Mid-Term Review

Definition: What is computer vision?

- **Computer vision** aims to duplicate the effect of human vision by electronically perceiving and understanding an image.
  - The study of recovering useful properties of the world (what, where)
  - from one or more images (by looking)
  - with an algorithmic level of specification

- Deals with the development of the **theoretical** and **algorithmic** basis by which useful information about the 3D world can be automatically extracted and analyzed from a **single** or **multiple** of 2D images of the world.

Computer Vision, Also Known As ...

- Image Analysis
- Scene Analysis
- Image Understanding

- **Problems the computer vision solves**
  - Computing properties of the world from one or more images
  - Properties of interest:
    - geometric (shape, position),
    - photometric (surface reflectance)
    - dynamic (velocity)

Some Related Disciplines

- Image processing
- Pattern recognition
- Computer graphics
- Robotics
- Artificial Intelligence

Why is Computer Vision Difficult?

- It is a **many-to-one** mapping
  - A variety of surfaces with different ***material*** and ***geometrical*** properties, possibly under different ***lighting*** conditions, could lead to identical images
  - Inverse mapping has non unique solution (a lot of information is **lost** in the transformation from the 3D world to the 2D image)

- It is **computationally intensive**
- We **do not understand** the recognition problem

Practical Considerations

- **Impose constraints** to recover the scene
  - Gather more data (images)
  - Make assumptions about the world
- **Computability** and **robustness**
  - Is the solution computable using reasonable resources?
  - Is the solution robust?

Computer Vision Applications

- Industrial inspection
- Surveillance, monitoring and security
- Person recognition (automated fingerprint, face, iris,…)
- Human computer interface (Gesture recognition)

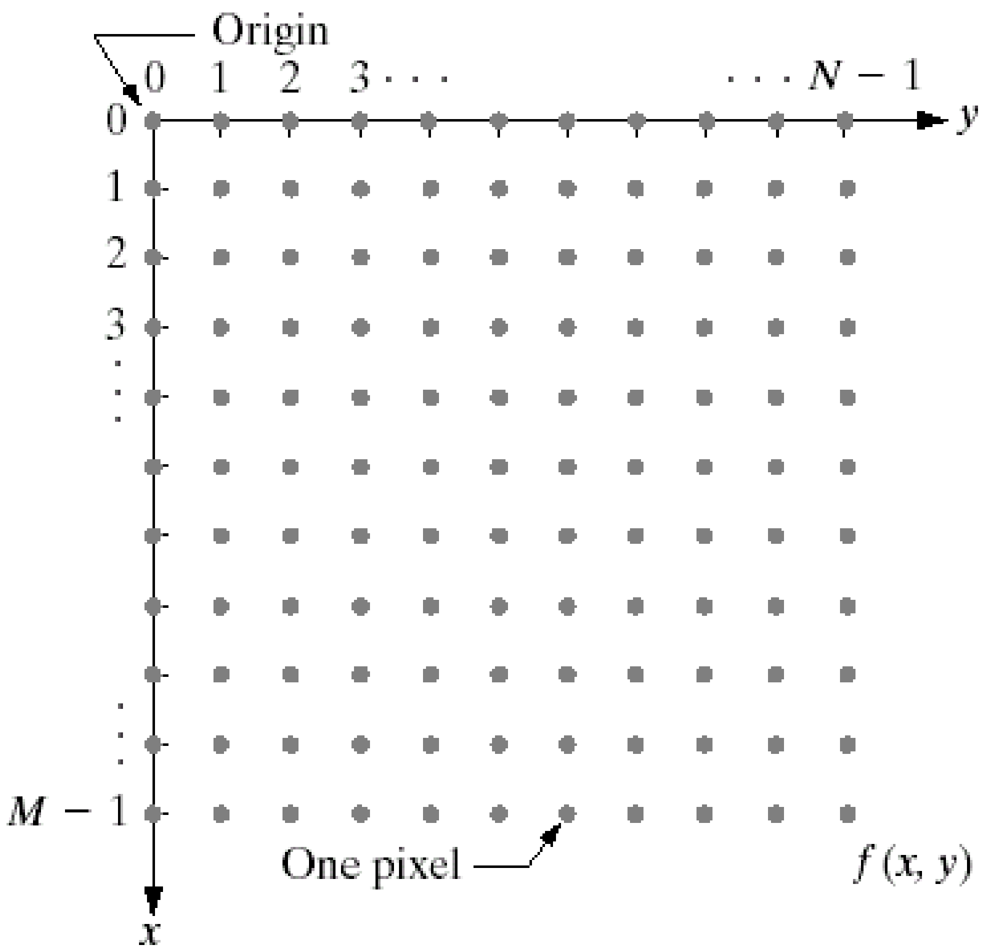
How are images represented in the computer?

- A scalar function may be sufficient to describe a monochromatic (單色的) image, while vector functions are to represent, for example, color images consisting of three component colors.

Image digitization

- ***Sampling*** means measuring the value of an image at a finite number of points.
- ***Quantization*** (gray-level) is the representation of the measured value at the sampled point by an integer.

