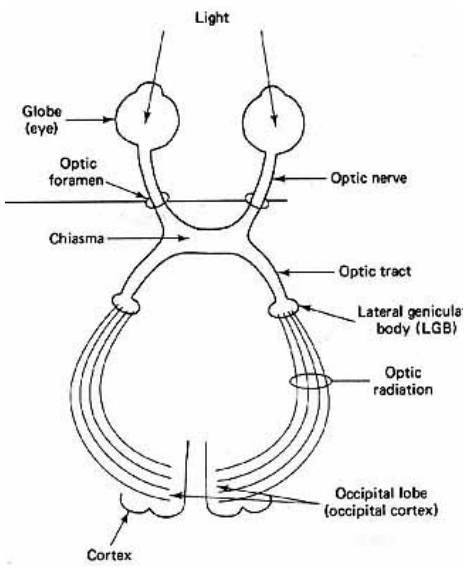
- the human visual system is a good model (it works!)

Anatomy and function

Eye



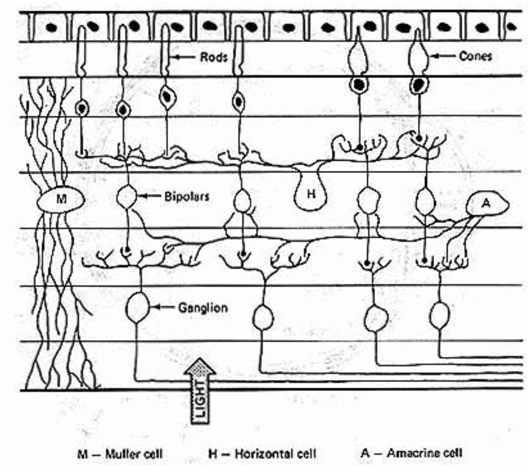
Visual pathways in the brain



Retina

Retina - receptors and optic nerve

- fovea is packed with cones, uniform resolution, 2000
- cones detect detail and colour, central, 6 million
- rods sensitive to light and motion, off centre, 120 million



Stereo

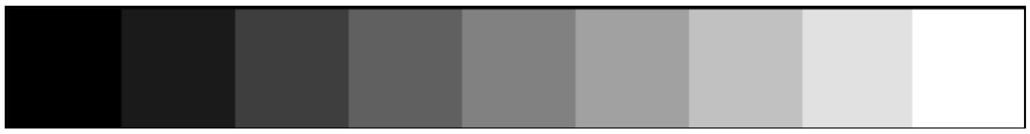
Two eyes look at the scene with angular separation of about 5˚. Convergence of two eyes changes with distance. Two eyes move together: they search for the target in saccades and converge together to focus. Each eye receives a slightly different view of the scene; this is called disparity and enables to perceive depth in the scene.

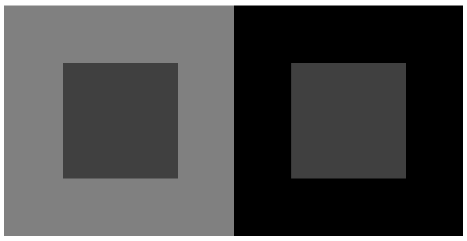
Perceptual phenomena

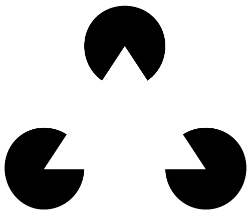
Visual input to which the eye responds is processed further by the brain in a very complex way. The nature of this processing is not yet very well understood. However, it is clear that vision is an “intelligent” process, as it is:

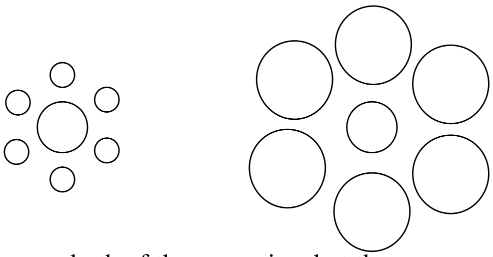
- active
- purposeful
- creative

Vision scientists identified many perceptual phenomena, some of which are illustrated below.


