M30242 Graphics and Computer Vision

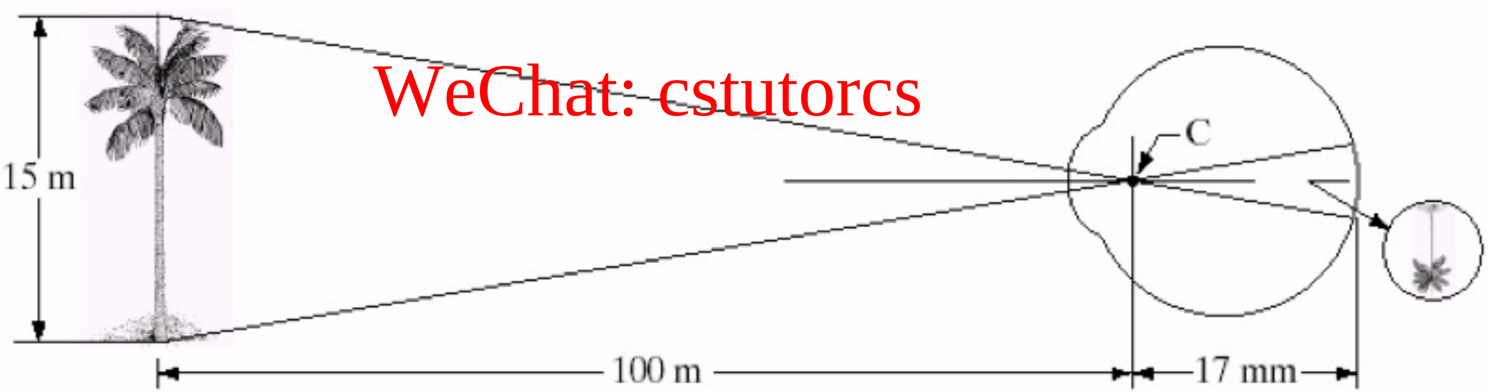
Lecture 2 Image Formation and Basic Image Operations

Overview

- Image formation, image coordinate system.
- Different image formats in Matlab.
- Grayscale image processing
  - Thresholding,
  - Histogram.
- Binary image operations
  - Pixel-wise properties: neighbourhood and connectedness,
  - Labelling,
  - Dilation and erosion,
  - Opening and closing.

Image Formation

- Geometrically, images are formed through perspective projection.
  - All light pass through the same point - the centre of projection.
  - It is a good approximation and abstraction of the optics of real lenses/eyes.

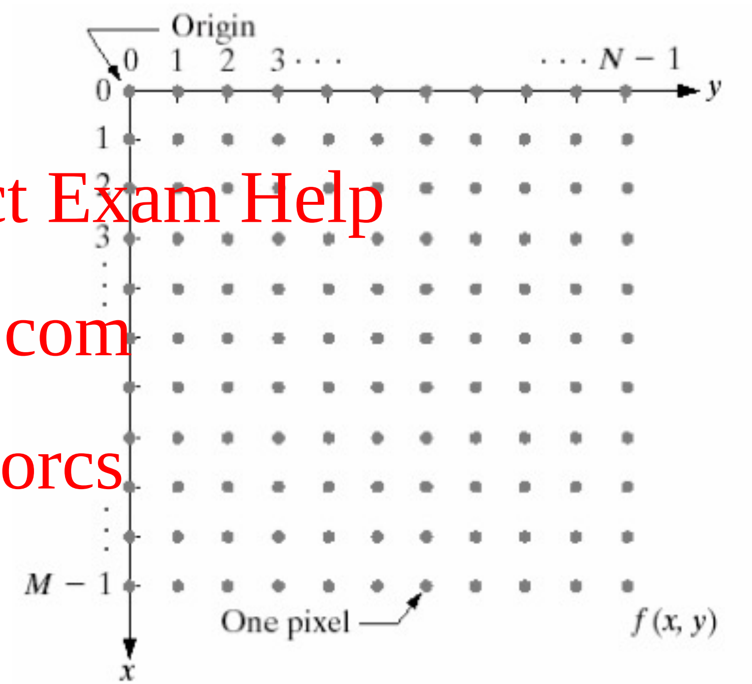


$$\frac{15m}{100m} = \frac{height}{17mm}$$

therefore h=2.55mm

Image Coordinate System

- Image coordinate system:
  - the origin is at upper left;
  - x is usually vertical.