Image coordinate system



Binary Image Processing -Advantages

- Easy to acquire
- Low memory requirement
- Simple processing

Binary Image Processing-disadvantages

- Limited application
- Cannot extend to 3D
- Specialized lighting is required for silhouettes (輪廓）

- In many algorithms, not only the value of a particular pixel, but also the values of its **neighbors** are used when processing that pixel.
  - ***4-neighbors***
  - ***8-neighbors*** of a pixel

	N	
W	*	**E**
	S	

4-neighbors

NW	**N**	**NE**
W	*	**E**
SW	**S**	**SE**

8-neighbors

Applying Masks to Images (filtering)

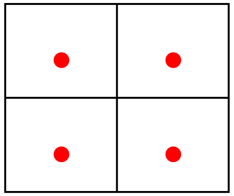
- ***Mask:*** A mask is a small matrix whose values are called ***weights***
- Each mask has an ***origin***, which is usually one of its positions.
  - The origins of symmetric masks are usually their center pixel position.
  - For non-symmetric masks, any pixel location may be chosen as the origin (depending on the intended use)

Counting the Objects in an Images

- Counting the number of foreground objects is an equivalent problem that can be performed with the same algorithm by merely swapping the roles of the two sets: **E** and **I**.
- **E** --- the external corner patterns are 2×2 masks that have three 0s and one 1-pixel.

Algorithm: Counting foreground objects

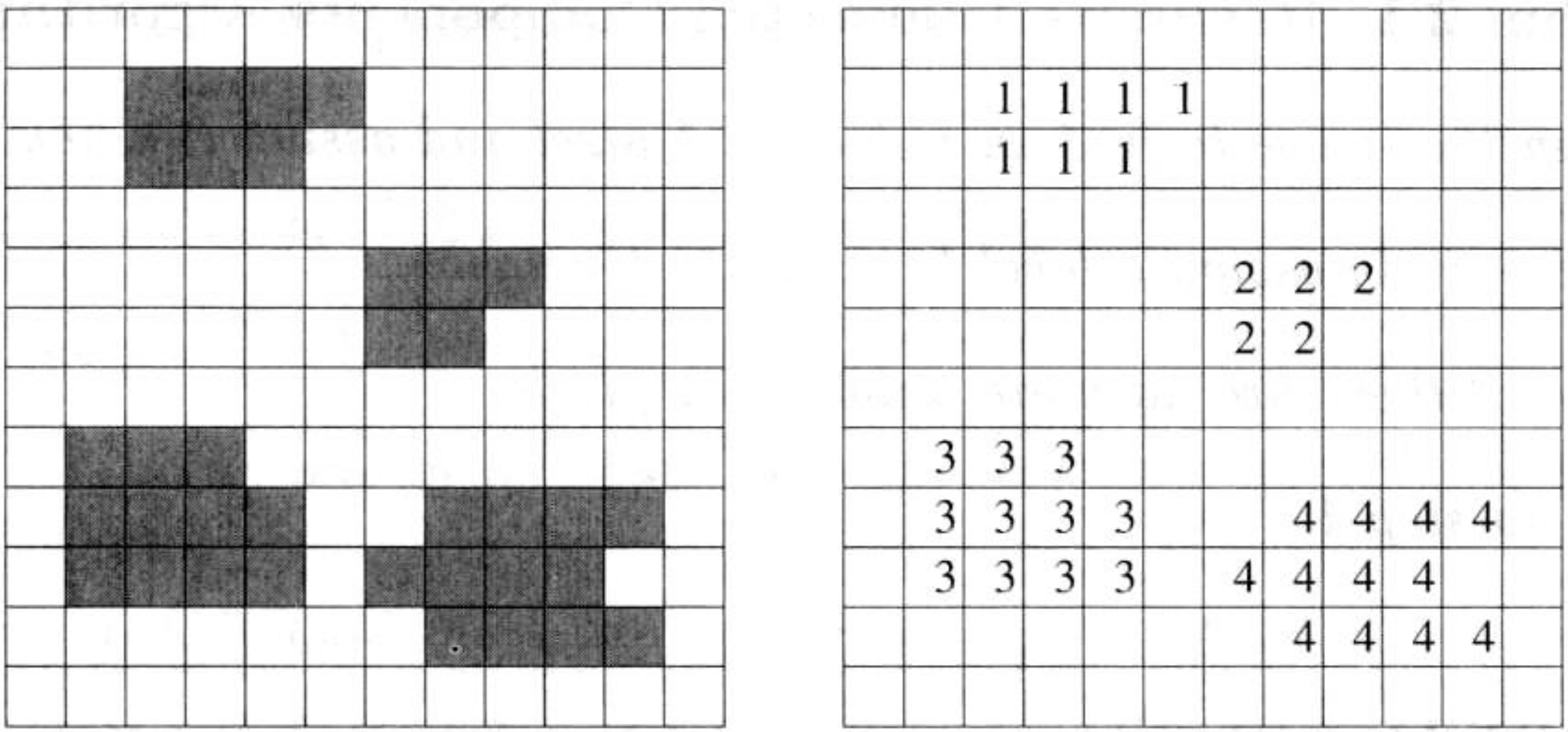
- **Compute the number of foreground objects of binary image B.**
- Objects are 4-connected and simply connected.



- **Step1: Counting the number of external corners (E) in region of 2×2 sub-images**
- **Step2: Counting the number of internal corners (I) in region of 2×2 sub-images**
- **Step3: Counting the number of foreground objects in whole image:**

The number of foreground objects = $\left| \dfrac{I - E}{4} \right|$

Connected Components Labeling



(a) (b)

