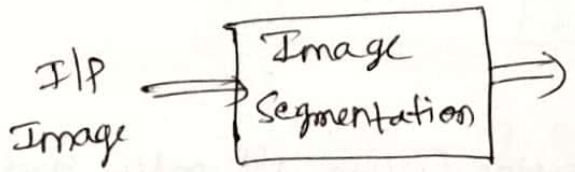
Module-5

Image Segmentation, Representation & Description

* Fundamentals
* Point, line & Edge detection
  - → Detection of isolated points
  - → Line detection
  - → Edge models
  - → Basic Edge detection [using 1st order gradient, ∇f]
  - → The Marr-Hildreth edge detector Algorithm
  - → The Canny Edge detector Algorithm
  - → Edge linking & Boundary detection
    - ↳ local processing
    - ↳ Regional processing
    - ↳ Global processing [Hough transform]
* Region-based Segmentation
  - → Region Growing
  - → Region Splitting & merging
* Thresholding
  - → Global thresholding
  - → Otsu's method
  - → Image Smoothing to improve Global thresholding
  - → Edges to improve Global thresholding
  - → Multiple thresholds, Variable & Multivariable Thresholding
* Segmentation using Morphological Watersheds
* Representation & Description.

Image Segmentation:

* Segmentation is one of the fundamental step in image processing where inputs are images but the outputs are attributes extracted from those images.



*Defn: "Segmentation is a process of Subdividing an image into its Constituent regions (or) objects". Then we analyze Each of these Constituents to Extract Some information, so that those informations are useful for high level machine vision applications.

* At which level this Subdivision Should Stop? ⇒ depends on the application (or) the problem being solved.

*For Example: If we are interested in detecting the movement of Vehicles on a road: Given an aerial image taken from the Satellite (or) helicopter, first road segment is done which seperates road from other regions like residential areas, water bodies, agricultural lands etc. Then further subdivision of the road for motion analysis.

*Here, we need not to divide agricultural lands, water bodies etc. So, level of Subdivision (or) Segmentation is application dependent.

* **Notation:** If R represent the Entire Spatial region occupied by an image then we view image Segmentation as a process that Partitions R into n Subregions, $R_1, R_2, \ldots, R_n$, Such that $\bigcup_{i=1}^{n} R_i = R$

↳ union of all subregions - $(R_1, R_2 \cdots R_n)$ will get back region-R.

* Image Segmentation approaches are mainly of 2 types
  - ① Discontinuity based approaches
  - ② Similarity based approaches.

* In the first method, the partition is carried out based on some abrupt changes in intensity levels in an image (or) in gray levels of image. using this discontinuity approach we usually find isolated points, lines & edges.

* In Similarity based approach, we try to group those pixels in an image which are similar in some sense Ex for this type of approaches are Thresholding, Region growing, Region Splitting & merging approach.

* First we will look into Discontinuity based approaches.

Note: * R_i is a Connected set, $i = 1, 2 \ldots n$

* $R_i \cap R_j = \phi$ for $\forall i \& j$, $i \neq j$

↳ Regions must be disjoint.

* Point, line, & Edge detection :

* The three types of image features in which we are intrested are isolated points, lines & edges.
* Edge pixels are pixels at which the intensity of an image function changes abruptly, and Edges are sets of connected edge pixels.
* An isolated point may be viewed as a line whose length & width are equal to one pixel.
* For detection of these 3 features the kind of approach that we take is => use of Mask wrt 1st order derivative & 2nd ...

* Detection of Isolated points :

* This point detection is based on second derivative.
* i.e using Laplacian based mask ie.

wkt from Laplacian $\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ — (1)

where $\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$

& $\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$

The Laplacian eqn (1) is then

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$ → (2)

result in mask =>

0	1	0
1	-4	1
0	1	0

known as Laplacian mask.

* we can extend above eqn (2) to include diagonal elements. that time mask becomes.

1	1	1
1	-8	1
1	1	1

=> This mask is used for point detection

* Yes, the mask used for point detection is ⇒

1	1	1
1	-8	1
1	1	1

⟹ Subjected to the Equation

$$g(x,y) = \begin{cases} 1 & \text{if } |R(x,y)| \geq T \\ 0 & \text{otherwise} \end{cases} \quad \text{(3)}$$

* This Laplacian mask is moved over the image and we say that a point has been detected at the location (x,y) on which the mask is centered if the absolute value of the response of the mask i.e. $|R(x,y)|$ at that point Exceeds a Specified threshold(T).

* Such points are labeled 1 in the o/p image & all others are labeled 0, thus producing a binary image.

* i.e. o/p image $g(x,y)$ is obtained by using the expression shown in (3). T is a non negative threshold, &

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9$$
$$= \sum_{k=1}^{9} w_k z_k$$

W ⇒ mask Co-efficients
Z ⇒ Image pixels.

Note:
* The idea is that the intensity of an isolated point will be quite different from its surroundings & thus will be Easily detectable by this type mask.

*

* Line Detection :

* Wkt for line detection, 2nd derivatives results in a stronger response & produce thinner lines than first derivatives.