The luminance of the squares in this grey level scale increases linearly, but does their perceived brightness does not. Does the interior of each square appear uniformly grey?


The inner squares have the same luminance, but their apparent brightness is clearly different.

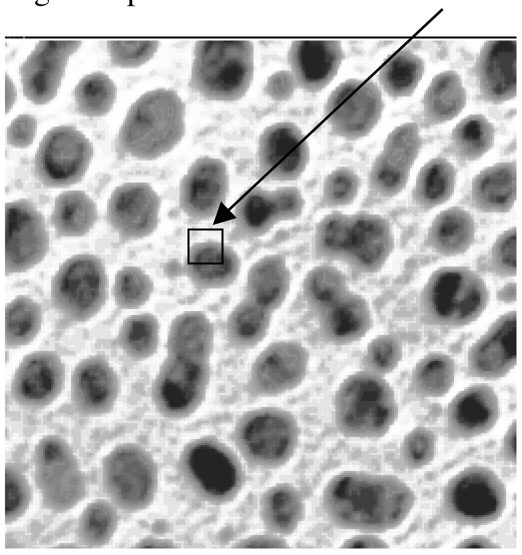

Interiors of regions based on edges seem to be filled even if edges are discontinuous.


The inner circles are both of the same size, but they appear to be different.

Digital image

Digital image is an image in digital form. Its representation in a computer normally takes a form of a 2-dimensional array (e.g. `int image[WIDTH][HEIGHT]`). Elements of the array are called *pixels*. A pixel is indexed by its spatial coordinates (e.g. `[x][y]`) and contains a value (`int pixel_value = image[x][y]`) which quantifies some property of an image (normally brightness) at a given location. Because images used to be stored only as monochrome (i.e. showing only brightness as a shade of grey, but not colour), pixel value is often referred to as a *grey level value*. Low pixel values correspond to dark shades of grey and high values correspond to bright shades.

An image and pixel values within a small window



204	163	163	163	204	204	234	234	234	204	204	204	204	204	204	204
204	163	163	163	204	204	234	234	234	234	234	234	234	234	234	204
204	163	163	204	204	204	234	234	234	234	234	234	234	234	234	234
204	204	204	204	234	234	234	234	234	234	234	234	248	248	248	234
204	204	204	204	234	234	234	204	204	163	163	204	204	204	234	234
234	234	234	234	234	234	234	204	163	134	134	134	163	163	204	204
234	204	163	134	134	119	112	102	98	94	94	98	102	106	112	119
234	163	119	112	106	98	90	84	77	69	69	77	84	90	98	106
163	112	102	94	87	80	69	77	60	54	54	60	65	69	80	87
134	102	90	80	65	60	54	39	31	31	31	39	47	54	77	65
112	94	80	65	54	47	39	31	31	31	31	39	47	47	54	54
106	87	65	54	39	31	31	31	31	31	31	39	47	39	39	39
94	77	60	39	21	21	31	31	31	31	31	31	39	31	21	21
87	65	47	31	15	21	31	31	31	31	31	31	31	21	15	15
80	77	39	21	15	21	31	31	31	21	21	21	31	21	15	15
77	60	39	21	15	21	31	31	31	21	21	21	31	21	21	15

Image sources

Digital images can be captured by digital cameras, scanners, or can be read directly from instruments. Pixel values normally quantify intensity of light, but they also can quantify other forms of energy such as infrared radiation, X-rays, ultrasound; or express other measures such as distance or time (e.g. radar images).

