# Module - 4

## Image Segmentation

↳ Divides an image into different regions so that each round we presents a meaningfull part of the img.

## Types Of Image Segmentation

### (1) Discontinuity - Based Segmentation

- Point detection :- Detects isolated points
- Line detection :- Detects straight line
- Edge detection :- Detects Objects boundaries where there is sharp change in color or brightness.

### (2) Similarity based detection :-

↳ Here we group pixels in some way.

- Thresholding
- Region growing
- Region splitting & merging

} Techniques

# → Detection Of Discontinuities

↳ The most common way to look for discontinuities is to scan a small mask over an image

$$R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

Result ↑

$w$ = weights from filter
$z$ = pixel values from image region

(a) Point discontinuous :- Isolated bright or dark spots that stand out.

mask example

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

↳ compares center pixel (8x) its value to its 8 neighbours (-1 each)

(b) Line Discontinuities :-

↳ Identify line of 1 pixel width in special orientations

① Horizontal ② +45° ③ Vertical ④ -45°

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Only line in target directions become visible when pixel align to form a line the result in large.

# what are derivaties in Images

① Small derivative : smooth region

② Large derivative : sharp change (edge or boundary)

→ First Order derivatives

↳ tells how fast the intensity is changing in images.

If the intensity jump suddenly between its neighbouring pixels, its probably an edge.

$\frac{\partial f}{\partial x}$ - change in intensity in horizontal direction

$\frac{\partial f}{\partial y}$ - change in intensity in vertical direction

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$ → together we define a gradient vector.

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

↓

magnitude :- tells u the strength of edge

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big| \frac{\partial f}{\partial x}\right)$$

↓

direction - tells you which direction the edge is going.

Prewitt mask for detecting diagonal edges

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Sobel mask for detecting diagonal edges

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

How do these masks work

- Place these mask over each pixel in an image
- Multiple each of the 9 surrounding values by the values in the mask.
- Add up the results you get gradient in x or y direction.
- Do this for both the masks (x & y)
- Combine the result to get gradient magnitude.

→ Difference between prewitt & sobel mask

| Prewitt | sobel |
| --- | --- |
| • more sensitive (Less smoothening) | • Less sensitive (more smoothening) |
| • slightly less accurate | • more Accurate due to weight |
| • Simpler averaging & differentiation | • Slightly more complex |
| • Slightly faster | • Slightly slower |
| • Good for simple & small images | • Better for natural images or raw data. |

## → Second Order Derivatives

→ tells us how fast the gradient itself is changing.

Detects → Rapid changes in the slope
→ Places where the intensity increases & decreases

### Common Operator (Laplacian)

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

→ Mask $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ or $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

→ Second Order derivative are sensitive, so we smooth the image first.

→ Gaussian smoothening (Step-1)

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

} Reduces high frequency noise before applying an edge detector

→ Laplacian Operator (step-2)

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

} It looks for edges & detects places where the sign of the second derivative changes.

## Step-3 :- Laplacian Of Gaussian

↳ apply this to create a LOG Kernel.

$$LoG(x,y) = \nabla^2 G(x,y)$$

$$= \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right) \times \exp\left(\frac{-x^2+y^2}{2\sigma^2}\right)$$

Defines the curvature shape (-ve) near center, +ve around sides

Controls the spread of the effect - smoother for larger $\sigma$

Why use LoG?

1. Gaussian smoothens out the image
2. Laplacian finds out the edges.

# Edge linking & Boundary detection

↳ After predicting edges through different methods, you don't always get a perfect connected edge, you may get broken or disjoint edges

→ Local Processing

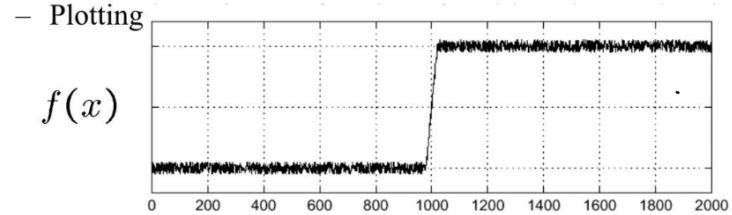
↳ use neighbouring pixels to decide whether edge should be connected or not.