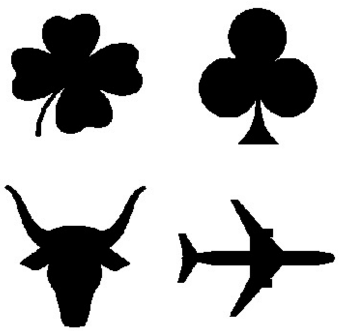


- By this coord. system, an image can be represented as a 2D matrix: $\boldsymbol{I}(r,c)$, where
  - r: index for rows,
  - c: index for columns.

- Matlab starts indices from (1,1), not (0,0) as in C.

Different Images: Binary Image

- Each pixel can have only two levels of intensity: either 0 or 1, which can be expressed as a binary number, hence comes its name.
- It is also called a *bi-level* image.
- It can be acquired by scanners or through thresholding *grayscale* images.
- Useful for detecting object from background.

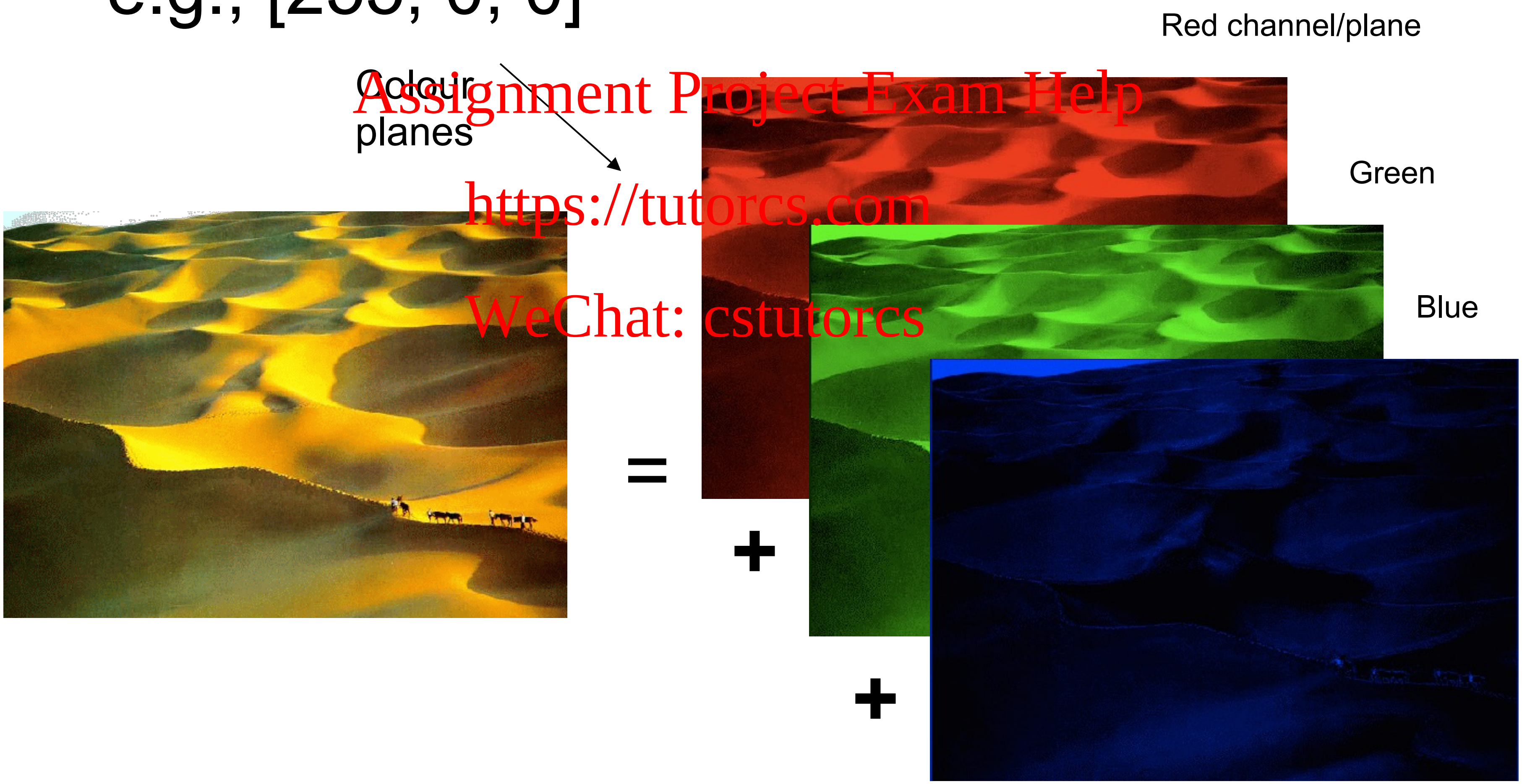


Different Images: Grayscale Image

- Intensity value of a pixel can range 0-255, but other values are possible depending upon its format:
  - [0, 255] (uint8), [0, 65535] (uint16), [-32768,32767] (int16)
  - [0.0,1.0] (single precision (32 bits) or double precision (64 bits))
- As a pixel has only a brightness intensity, a grayscale image does not have colours.

Colour Image

- A pixel has three colour (RGB) values, e.g., [255, 0, 0]



Colour Image

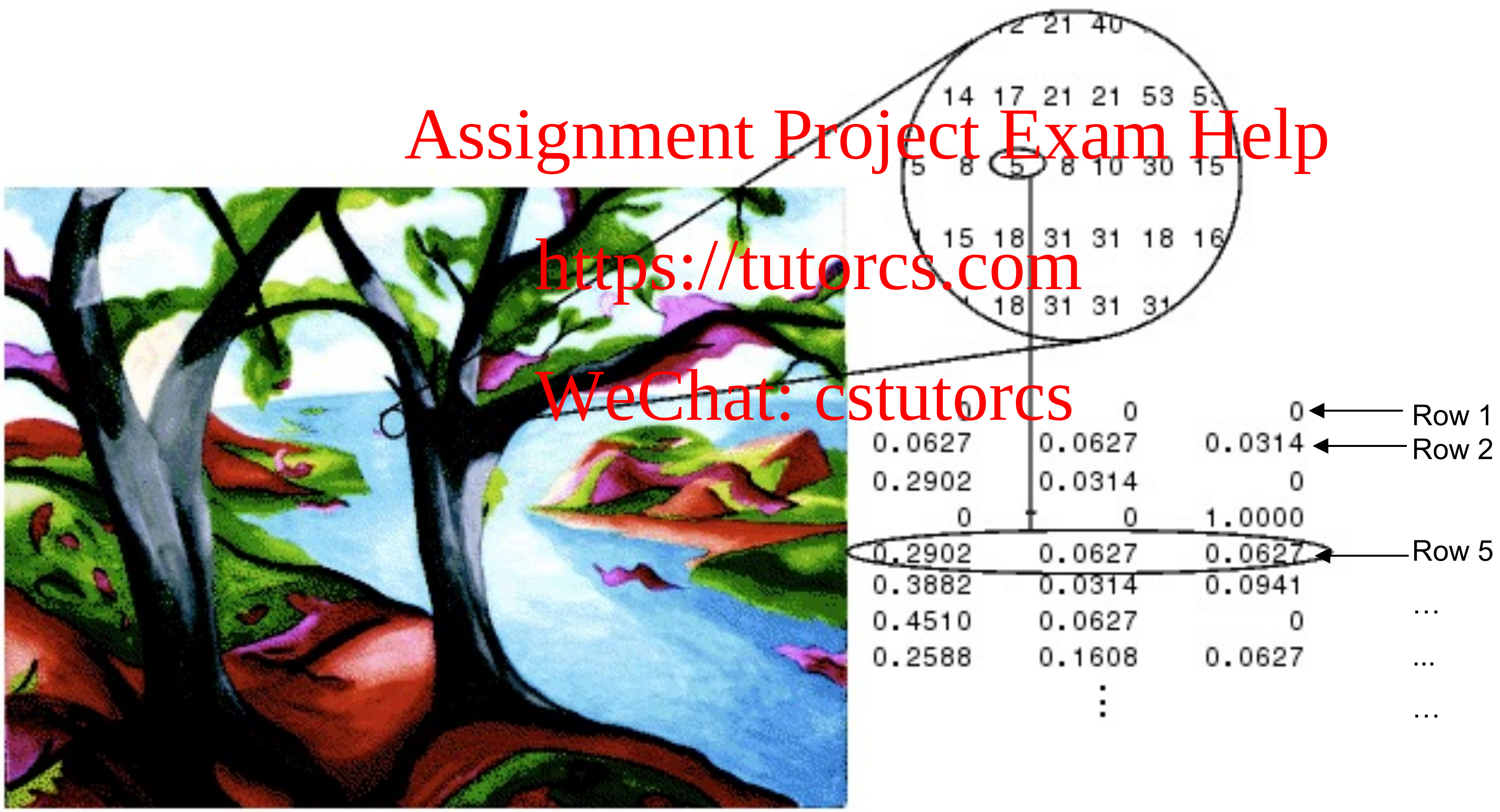
- In Matlab, colour images are usually stored as three dimensional arrays of sizes M-by-N-by-3 (other formats are available)
- The elements of the arrays could have different ranges of values depending upon the format of an image:
  - uint8 ([0, 255]),
  - uint16 ([0, 65536]), or
  - single/double ([0.0,1.0])