* Thus, we can use 2nd derivative based Laplacian mask i.e.

1	1	1
1	-8	1
1	1	1

⇒ but this particular laplacian detector is isotropic i.e. its response is independent of direction.

* often, we are intrested in detecting lines in specified directions i.e. line detection in Horizontal direction (or) Vertical direction (or) +45° angle direction (or) -45° etc.

* For this purpose we make use of following masks for line detection in one particular direction. they are

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

2	-1	-1
-1	2	-1
-1	-1	2

+45°

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

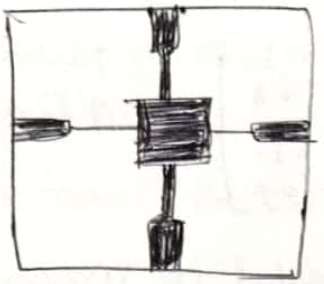
-1	-1	2
-1	2	-1
2	-1	-1

-45°

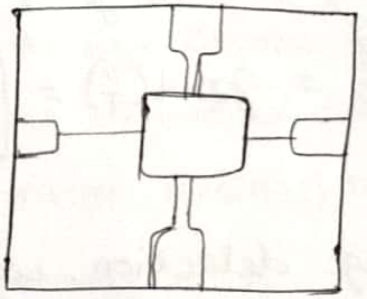
* Let R_1, R_2, R_3 & R_4 be response of these masks from left to right. where response $R_s = \sum_{k=1}^{i} W_k R_k$

* If at a given point in the image, $|R_k| > |R_j|$ for $j \neq k$, that point is said to be more likely associated with a line in the direction of mask k.

* In other words, if we are interested in detecting all the lines in an image, then we simply run all masks through the image & threshold the absolute value of the result.



original image



* Absolute value of the Laplacian

← o/p after line detection using Laplacian mask.

* Edge Dection:

* Edge Models: In pactice we come across with different types of Edges, most common are
  - Step Edge
  - ramp Edge
  - roof Edge

* It is not unusual to find images that contain all three types of Edges. ⇒

* Basically Edge detection is based on segmenting images based on abrupt (local) changes in intensity.

* There are 3- fundamental steps performed in Edge detection:
  1) Image Smoothing for noise reduction.
  2) Detection of Edge points.
  3) Edge localization. ⇒ In this step we select only points that are true members of the set of points comprising an Edge.

* Detecting changes in intensity for the purpose of finding Edges can be accomplished using first or second-order derivatives.

* First, we discuss use of first-order derivative in Edge detection.

* First order derivative in Edge detection: [Image gradient, ∇f]

* wkt 1st order derivative computed using gradiant operator.
* When we have an image say $f(x,y)$, we can define the gradient of this image $f(x,y)$ as

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$ → Partial derivative of 'f' along x-direction

* For Edge detection, we are interested in magnitude of gradient i.e.

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2}$$ ⇒ Taking square, square root, Computational Complex.

So, we will approximate above eqn as

$$\boxed{mag(\nabla f) = |g_x| + |g_y|}$$

* The above magnitude of gradient gives information about Strength of Edge at (x,y), but direction of Edge is still unknown.
* So, direction of ∇f is

$$\boxed{\alpha(x,y) = \tan^{-1}\left(\frac{g_x}{g_y}\right)}$$ ⇒ gives direction of Edge.

* The following masks are derived from 1st order derivative gradient to detect the edges,

Roberts Edge detector

-1	0
0	1

0	-1
1	0

Prewitt Edge detector

-1	-1	-1
0	0	0
1	1	1

⇒ To detect Horizontal Edges ←

-1	0	1
-1	0	1
-1	0	1

⇒ To detect Vertical Edges ←

Sobel Edge detector

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Note: The above masks derived from basic equations of gradient. Discussed in Module-2, under the topic of "Image sharpening".

* More advanced Techniques for Edge Detection:

* The masks which we discussed till now for edge detection have no provisions being made for edge characteristics & noise content. So, we discuss more advanced techniques which account factors such as image noise & nature of edges.

* The Marr-Hildreth Edge detector:

* This edge detector techniaue suggest that an operator used for edge detection should have two salient features ie.

i) It should be a differential operator capable of computing first & second derivative at every point in the image.

ii) It should be capable of being "tuned" to act at any desired scale.

* According to Marr-Hildreth, the most satisfactory operator fulfilling these conditions is the filter $\nabla^2 G$.

* In $\nabla^2 G \Rightarrow \nabla^2$ is a Laplacian operator, $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$

& G is the 2-D Gaussian function.

* Wkt the Gaussian fun G on 2-D image $f(x,y)$ is given by,

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

where σ is the std. deviation.