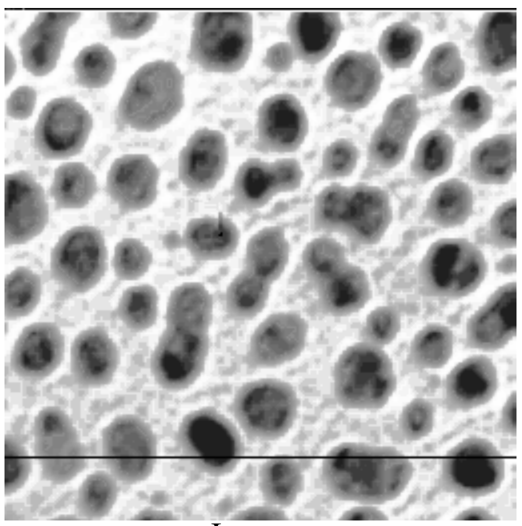
Simple tools for examining digital images

Pixel value

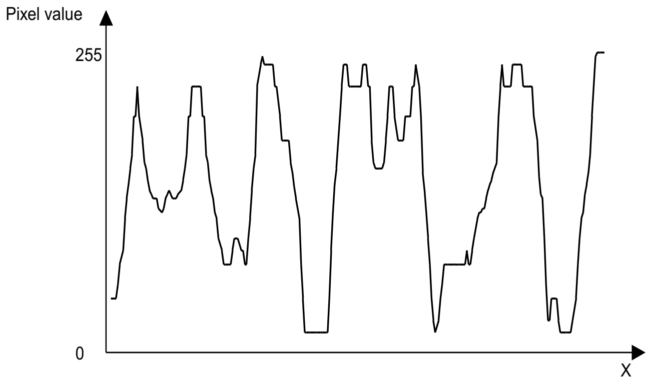
Most interactive packages provide a facility for looking at the value of a single pixel by positioning a cursor over the pixel.

Line profile

Pixel values along a line are displayed in the form of a graph.



Image



Line profile

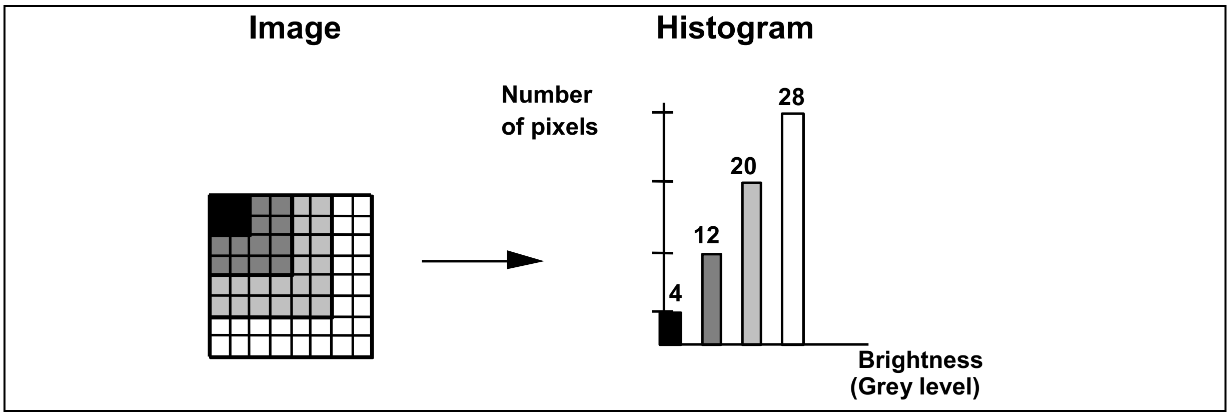
Outline of implementation:

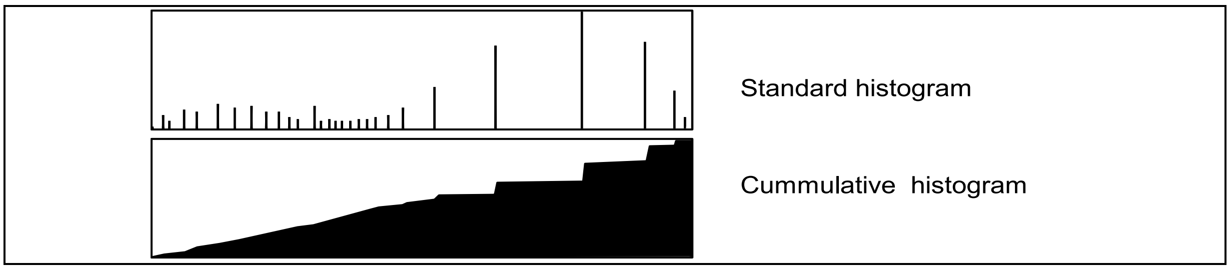
```
y = line_number;

for( x=0; x<IMAGE_WIDTH; x++ )
{
            PlotPointAt(x, image[x][y]);
}
```