Binary Image Morphology

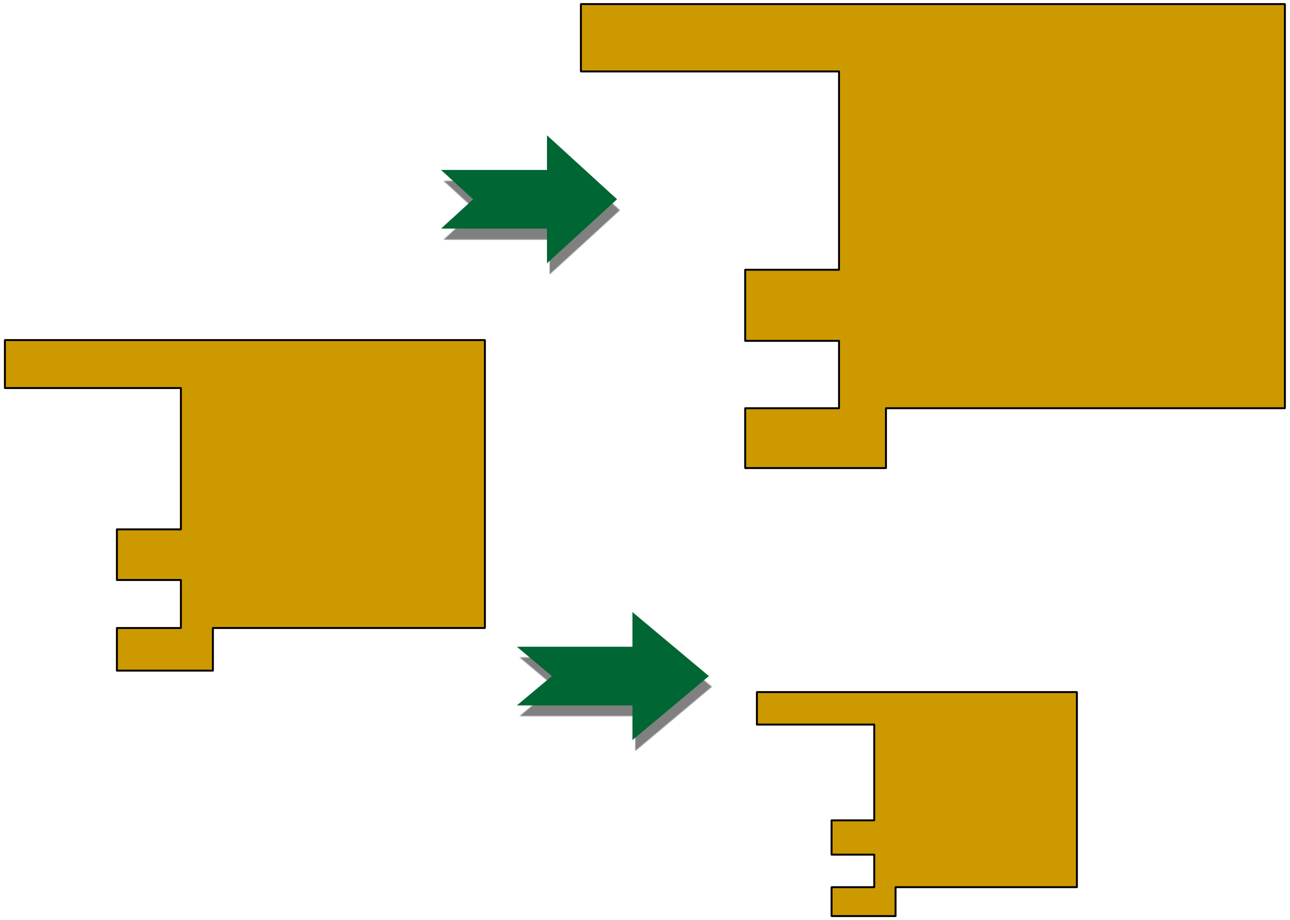
- **Definition ---** The word *morphology* 形態學 refers to form and structure; in computer vision it can be used to refer to the shape of a region.
  - The operations of *mathematical morphology* were originally defined as set operations and shown to be useful for processing sets of 2D points.
  - In computer vision, we define the operations of binary morphology and show how they can be useful in processing the regions derived from the connected components labeling operation.