* Now, to find an expression for $\nabla^2 G$, we perform following differentiations;

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

$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

After partial differentiation,

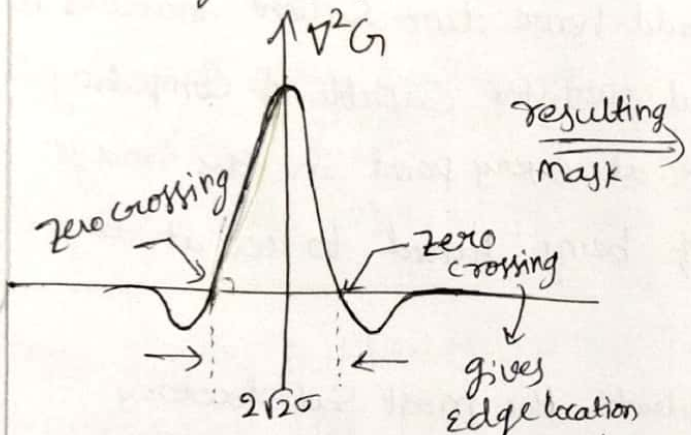
$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

* collecting terms gives the final Expression:

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \Rightarrow LoG$$

This Expression is called the Laplacian of a Gaussian (LoG).

* The Cross-Sectional View of this LoG operator & its mask given as,



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

⇒ The co-efficients must sum to zero.

* a positive central term surrounded by a adjacent, negative region whose value increases as a function of distance from origin & a zero out region.
* Size of mask can be varied by -Scaling.

* This operator $\nabla^2 G$ has 2 part. one is Gaussian part to remove noise & ∇^2 Laplacian part to detect edge.

* The Marr-Hildreth algorithm consists of convolving the LoG filter with an I/p image, $f(x,y)$ ie.

$$g(x,y) = \left[\nabla^2 G(x,y)\right] * f(x,y)$$

and then finding the zero crossings of $g(x,y)$ to locate Edges of $f(x,y)$

* Since these are linear processes, the above eqn can be written as

$$g(x,y) = \nabla^2 \left[G(x,y) * f(x,y)\right]$$

* i.e. $g(x,y) = \nabla^2[G(x,y) * f(x,y)]$

The above eqn indicates that we can smooth the image first with the Gaussian filter $G(x,y)$ & then compute the Laplacian of the result with Laplacian operator ∇^2. The above 2 eqn's gives identical result.

* The Marr-Hildreth Edge-detection algorithm may be summarized as follows: $g(x,y) = \nabla^2[G(x,y) * f(x,y)]$

1) Filter the input image with an nxn Gaussian lowpass filter i.e. $G(x,y)$. $\Rightarrow \{G(x,y) * f(x,y)\}$
2) Compute the Laplacian of the image resulting from Step1. ∇^2[Step1] i.e. $\nabla^2\{G(x,y) * f(x,y)\}$.
3) Find the zero crossings of the image from Step2.

Note: Marr & Hildreth noted that it is possible to approximate the LOG filter by a difference of Gaussians (DoG):

$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} \cdot e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} \cdot e^{-\frac{x^2+y^2}{2\sigma_2^2}} \quad \text{with } \sigma_1 > \sigma_2$$

* The Canny Edge detector:

* The Performance of the Canny Edge detector algorithm is superior compared to the edge detectors discussed thus far but at the cost of more complexity.
* Canny's approach is based on three basic objectives:
  i) low error rate
  ii) Edge points should be well localized
  iii) Single edge point response.

* To satisfy the above objective, the Canny algorithm consists of the following basic steps:

1) Smooth the input image with a Gaussian filter.
2) Compute the gradient magnitude and angle images.
3) Apply nonmaxima suppression to the gradient - magnitude image.
4) Use double thresholding and connectivity analysis to detect & link edges.

Step1: Let $f(x,y)$ denote the I/p image & $G_1(x,y)$ denote the Gaussian function i.e.

$$G_1(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

* we form the smoothed image, $f_s(x,y)$ by convolving G_1 & f:

$$f_s(x,y) = G_1(x,y) * f(x,y)$$

Step2: Step1 is followed by computing the gradient magnitude & direction(angle), as

$$M(x,y) = \sqrt{g_x^2 + g_y^2}$$

& $$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

where $g_x = \frac{\partial f_s}{\partial x}$ & $g_y = \frac{\partial f_s}{\partial y}$

* Prewitt masks (or) Sobel masks can be used to obtain g_x & g_y.

Step3 & 4:

* After performing Step1 & 2, the magnitude image $M(x,y)$ typically contains wide ridges around local maxima.
* To thin these ridges, one approach is to use - non-maxima suppression with hysteresis thresholding.
* First, a nonmaxima-suppressed image is $g_N(x,y)$ is obtained, which is then subjected to high thresholding i.e. $g_{NH}(x,y)$ & low thresholding $g_{NL}(x,y)$. Then difference is performed i.e. $\boxed{g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)}$

⇒ The non zero pixels in $g_{NH}(x,y)$ are ⇒ "strong" edge pixels.

⇒ The non zero pixels in $g_{NL}(x,y)$ are ⇒ "weak" edge pixels.

* After thresholding all strong pixels in $g_{NH}(x,y)$ are assumed to be valid edge pixels & are so marked immediately.