Colour Image: Indexed

- An indexed image consists of two arrays
  - An M-by-N image matrix of numbers.
    - M X N is the image size (M-row by N-column of pixels).
    - The numbers are usually integers (uint8, uint16), but it could be values of other types.
    - The numbers are used to *index into* the Colormap matrix.
  - A colormap – an two dimensional array num-by-3 matrix of double ([0.0, 1.0]), where
    - num is the number of rows (the number of colours) – the *length* of a particular colormap;
    - each row is a tuple of RGB values (called a colour vector) that defines one color. The *k-th* row of the colormap defines the *k-th* color.

Colour Image: Indexed

Image matrix



Colourmap (matrix)

Binary Image Applications

- Many applications are implemented by processing binary images.
- Typical applications:

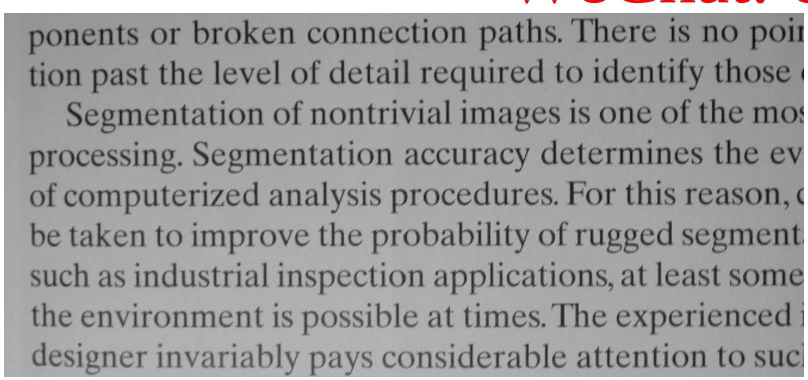
  - detect objects on a conveyor belt;
  - recognise characters and texts, maps;

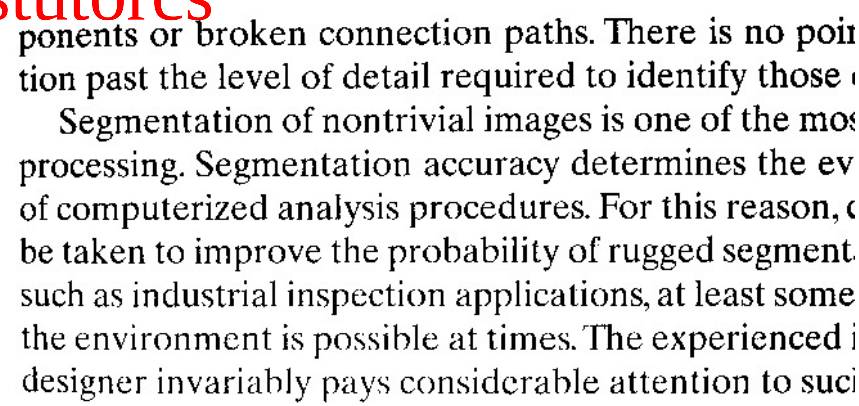
  - read fingerprints;

  - check circuit boards, and etc.
- In such applications, we are interested in the presence of objects, not their colours or levels of brightness or grayscales (shades).

An Example

- To read and recognise the text, an important thing is to separate the text from the background.
- The colour or grayscale of the font or background does not matter.
- For such applications, binary images suffice.


ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason,
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced
designer invariably pays considerable attention to suc


ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason,
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced
designer invariably pays considerable attention to suc

Obtaining A Binary Image

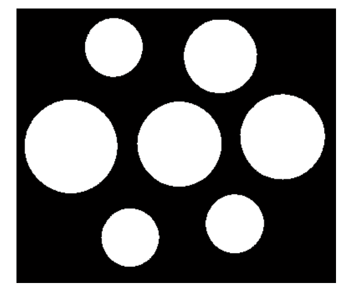
- Images are normally captured as grayscale or colour images.
- Binary images are obtained by processing them.

Colour image

Converted to

Grayscale image

Converted to



Binary image

Thresholding Grayscale Images

- Binary images can be obtained by “thresholding” grayscale ones:
  - First, we decide a grayscale level - a *threshold*
    - In many cases, a threshold is expressed as the percentage of the range of grayscale values. E.g., a gray level of 128 would be a threshold of approximately 0.5 (50%) of the range of 256.
  - Then, go through each pixel of an image and compare the value of the pixel with this threshold.
  - If the pixel value < threshold, then assign value "0" to the pixel indicating it belongs to background (or foreground).
  - Otherwise, assign "1" to the pixel indicating it belong to foreground (or background).
- A practical issue: how do we choose a threshold so that we can effectively separate the foreground from the background?

Choose A Threshold

- How do we decide a threshold so that the foreground (the tall tree) can be separated from the background (ground and bushes)?

Grayscale image

0.1

0.2

0.3

0.5

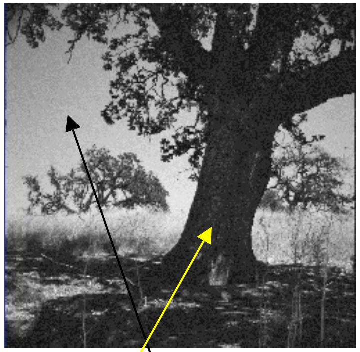
0.7

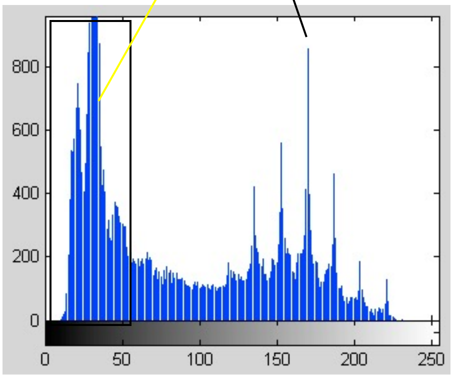
Pixel Histogram

- Deciding a threshold of best separation often requires us to get some statistics of the pixels – pixel histogram.

- Pixel histogram shows the total number of pixels of each gray level, e.g. there are 10 pixels having gray value 0, 15 pixel having gray value 1, …, 2 pixels having gray value 255.

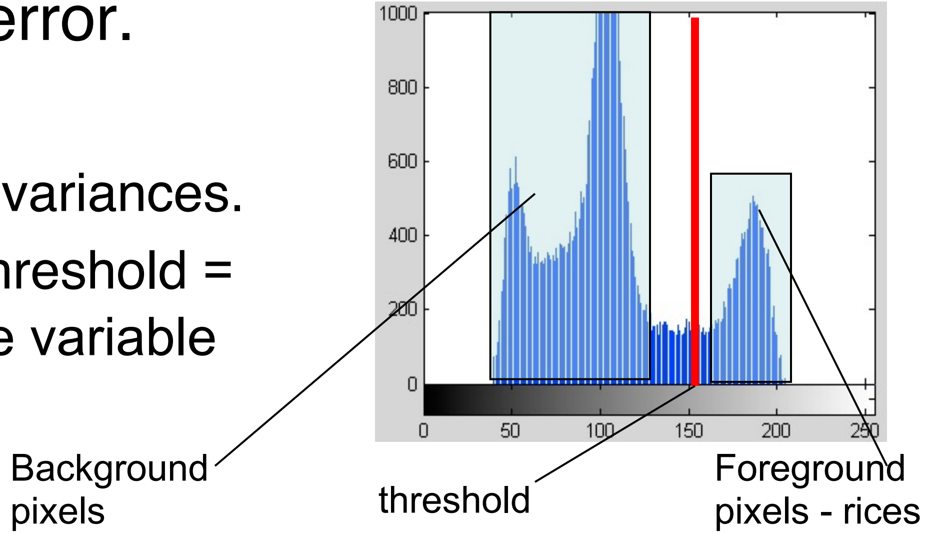
- The histogram on the right says:
  - Most of the pixel values fall between 10 and 225;
  - Lots of pixels have an intensity value lower than 60 (trees and ground).