Basic Operations

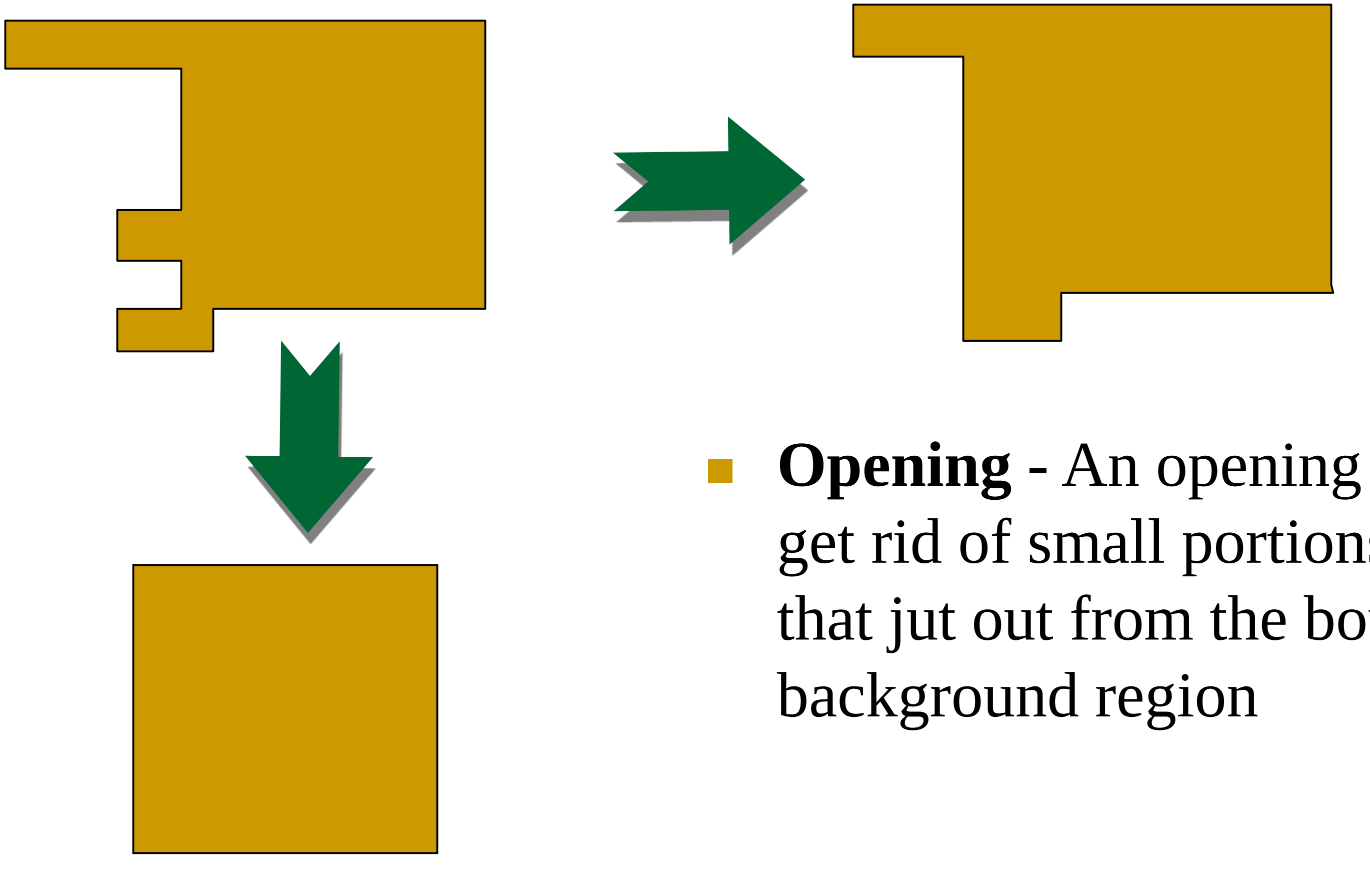
- The basic operations of binary morphology are ***dilation, erosion, closing***, and ***opening*** 膨脹、腐蝕、開、閉運算

- **Dilation** -
  A dilation operation enlarges a region

- **Erosion** -
  An erosion makes the region smaller.



- **Closing** - A closing operation can close up internal holes in a region and eliminate *bays* along the boundary



- **Opening** - An opening operation can get rid of small portions of the region that jut out from the boundary into the background region

Dilation Operation

- **The dilation of binary image B by structuring element S is denoted by B ⊕ S and is defined by**

$$B \oplus S = \bigcup_{b \in B} S_b$$

- ***Dilation*: if at least one pixel of the SE is inside the region, *f[i,j] = 1***

  –dilation expands the region

Erosion Operation

- **The erosion of binary image B by structuring element S is denoted by B⊖S and is defined by**

$$B \Theta S = \{ b \mid b + s \in B \;\; \forall s \in S \}$$

Closing and Opening Operations

- The closing of binary image B by structuring element S is denoted by B • S and is defined by

$$B \bullet S = (B \oplus S)\Theta S$$

- ***Closing***: dilation followed by erosion with the same ***SE***

  – smooth by expansion, fills gaps and holes smaller than the ***SE***

- **The opening of binary image B by structuring element S is denoted B∘S and is defined by**

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

- ***Opening*: erosion followed by dilation with the same *SE***

  – filters out "positive" detail, shrinks the region

Contour Tracing (*border following* or *boundary following*)

- There are 2 kinds of boundary (or ***border***) pixels: ***4-border*** pixels and ***8-border*** pixels.
  - A black pixel is considered a ***4-border*** pixel if it shares an **edge** with at least one white pixel.
  - A black pixel is considered an ***8-border*** pixel if it shares an **edge or a vertex** with at least one white pixel
  - A 4-border pixel is also an 8-border pixel. An 8-border pixel may or may not be a 4-border pixel.

1. Square Tracing Algorithm

- Given a digital pattern i.e. a group of black pixels, on a background of white pixels i.e. a grid; locate a black pixel and declare it as your "**start**" pixel.
  - Locating a "**start**" pixel can be done in a number of ways
  - we'll start at the bottom left corner of the grid
  - scan each column of pixels from the bottom going upwards -starting from the leftmost column and proceeding to the right- until we encounter a black pixel. We'll declare that pixel as our "**start**" pixel.

- ***Algorithm***
- Imagine that you are a bug (ladybird) standing on the **start** pixel as in ***Figure 1***. In order to extract the contour of the pattern, you have to do the following:

1. **every time you find yourself standing on a black pixel, turn left, and**
2. **every time you find yourself standing on a white pixel, turn right,**
3. **until you encounter the start pixel again.**

The black pixels you walked over will be the contour of the pattern.