Histogram

Histogram is a frequency distribution graph; it shows the number of pixels in the image having a particular grey level or a range of grey levels. The first figure below illustrates the idea. The second figure shows two histograms of the cell image above: the top is a standard histogram, as explained above; the bottom is a cumulative histogram, where each column indicates a number of pixels with values less or equal certain grey level value.





Outline of implementation:

```
int image[][];
image = new int[IMAGE_HEIGHT][IMAGE_WIDTH];
int histogram[];
histogram = new int[NO_OF_GREY_LEVELS];

for( g=0; g<NO_OF_GREY_LEVELS; g++ )
     histogram[g] = 0;

// Compute histogram
for( x=0; x<IMAGE_WIDTH; x++ )
     for( y=0; y<IMAGE_HEIGHT; y++ )
     {
             histogram[image[x][y]]++;
     }
// Plot histogram
for( g=0; g<NO_OF_GREY_LEVELS; g++ )
     PlotPointAt(g, histogram[g]);
```

Thresholding

Thresholding is a simple but effective method for separating objects from the background in an image. It is an example of *image segmentation*. Thresholding changes pixel values according to a simple rule:

$$I'(x,y) = 0 \quad \text{if } I(x,y) < T$$
$$I'(x,y) = 1 \quad \text{otherwise}$$

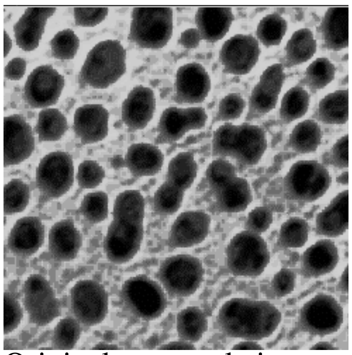
T is known as a *threshold value*. Finding a suitable threshold value is the key problem in this technique. The simplest solution is to build a histogram and find the valley between two peaks (why?) There exist methods for automatic threshold selection (see, e.g. Sonka, section 5.1.1 and HIPR).

Outline of implementation:

```
int image[][];
image = new[IMAGE_HEIGHT][IMAGE_WIDTH];
int new_image[][];
new_image = new[IMAGE_HEIGHT][IMAGE_WIDTH];

for( x=0; x<IMAGE_WIDTH; x++ )
     for( y=0; y<IMAGE_HEIGHT; y++ )
    {
       if( image[x][y] < threshold )
            new_image[x][y] = 0;
       else
            new_image[x][y] = 1;
    }
```

Example:


Original grey scale image

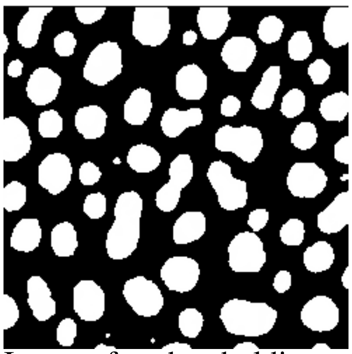


Image after thresholding

We shall study thresholding further in Unit 7.

Further reading and exploration

Bruce, Green & Georgeson, Chapter 1. (Biological visual systems)
Umbaugh, Chapter 1 (Introduction, human vision)

HIPR

- User Guide
- Image Analysis->Intensity Histogram.

CVIP

- Load any image
- Examine pixel values in monochrome and colour images (middle mouse button)
- Create and analyse image histogram for monochrome and colour images (View Histogram)