* Edge linking & Boundary Detection:

* Edge detection typically is followed by linking algorithms - designed to assemble edge pixels into meaningful edges and/or boundary regions.
* Here we study 3-fundamental approaches to edge linking ie.
  i) Local processing
  ii) Regional processing
  iii) Global processing (using the Hough transform)

P.T.O

1) Local Processing :

* One of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood about every point (x,y) that has been declared an edge point by one of the techniques discussed in previous sections.
* All points that are similar in some sense are linked.
* The two principal properties used for establishing similarity criteria of edge pixels in this approach are
  1) the strength (magnitude), $M(x,y) = \sqrt{g_x^2 + g_y^2}$
  2) the direction (angle) of gradient vector, $\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$
* Let S_{xy} denote the set of co-ordinates of a neighbor-hood centered at point (x,y) in an image.
* An edge pixel with co-ordinates (s,t) in S_{xy} is similar in magnitude to the pixel at (x,y) if

$$|M(s,t) - M(x,y)| \leq E$$

where E is a positive threshold.

* An edge pixel with co-ordinates (s,t) in S_{xy} has an angle similar to the pixel at (x,y) if

$$|\alpha(s,t) - \alpha(x,y)| \leq A$$

where A is positive angle threshold.

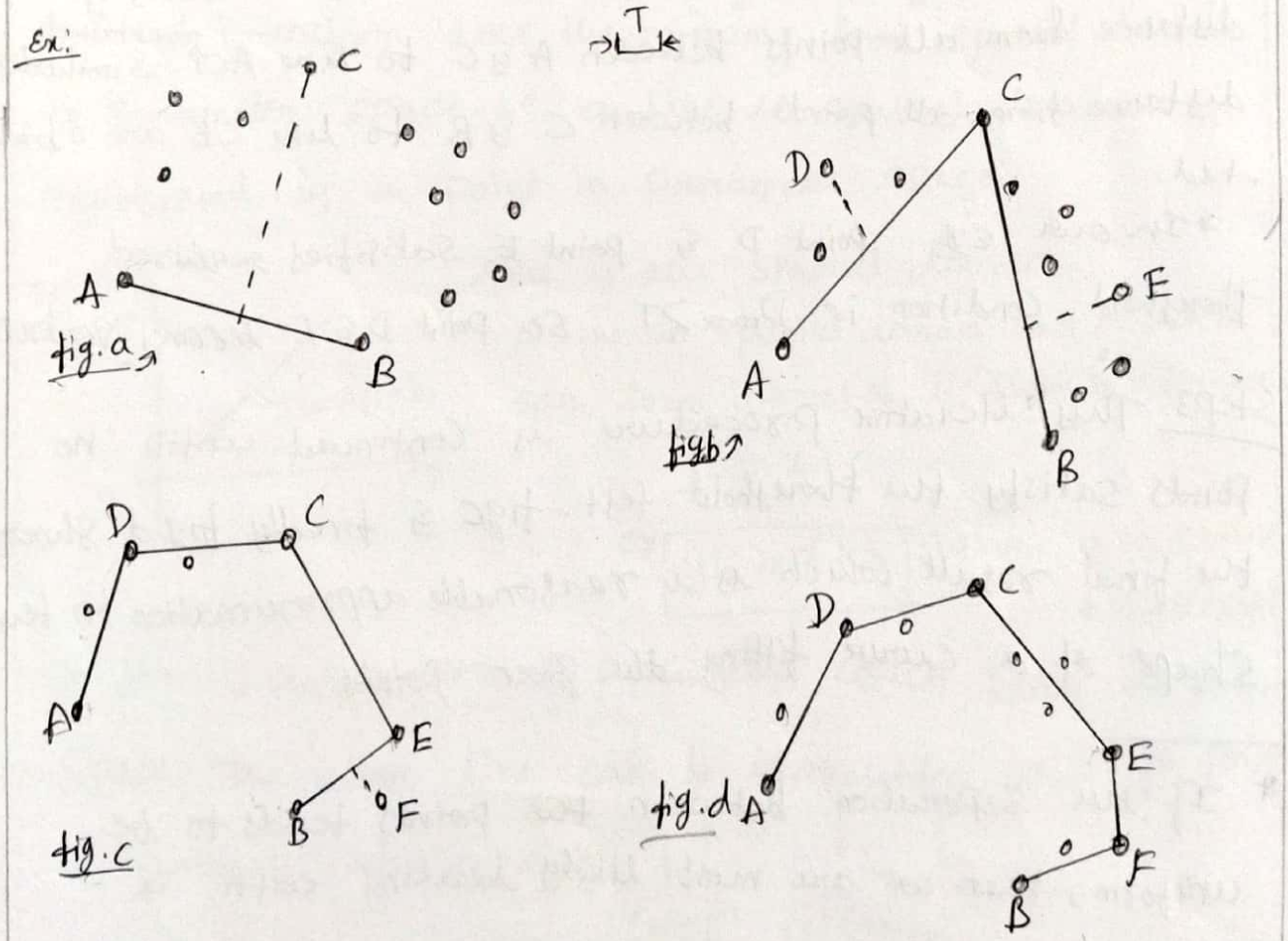
* A pixel with co-ordinates (s,t) in S_{xy} is linked to the pixel at (x,y) if both magnitude & directeria are satisfied. This process is repeated at every location in the image.

ii) Regional Processing:

* Often, the location of regions of interest in an image are known (or) can be determined.
* One approach to this type of processing is functional approximation, where we fit a 2-D curve to the known points.
* Polygonal approximations are particularly attractive because they can capture the essential shape features of a region while keeping the representation of the boundary relatively simple.