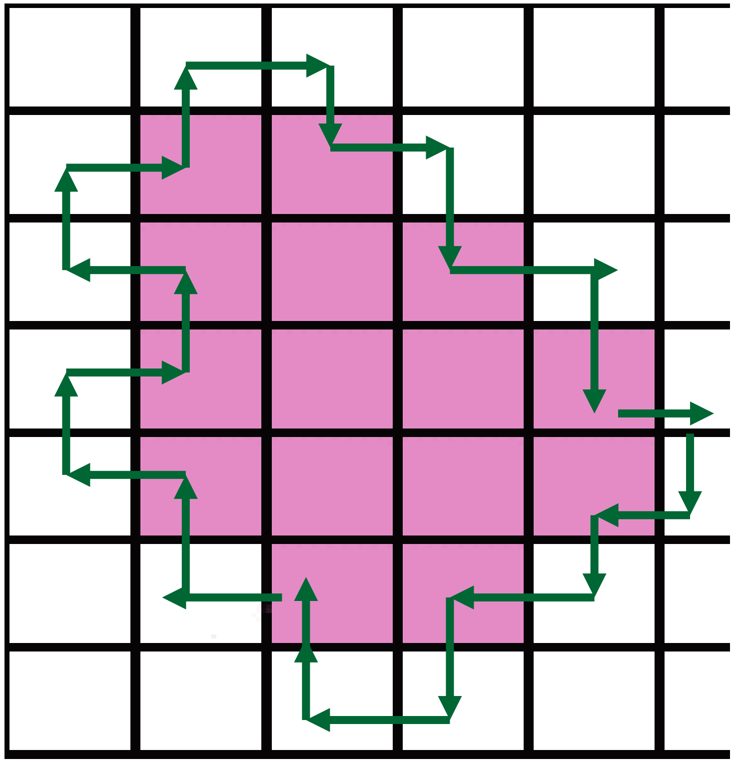
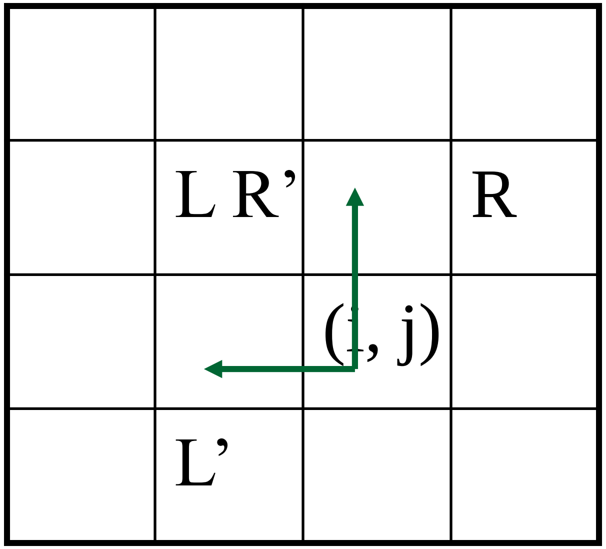


Figure 4



2. Moore neighborhood tracing algorithm

- **Moore Neighborhood**

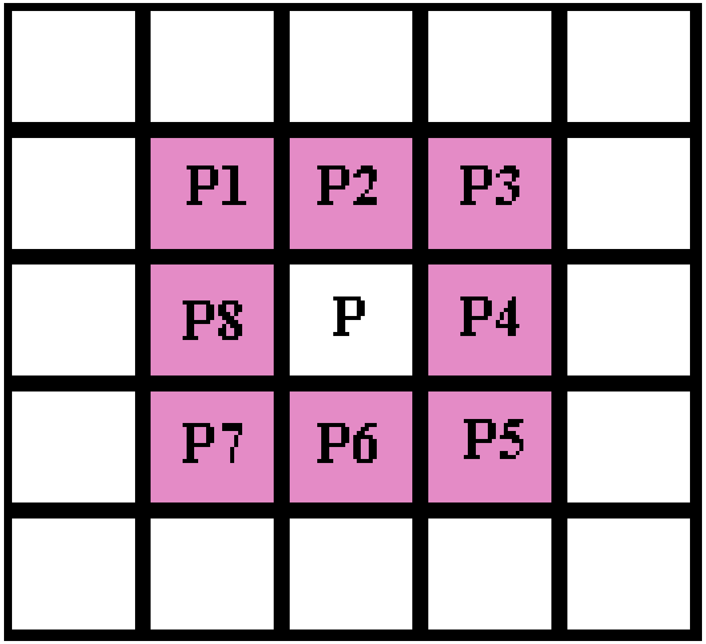


Figure 1

a. Given a digital pattern i.e. a group of black pixels, on a background of white pixels i.e. a grid
b. locate a black pixel and declare it as your "**start**" pixel. (we'll start at the bottom left corner of the grid.)
c. extract the contour by going around the pattern in a clockwise direction.

- every time you hit a black pixel, **P**, backtrack i.e. go back to the white pixel you were previously standing on,
- then, go **around** pixel **P** in a clockwise direction, visiting each pixel in its Moore neighborhood, until you hit a black pixel.
- The algorithm terminates when the start pixel is visited for a second time.

The black pixels you walked over will be the contour of the pattern.