Determine the Threshold

- Determine the threshold is NOT a trivial issue.
- We usually assume a *bimodal* (has two peaks) histogram and choose a threshold that separates modes (peaks) of histogram and minimizes classification error.
- Otsu algorithm
  - Minimize the within-group variances.
  - Used by Matlab function threshold = graythresh(I) where I is the variable for an image.



Obtaining A Binary Image

Procedure

- Read an image into a variable:

  img = imread('some_pic.jpg');

- Convert the image to grayscale one if it is a coloured one:

  gray_img = rgb2gray (img); or use im2gray (img);

- Compute a threshold level:

  level = graythresh(gray_img)

- Convert the grayscale image to binary by thresholding:

  bw_img = im2bw(gray_img, level).

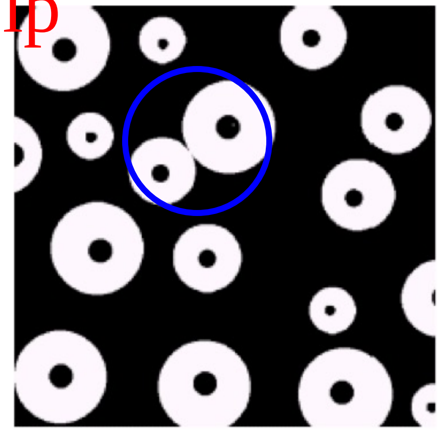
Identify Regions In Binary Images

- Computer vision applications often involve counting and/or measuring shapes of 2D *regions* of binary images.

- Key question: what characteristics define a region?

Define A Region

- A region is defined as the collections of connected pixels that possess the same (or more broadly, similar) values.



- How do we measure whether or not if a pixel is connected with others?

- We consider the immediate neighbours of the pixel.

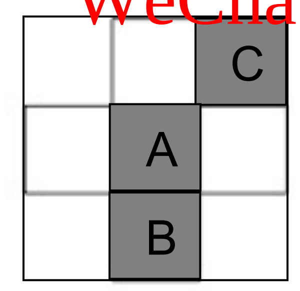
Pixel Neighbourhood

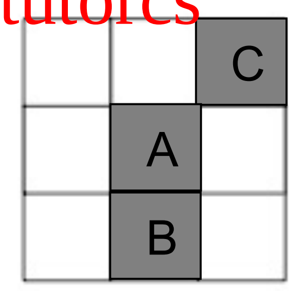
- Neighbourhood of pixels
  - 4- and 8-neighbourhood/adjacency

- The way of the neighbourhood is defined determines the connectedness of a pixel with its neighbours.
  - A pixel can be 4- or 8-connected with its adjacent pixels

Connectedness

- Based on neighbourhood definition, we can determine if an image region is connected with, or separated from, others.
- E.g





4-neighbourhood:

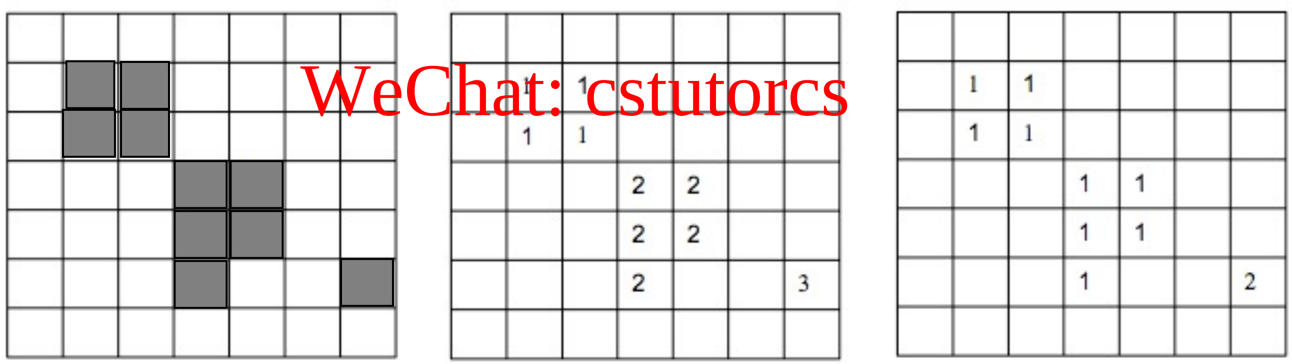
Pixel A is connected to pixel B but not to C