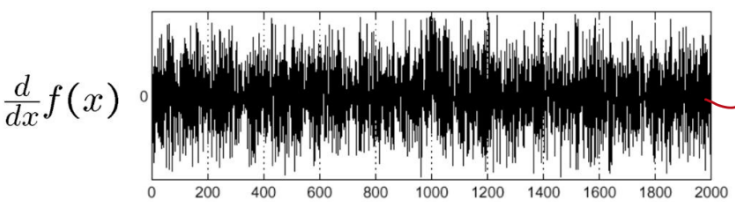
Key Concepts → Edge Magnitude
→ Edge Direction

- We check if two adjacent edge pixels have same magnitude & direction

→ Effect of Noise

- Consider a single row or column of the image
  - Plotting




Where is the edge??

There is no noise in the first image so the spike is detected

Here when we take the first derivative i.e derivative of those intensities there might be some small random changes which resulted in noise & the spike was hidden.

→ You can use first derivative to find intensity changes in order to find the edge

→ But noise can create fake changes so derivative becomes noisy

→ Canny Edge Operator

① Smooth image I with 2D-Gaussian $G_\sigma * I$

② Find local edge normal direction $\bar{n} = \frac{\nabla (G_\sigma * I)}{|\nabla (G_\sigma * I)|}$ for each pixel

③ Locate edge by finding zero-crossings along edge normal directions $\frac{\partial^2 (G_\sigma * I)}{\partial \bar{n}^2} = 0$

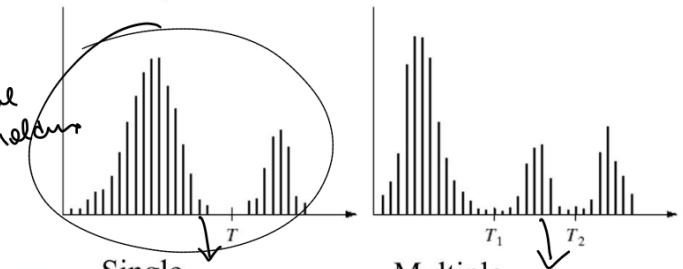
→ Thresholding

↳ methods to seperate Objects from BG based on pixel intensity.

- Assumption: the range of intensity levels covered by objects of interest is different from the background.

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

Bimodal Peaks for Global Thresholding




Single Multiple

f(x,y)- Intensity
T - Threshold value

g(x,y)= Output (1 = white, 0 = black)

2 spikes
① Background
② Object

3 spikes
① BG
② Object1
③ Object2

## → Basic Adaptive Thresholding

↳ Calculates different thresholds for each pixel based on its local neighbourhood, instead of using global threshold for the whole image.

Global Thresholding works only if:
- The lightening is even
- BG & Object have clear intesity seperation

Basic Adaptive Thresholding solves this problem

## → Optimal Global & Adaptive Thresholding

↳ This method treats pixel values as probability density functions.
The goal is to reduce the probability of misclassifying pixels as either Objects or BGs.

## → Region Based Segmentation

↳ Dividing an image into connected regions where pixels within the region are similar to each other.

- Here each region must be uniform
- Connectivity of pixels within the region is v imp.

## → Region Growing

→ Procedure that groups pixels or sub-region into larger regions.

- It is a technique that starts from a seed pixel & grows the region by adding neighbouring pixels that are similar to the seed.

- Region growing based technique are better than the edge-based techniques in noisy environment.

## → Region Splitting & Merging

→ Top-down & bottom-up segmentation method that breaks the images into smaller parts & then combines them when they meet a homogeneity condition.

## → Region Splitting

- Start with an entire image as one big-region
- Check if its uniform, if not split in 4 equal sub-region
- Repeat recursively until all regions are uniform.

## → Region merging

- Look for neighbouring regions, if they are similar enough merge them.
- Continue until no-more merging is possible.