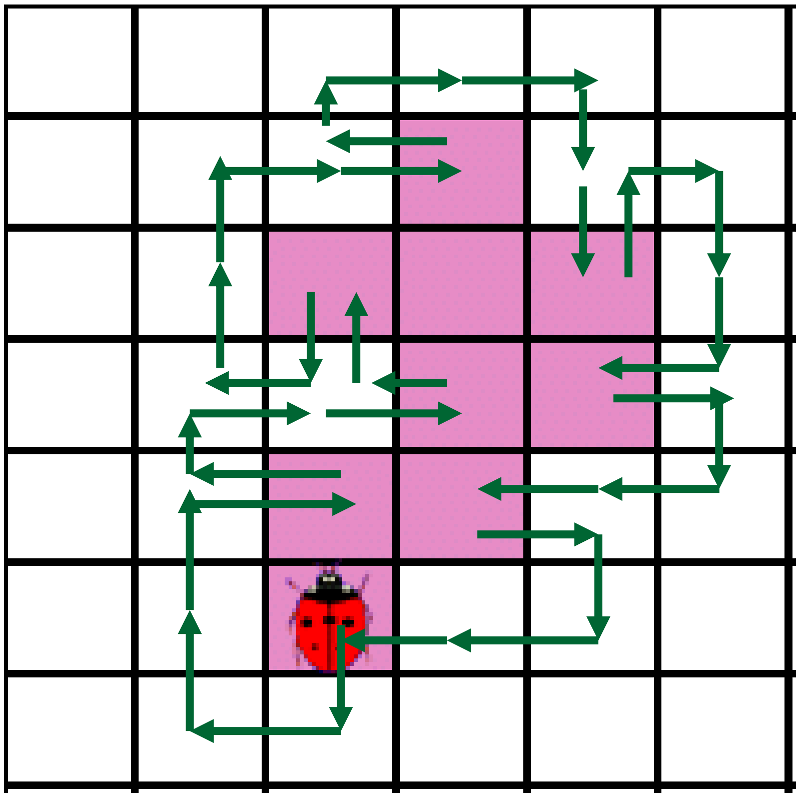


Figure 2

Thresholding Gray Level Images

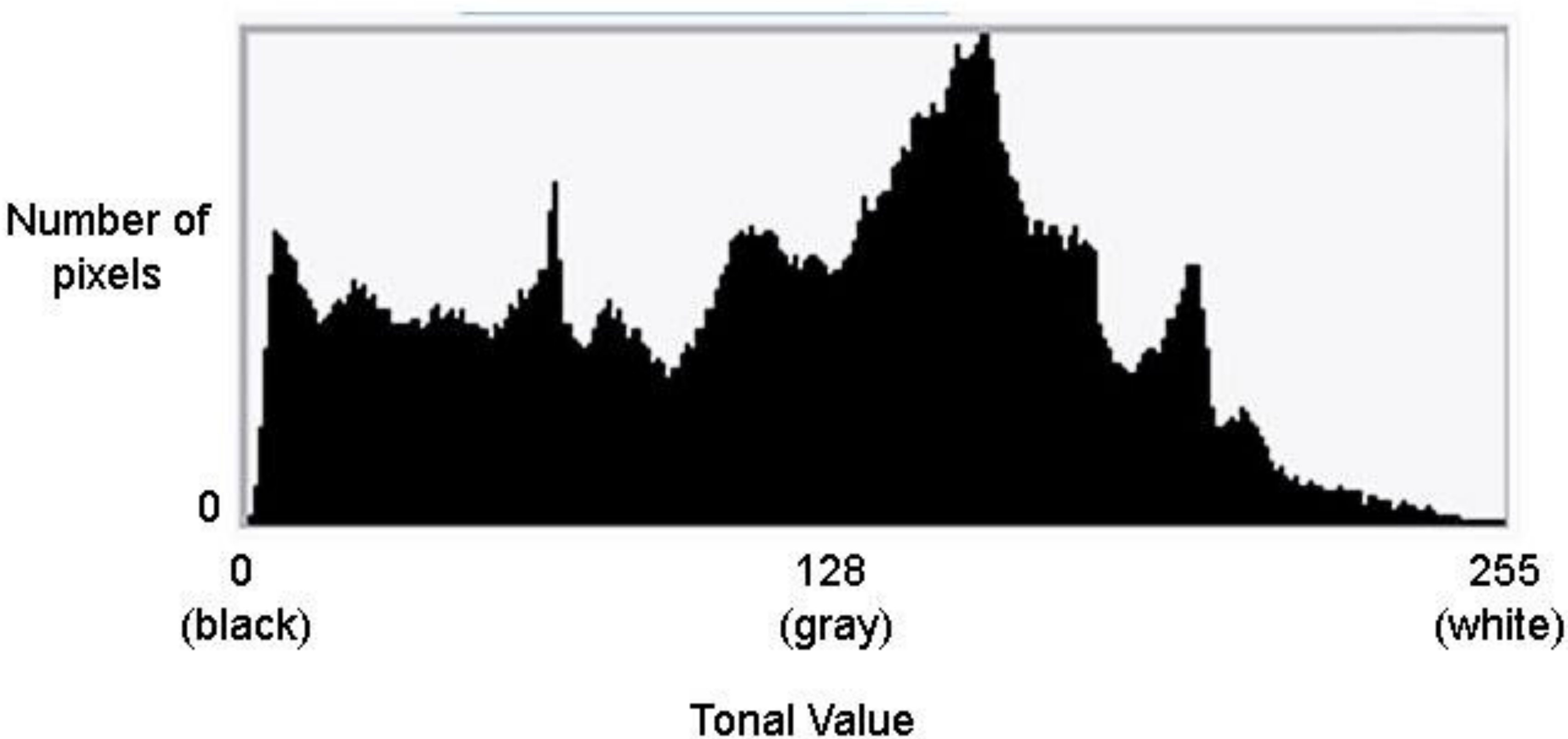
- **The simplest approach to segment an image is using thresholding.**

- **Single value thresholding can be given mathematically as follows:**

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

What is a histogram?

- A histogram is a "bar chart".



This bar chart has 256 bars, moved together so there is no space between the bars.