* Algorithm with Example:

Ex:



Step1: fig a. shows a set of points representing an open curve in which the end points have been labeled A & B, & these points are by definition Vertices of the Polygon.

Step1: we begin by computing the parameters of a straight line passing through A & B. Then, we compute the perpendicular distance from (Dmax) from all other points in the curve to this line & select the point that yielded the maximum distance (Dmax).

* If this distance Exceeds a specified threshold, T

i.e. $D_{max} > T$, the corresponding point is declared as Vertex. In our Ex. it is the point C as shown in fig a.

Step2: Lines from A to C & C to B are then Established, & distance from all points between A & C to line AC are obtained. Similarly distance from all points between C & B to line CB are obtained.

* In our Ex. Point D & Point E satisfies required threshold condition i.e. $D_{max} > T$, So Point D & E becomes Vertex Now.

Step3: This iterative procedure is Continued untill no Points satisfy the threshold test. fig c & finally fig. d shows the final result which is a reasonable approximation to the Shape of a curve fitting the given points.

---

* If the Seperation between the points tends to be uniform, then we are most likely dealing with a closed curve.

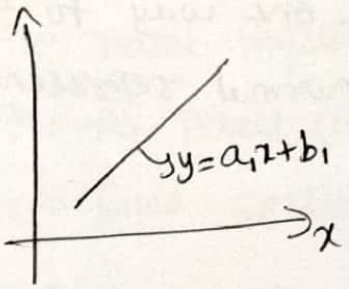
iii) Global Processing [using the Hough transform]:

* In practice, we have to work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be. *) In such situations we use global properties i.e. we develop an approach based on whether sets of pixels lie on curves of a specified shape. Once detected then these curves form the edges or region boundries of interest.

* **Hough Transform**: A way of finding edge points in an image that lie along a straight line.

* Hough transform does the mapping from spatial domain to parameter space. i.e. a line in spatial domain represented by a point in parameter space.

Ex:



* xy is the spatial domain.
* A line in spatial domain can be put in eqn. form interms of its slope & intercept as

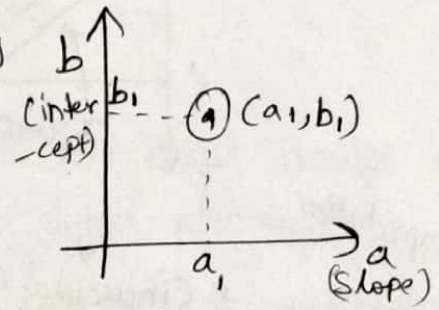
$$y = mx + c$$

or $\boxed{y = ax + b}$ where a ⇒ slope
b ⇒ intercept.

* Here, a & b are 2 parameters which forms ab parameter Space. The above line can be represented by single point in parameter ab-space as

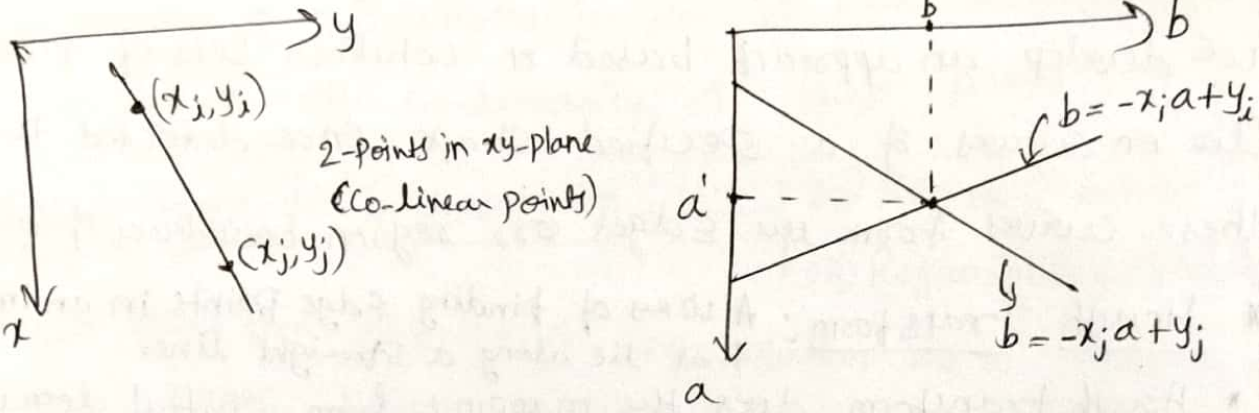
$$y = a_1x + b_1$$

$$\Rightarrow b_1 = -a_1x + y$$



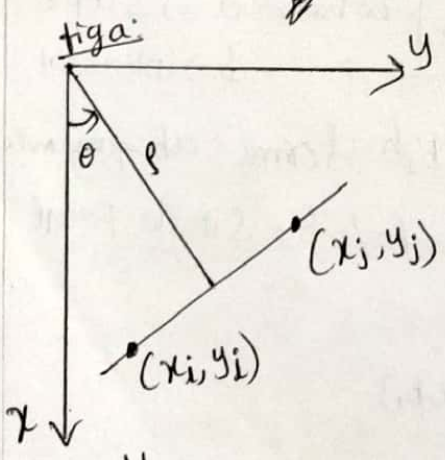
* Vice-Versa is true ie. a point in Spatial domain by line in parameter-space.