8-neighbourhood:

Pixel A is connected to both pixel B and pixel C

Labelling

- The operation that creates image regions according to the specified connectedness is called labelling.

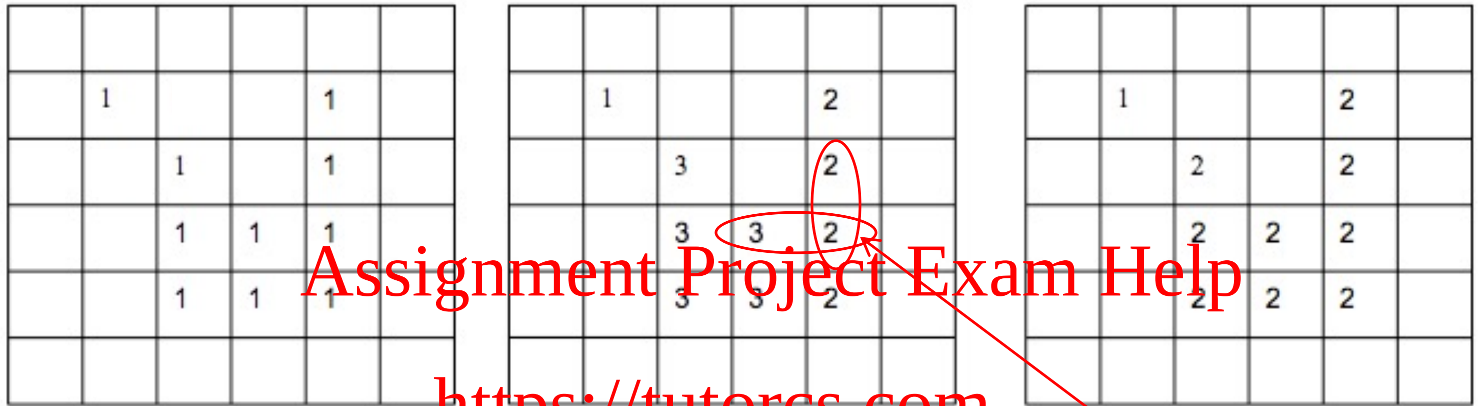


Binary image | Labeled image (4-connected) | Labeled image (8-connected)

Labelling Algorithm

- It is a two-pass process.
- The first pass goes through image pixel-by-pixel to assign ***temporary*** region labels to pixels according to the specified connectedness
- It decides the region label for a pixel by checking the **top** and **left** neighbour (an isolated pixel is considered as a new region) and ***record equivalences***
- The second pass replaces the temporary labels with equivalence labels.

Example (4-Neighbourhood)



Binary image

Temporary labels after 1st pass

Final (equivalence) labels after 2nd pass

- Equivalences between temporary labels are recorded during 1st pass
- Temporary labels are replaced with equivalence labels during 2nd pass

Equiv	Temp
1	1
2	2,3

An equivalence is found at this pixel

This algorithm is implemented as bwlabel(), which labels **binary images only.**

B(r,c): input binary image

L(r,c): output labelled image

The algorithm assumes the foreground pixels are white (value 1)