Basic Global Thresholding Algorithm

- The basic global threshold, T, is calculated

as follows:

1. Select an initial estimate for T (typically the average grey level in the image)
2. Segment the image using T to produce two groups of pixels: G_1 consisting of pixels with grey levels >T and G_2 consisting pixels with grey levels $\leq$ T
3. Compute the average grey levels of pixels in G_1 to give μ_1 and G_2 to give μ_2

4. Compute a new threshold value:

$$T = \frac{\mu_1 + \mu_2}{2}$$

5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit T_∞

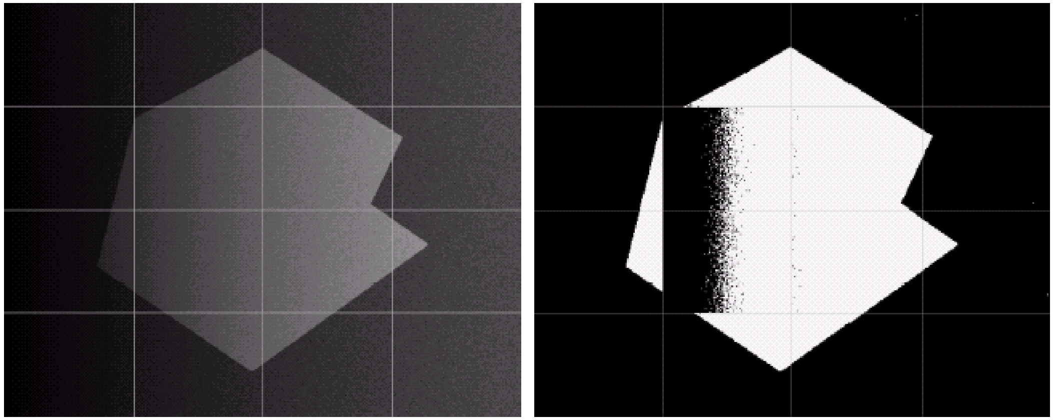
- **This algorithm works very well for finding thresholds when the histogram is suitable**

Basic Adaptive Thresholding

- An approach to handling situations in which single value thresholding will not work is to divide an image into sub images and threshold these individually
- Since the threshold for each pixel depends on its location within an image this technique is said to *adaptive*

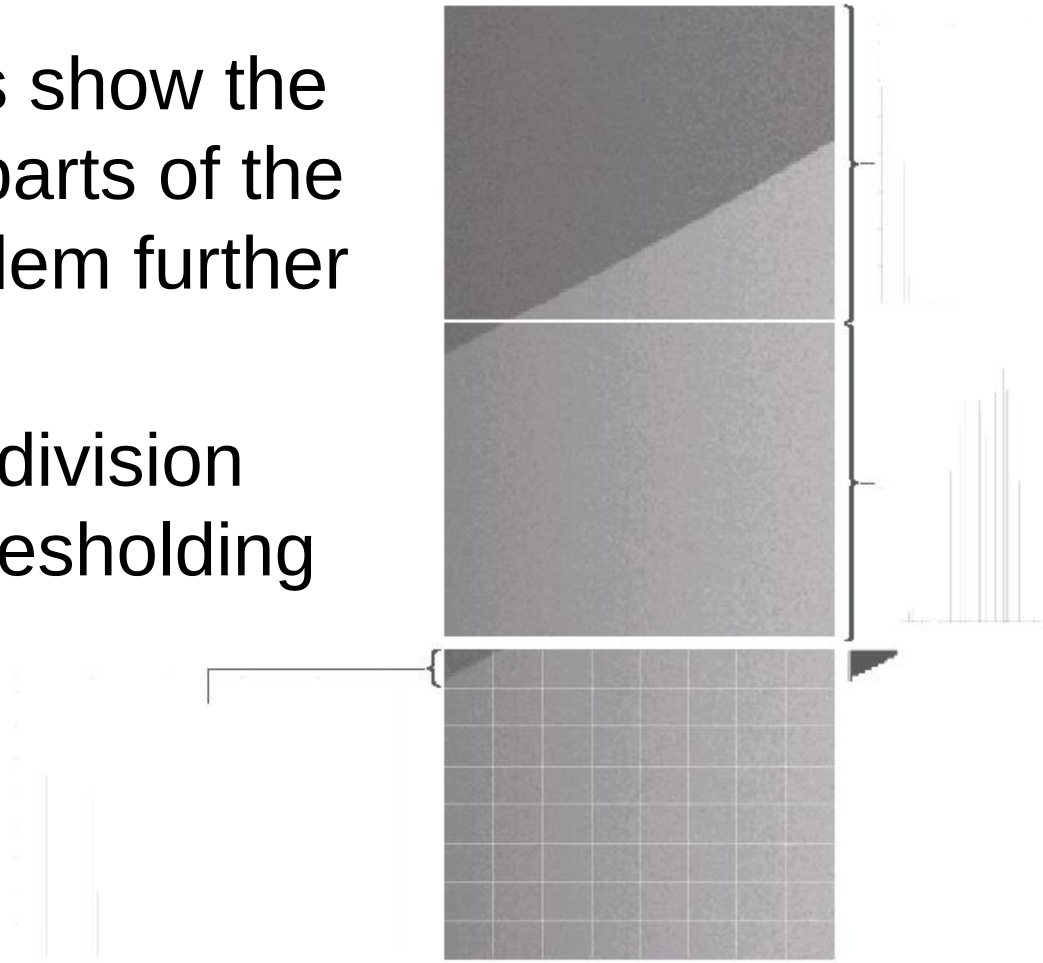
Basic Adaptive Thresholding Example

- The image below shows an example of using adaptive thresholding with the image shown previously



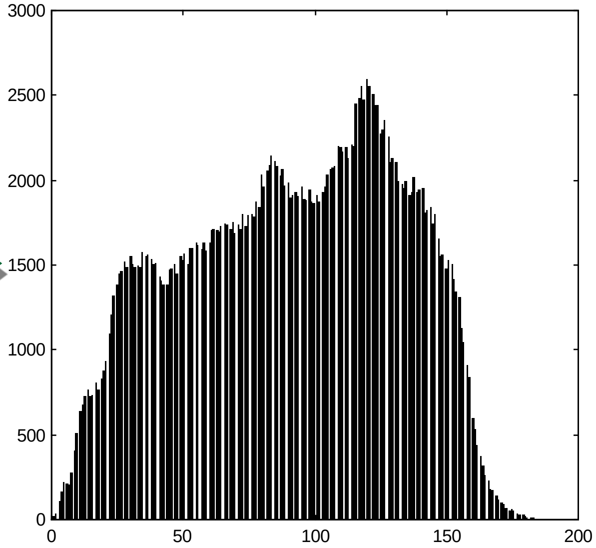
- As can be seen success is mixed
- But, we can further subdivide the troublesome sub images for more success