* If we have 2 - points lying on same straight line in xy-plane i.e. 2 - colinear points (x_i, y_i) & (x_j, y_j) then these points can be represented by 2 straight lines in parameter space - ab, passing through single point (a', b') as shown in fig.



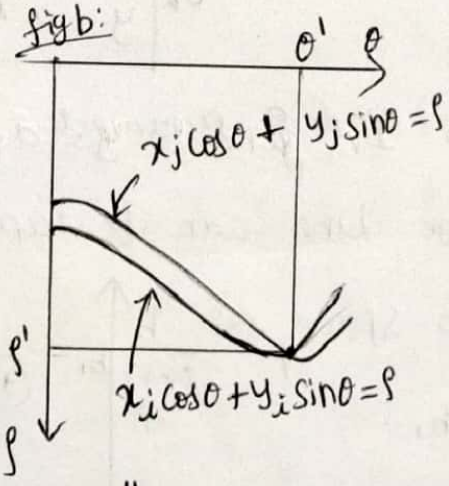
* A practical difficulty with this approach, is that when slope (i.e. a) of the line approaches infinity as the line approaches the vertical direction. one way to come around this difficulty is to use the normal representation of a line: (i.e)

$$x\cos\theta + y\sin\theta = \rho \quad \text{---} (1)$$



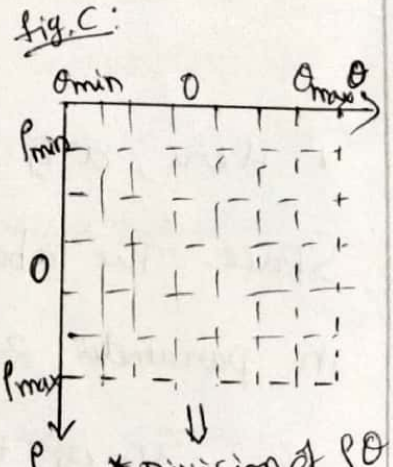
⇓

Geometrical interpretation of Eqn. (1)



⇓

* Sinusoidal curves in ρθ-plane; the pt of intersection (ρ', θ') corresponds to the line - passing through points (x_i, y_i) & (x_j, y_j) in xy-Plane.



⇓

* Division of ρθ -Plane into accumulator cells.

* Hough transform does the sub-dividing of $\rho\theta$-parameter space into so-called accumulator cells as shown in fig.c where (ρ_{min}, ρ_{max}) & $(\theta_{min}, \theta_{max})$ are the expected ranges of the parameter values i.e. $-90^\circ \leq \theta \leq 90^\circ$ & $-D \leq \rho \leq D$, where D is the maximum distance between opposite corners in an image.
* The number of subdivisions in the $\rho\theta$-plane determines the accuracy of the co-linearity of these points.

* Region Based Segmentation :
- i) Region Growing
- ii) Region Splitting & Merging

* As we know that objective of Segmentation is to partition an image into regions. Here, we discuss segmentation techniques that are based on finding the regions directly.

i) Region Growing :

* As the name implies, region growing is a procedure that groups pixels (or) subregions into larger regions based on predefined criteria for growth.
* The basic approach is to start with a set of "Seed" points and from these grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as specific ranges of intensity or color).

Algorithm: * Let $f(x,y)$ denote an input image array;
* $S(x,y)$ denote a seed array;
* Q denote a predicate to be applied at each location (x,y).

* Arrays f & S are assumed to be of the same size.
* A basic region growing algorithm based on 8-Connectivity may be stated as follows.

1) Find all connected components in S(x,y) & Erode each Connected Component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.

2) Form an image f_Q such that, at a pair of co-ordi (x,y)

$$f_Q(x,y) = \begin{cases} 1 & ; \text{ if the I/p image satisfies the given predicate, } Q \\ 0 & ; \text{ otherwise.} \end{cases}$$

3) Let g be an image formed by appending to each Seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.

4) Label each connected component in g with a different region label (Ex: 1,2,3 ...). This is the segmented image obtained by region growing.