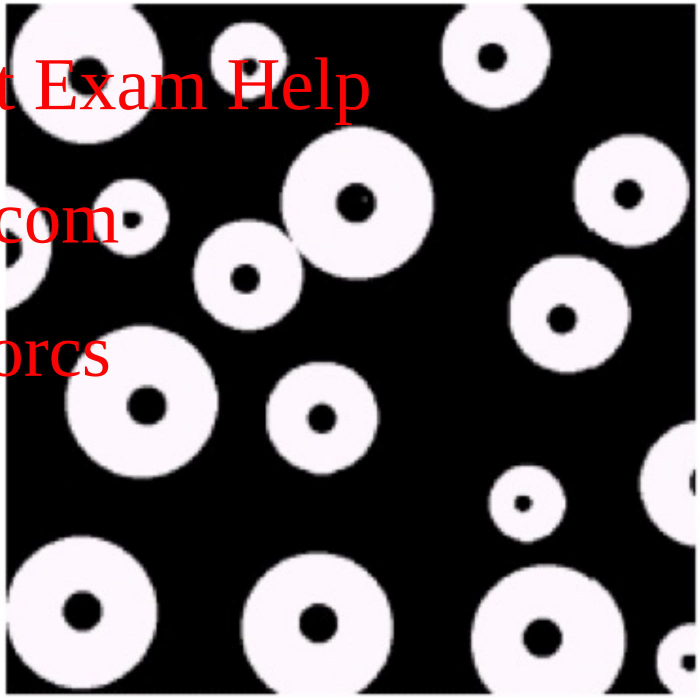
```
for c=1 to MAXCOL {
    for r=1 to MAXROW {
        if B(r,c) == 0 then
            L(r,c) = 0;          % if pixel not white, assign no label
        else {
            % check if top neighbour only has been labelled
            if  L(r,c-1) == 0  AND  L(r-1,c) != 0  then
                L(r,c) = L(r-1,c);      % assign to top label

        % check if left neighbour only has been labelled
        else if  L(r,c-1) != 0  AND  L(r-1,c) == 0  then
            L(r,c) = L(r, c-1);    % assign to left label

        % check if neither neighbour has been labelled
        else if  L(r,c-1) == 0  AND  L(r-1,c) == 0  then % both neighbours are not labelled
            L(r,c) = NumLabel++;          % create new label

        % we have the case where both neighbours are labelled
        else  {
            L(r,c) = L(r-1,c);     % assign to one of them
            RecordEquivalent( L(r,c-1),  L(r-1,c) );
        }
      }
    }
}
```

	r-1,c	
r,c-1	**r,c**	r,c+1
	r+1,c	

Assign the current pixel the same label as its top or left neighbour

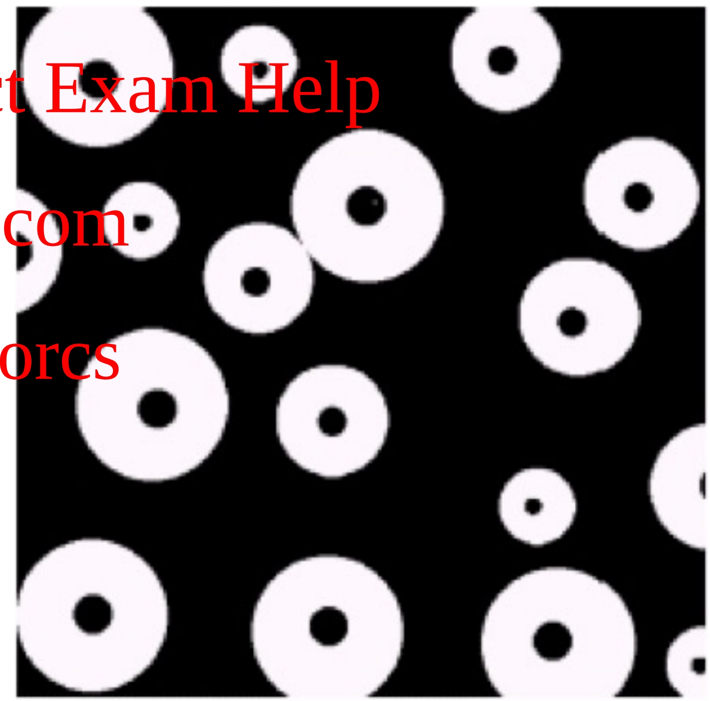
An Example

- An image of washers on a conveying belt.
- Questions that need to be answered in practical applications:
  - How many washers in this image?
  - What are the positions of the washers?



Cont'd

- A possible Matlab procedure that could get the the job done (i.e., counting the washers) would involve the following functions:
  - im2bw() or imbinarizer()(convert the image into a binary image so that the labelling algorithm can be applied)
  - bwlabel()



Readings

- Shapiro, L.G., Stockman, G.C., Computer Vision, Prentice-Hall, 2001, ISBN 0-13-030796-3
  - Chapter 2
  - Chapter 3: Section 1~4