- These images show the troublesome parts of the previous problem further subdivided
- After this sub division successful thresholding can be achieved



Histogram based Enhancement

- Histogram of an image represents the relative frequency of occurrence of various gray levels in the image



MATLAB function >imhist(x)

Histogram Equalization

- ***Histogram equalization*** is an important method in histogram modification.
- Transform the intensity values so that the histogram of the output image approximately matches the flat (uniform) histogram
- Histogram equalization is often tried to enhance an image.

Normalised histogram function

- **The normalised histogram function is the histogram function divided by the total number of the pixels of the image:**

$$p(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n}$$

- It gives a measure of how likely is for a pixel to have a certain intensity. That is, it gives the probability of occurrence the intensity.

- The sum of the normalised histogram function over the range of all intensities is 1.

Example

- Consider a 5x5 image with integer intensities in the range between one and eight:

1	8	4	3	4
1	1	1	7	8
8	8	3	3	1
2	2	1	5	2
1	1	8	5	2

Normalised histogram function

$$h(r_1) = 8 \qquad\qquad p(r_1) = 8/25 = 0.32$$
$$h(r_2) = 4 \qquad\qquad p(r_2) = 4/25 = 0.16$$
$$h(r_3) = 3 \qquad\qquad p(r_3) = 3/25 = 0.12$$
$$h(r_4) = 2 \quad \Rightarrow \quad p(r_4) = 2/25 = 0.08$$
$$h(r_5) = 2 \qquad\qquad p(r_5) = 2/25 = 0.08$$
$$h(r_6) = 0 \qquad\qquad p(r_6) = 0/25 = 0.00$$
$$h(r_7) = 1 \qquad\qquad p(r_7) = 1/25 = 0.04$$
$$h(r_8) = 5 \qquad\qquad p(r_8) = 5/25 = 0.20$$

- **Histogram equalization**
  - find a map f(x) such that the histogram of the modified (equalized) image is flat (uniform).
  - Key motivation: cumulative probability function of a random variable approximates a uniform distribution

Suppose h(t) is the histogram

$$T(x) = \sum_{t=0}^{x} p(t)$$

Example

Normalised histogram function	**Intensity transformation function**
$p(r_1) = 0.32$	$T(r_1) = 0.32$
$p(r_2) = 0.16$	$T(r_2) = 0.32 + 0.16 = 0.48$
$p(r_3) = 0.12$	$T(r_3) = 0.32 + 0.16 + 0.12 = 0.60$
$p(r_4) = 0.08$	$T(r_4) = 0.32 + 0.16 + 0.12 + 0.08 = 0.68$
$p(r_5) = 0.08$	$T(r_5) = 0.76$
$p(r_6) = 0.00$	$T(r_6) = 0.76$
$p(r_7) = 0.04$	$T(r_7) = 0.80$
$p(r_8) = 0.20$	$T(r_8) = 1.00$

$$p(r_1) = 0.32 \quad T(r_1) = 0.32 \times 8 \to 3$$
$$p(r_2) = 0.16 \quad T(r_2) = 0.48 \times 8 \to 4$$
$$p(r_3) = 0.12 \quad T(r_3) = 0.60 \times 8 \to 5$$
$$p(r_4) = 0.08 \quad T(r_4) = 0.68 \times 8 \to 5$$
$$p(r_5) = 0.08 \quad T(r_5) = 0.76 \times 8 \to 6$$
$$p(r_6) = 0.00 \quad T(r_6) = 0.76 \times 8 \to 6$$
$$p(r_7) = 0.04 \quad T(r_7) = 0.80 \times 8 \to 6$$
$$p(r_8) = 0.20 \quad T(r_8) = 1.00 \times 8 \to 8$$

The 32% of the pixels have intensity r1. We expect them to cover 32% of the possible intensities.

The 48% of the pixels have intensity r2 or less. We expect them to cover 48% of the possible intensities.

The 60% of the pixels have intensity r3 or less. We expect them to cover 60% of the possible intensities.

…………………………………

Histogram equalisation algorithm I:

- **Let** $r_k, k = 1,2,\ldots,m$ **be the intensities of the image**
- **Let** $p(r_k)$ **be its normalised histogram function.**
- **The intensity transformation function for histogram equalisation is**

$$T(r_k) = \sum_{j=1}^{k} p(r_k)$$

- **That is, we add the values of the normalised histogram function from 1 to k to find where the intensity** r_k **will be mapped.**

- **Multiply cumulative values by the maximum gray-level value and round the results to obtain** $r_k^{'}$
- **Map the original gay-level value to the resulting value**

Algorithm II

- The two requirements on the operator are that

  (a) the output image should use all available gray levels

  This requirement means that the target output image uses all gray values *z = z1, z = z2, … z = zn*

  (b) the output image has approximately the same number of pixels of each gray level.

  This requirement indicates each gray level *zk* is used approximately $q = (R \times C)/n$ times, where *R, C* are the number of rows and columns of the image.

- Compute the average number of pixels per gray level.
- Starting from the lowest gray-level band, accumulate the number of pixels until the sum is closest to the average. All of these pixels are then rescaled to the new reconstruction levels.
- If an old gray-level band is to be divided into several new bands, either do it randomly or adopt a rational strategy----one being to distribute them by region.