* Pre-requirement: i) Selecting a set of starting points often can be based on the nature of the problem.

ii) The selection of Similarity criteria depends not only on the problem under consideration, but also on the type of image data available.

iii) Region growth should stop when no more pixels satisfy the criteria for inclusion in that region.

ii) Region Splitting and Merging:

* This method subdivide an image initially into a set of arbitrary, disjoint regions & then merge and/or split the regions in an attempt to satisfy the conditions of segmentation.
* Let R represent the entire image region and select a predicate (criteria) Q.
* One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $Q(R_i) = TRUE$.
* We start with the entire region. If $Q(R) = FALSE$, we divide the image into quadrants. If Q is FALSE for any quadrant, we subdivide that quadrant into subquadrants & so on.
* This particular splitting technique has a convenient representation in the form of so-called 'quadtrees' i.e trees in which each node has exactly four descendants as shown in fig. below.

fig a

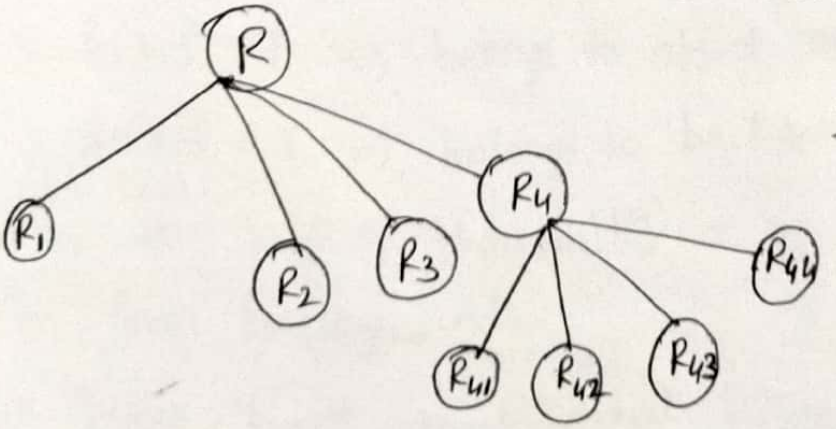
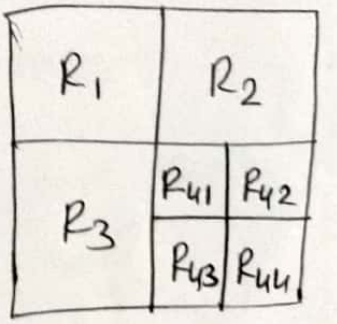


fig b



* Note that the root of the tree corresponds to the entire image & that each node corresponds to the subdivision of a node into four descendant nodes. In this case, only R_4 was subdivided further.

* After splitting merging is done i.e two adjacent regions R_j & R_k are merged only if $Q(R_j \cup R_k) = TRUE$.

⇒ The preceding discussion can be summarized by the following procedure in which, at any step, we

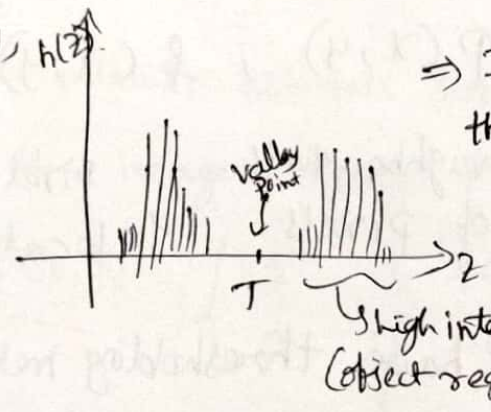
1) Split into four disjoint quadrants any region R_i, for which $Q(R_i) = FALSE$.
2) When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) = TRUE$.
3) Stop when no further merging is possible.

*

* Segmentation using "Thresholding": (1)

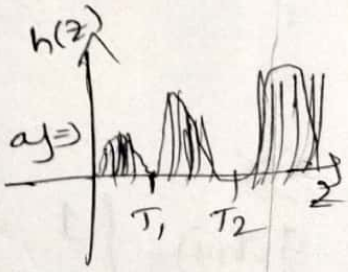
⇒ Basics of Thresholding:

* Image thresholding enjoys a central position in applications of image segmentation.
* Here we discuss techniques for partitioning images directly into regions based on intensity values and/or properties of these values.
* If we have an image $f(x,y)$, composed of light objects (high valued pixels) on a dark background then histogram of such image is given by,



⇒ In the histogram we observe that two peak modes seperated by a valley point.

* So, if we choose a threshold value 'T' at valley point, then $f(x,y) > T \Rightarrow$ belongs to object region
& $f(x,y) < T \Rightarrow$ belongs to background region.
* With the help of threshold(T) we can segment object region from background.
* We may have multimodal histogram as ⇒



In such cases we define 2-threshold values i.e. T_1 & T_2.

* Important point to be noted is, we need to choose a proper threshold value.

Different approaches (or) algorithms followed to choose a proper threshold value.

* Various factors are taken into consideration while deciding threshold value. Factors such as noise, illumination effect, reflectance effect etc.

* Basically, the value of T depends on →

$$T = T[(x,y), p(x,y), f(x,y)]$$

→ Pixel location, neighbourhoods of pixels, pixel intensity at location (x,y)

* Based on this, we have thresholding methods classification as

T
- Global thresholding ⇒ $T[f(x,y)]$ ⇒ Decision of threshold value 'T' depends on intensity value of pixel at location (x,y)
- Local thresholding ⇒ $T[f(x,y), p(x,y)]$ ⇒ along with pixel intensity, it depends on neighborhood of pixels.
- Adaptive thresholding ⇒ $T[f(x,y), (x,y), p(x,y)]$ ⇒ 'T' depends on all 3 parameters.