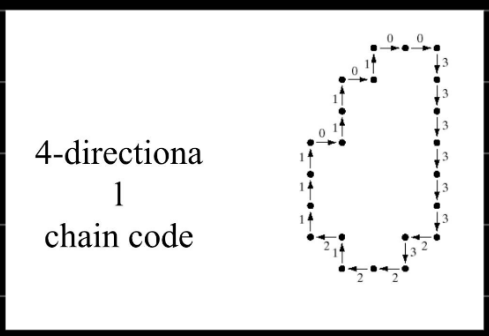
## → Boundary - Descriptors

↳ It is a way to represent the shape or outline of the image

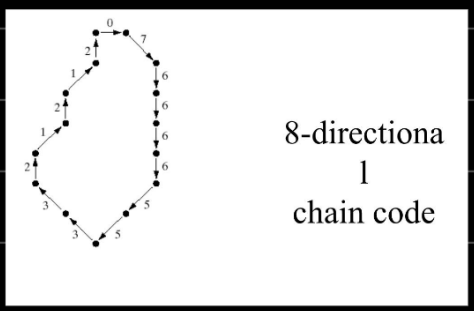
## → Chain code

↳ It is a way to encode the boundary of a shape as a sequence of directions

## Direction Encoding




Can move only 4 ways




Can move 8 ways (including diagonal)

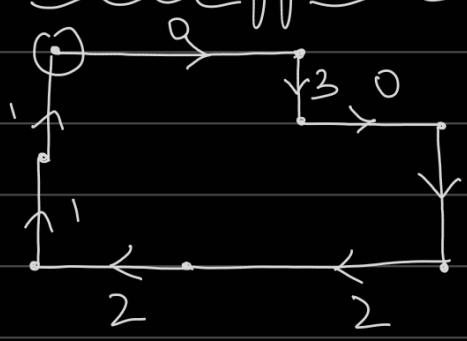
**Limitations** :- • Starting point matters → same shape gives different codes if traced from diff. positions

• Rotating the Object changes the code

• Sensitive to Noise.

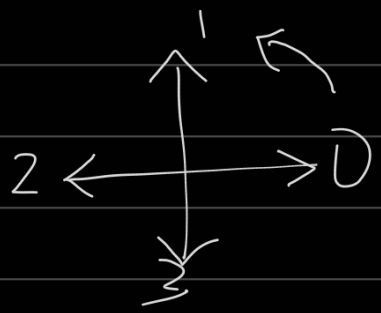
## → Improvements

### ① First difference of chain-code




→ 0303221 1

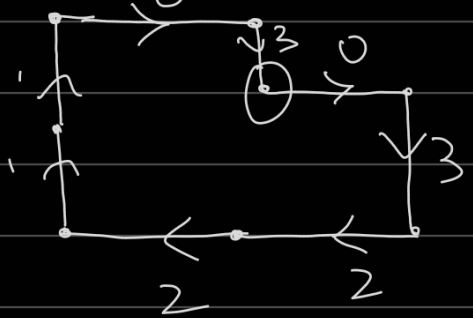
↳ chain code




First diff: 3133 0303

That's we normalize

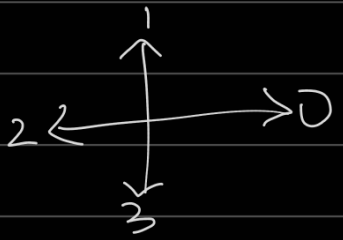
Normalize 03033133 → same




03221103

33030331

03033133




## → Image Moments

**moment** :- Contain particular weighted avg.

**Image moment** :- Particular weighted avg. of pixel's intensities

They are a set of statistical parameters to measure the distribution of pixels & their intensities

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

↳ Intensity at location x,y.

→ For binary image the zeroth order moment corresponds to area

$$M_{00} = \sum_x \sum_y I(x,y)$$

(by counting all non zero pixels)

→ Centroid :- Arithmatic mean position of all the points

$$\text{Centroid } \{\bar{x}, \bar{y}\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\}$$

Eg

| x / y | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 |

$$M_{00} = \sum_x \sum_y \bar{x}\bar{y}\, I(x,y)$$

$$= \sum_x \sum_y I(x,y)$$

$$= 8$$

$$M_{10} = \sum_x \sum_y x I(x,y) = 1(0+0+0+0) + 2(1+1+1+1) + 3(1+1+1+1) + 4(0+0+0+0) = 20$$

$$M_{01} = 1(0+1+1+0) + 2(0+1+1+0) + 3(0+1+1+0) + 4(0+1+1+0)$$

$$= 1(2) + 2(2) + 3(2) + 4(2) = 2+4+6+8 = 20$$

$$\bar{X} = \frac{M_{10}}{M_{00}} = \frac{20}{8} \qquad \bar{Y} = \frac{M_{01}}{M_{00}} = \frac{20}{8}$$