* Ex:

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

⇒ The process given in this equation is referred as global thresholding provided T is constant over entire image.

* If the value of T changes over an image, then we use term variable thresholding.

Note: In this chapter we focus on Global thresholding only

* Basic Global Thresholding: (2)

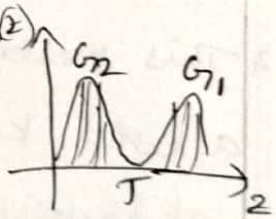
* As discussed in previous session, when the intensity distributions of objects and background pixels are sufficiently distinct, it is possible to use a single (global) threshold applicable over the entire image.

* The following iterative algorithm can be used for this purpose:

1) Select an initial estimate for the global threshold, T.

2) Segment the image using T as

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



This will produce two groups of pixels:

* G_1 Consisting of all pixels with intensity values $> T$, and * G_2 Consisting of pixels with values $\leq T$

3) Compute the average (mean) intensity values m_1 & m_2 for the pixels in G_1 and G_2, respectively.

4) Compute a new threshold value:

$$T = \frac{1}{2}(m_1 + m_2)$$

5) Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than predefined parameter ΔT.

This simple algorithm works well in situations where there is a reasonably clear valley between the modes of the histogram related to objects & background.

* The preceding algorithm was stated interms of - Successively thresholding the input image & calculating the means at each step.
* However, it is possible to develop a more efficient procedure which has to be computed only once.

Optimum Global Thresholding Using "Otsu's Method":

* The otsu's method is an attractive alterate method.
* This method or algorithm maximizes the "between-class variance", a well known measure used in statistical discriminant analysis.
* Algorithm: Let $\{0, 1, 2 \cdots L-1\}$ denote the L distinct intensity levels in a digital image of size $M \times N$ pixels.
* Let n_i denote the number of pixels with intensity i.
* The total number, MN, of pixels in the image is

$$MN = n_0 + n_1 + n_2 + \cdots + n_{L-1}.$$

* The normalized histogram has components $p_i = n_i / MN$, such that

$$\sum_{i=0}^{L-1} p_i = 1 \quad , \quad p_i \geq 0 \qquad \text{---(1)}$$

* Now, if we select a threshold $T(k) = k$, $0 < k < L-1$, & if we use this threshold to threshold the input image into two classes, C_1 & C_2 where (3)

* C_1 ⇒ having pixels with intensity values in the range $[0, k]$
* C_2 ⇒ " " " " " " $[k+1, L-1]$

* Using this threshold, the probability, $P_1(k)$, that a pixel is assigned to class C_1 is given by cumulative sum as

$$P_1(k) = \sum_{i=0}^{k} P_i \quad \text{—(2)}$$

* Similarly, the probability of class C_2 occurring is

$$P_2(k) = \sum_{i=k+1}^{L-1} P_i = 1 - P_1(k) \quad \text{—(3)}$$

* The mean intensity value of the pixels assigned to class C_1 is,

$$m_1(k) = \sum_{i=0}^{k} i\, P(i|C_1) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i\, P_i \quad \text{—(4)}$$

* Similarly, the mean intensity value of the pixels assigned to class C_2 is

$$m_2(k) = \sum_{i=k+1}^{L-1} i\, P(i|C_2) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i\, P_i \quad \text{—(5)}$$

* The cumulative mean (average intensity) up to level k is given by,

$$m(k) = \sum_{i=0}^{k} i \cdot P_i \quad \text{—(6)}$$

* & the average intensity of the Entire image (i.e the global mean) is given by,

$$m_G = \sum_{i=0}^{L-1} i\, p_i \quad \text{---(7)}$$

* In order to evaluate the "goodness" of the threshold at level k we use the normalized, dimensionless metric

$$\eta = \frac{\sigma_B^2}{\sigma_G^2} \quad \text{---(8)}$$

where σ_G^2 is the global variance, given by

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i \quad \text{---(9)}$$

& σ_B^2 is the 'between-class variance', defined as

$$\sigma_B^2 = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2 \quad \text{---(10)}$$

This Expression can be written also as

$$\sigma_B^2 = P_1 P_2 (m_1 - m_2)^2 = \frac{(m_G P_1 - m)^2}{P_1 (1 - P_1)} \quad \text{---(11)}$$

* Reintroducing k, we have the final results:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \quad \text{---(12)}$$

&

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \quad \text{---(13)}$$

* then, the optimum threshold is the value, k^*, that maximizes $\sigma_B^2(k)$: (4)

$$\sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k)$$

* once k^* has been obtained, the I/p image $f(x,y)$ is segmented as before:

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

* Separability measure, η^*, by evaluating above Eqn $\eta(k)$ at $k = k^*$ ∴ it should be

$$0 \le \eta(k^*) \le 1$$

* So, the above Otsu's algorithm may be summarized as:

1) Compute the normalized histogram of the I/p image. Denote the components of the histogram by p_i, $i = 0, 1, 2 \cdots L-1$.

2) Compute the cumulative sums, $P_1(k)$, for $k = 0, 1, 2 \cdots L-1$, using Eqn (2)

3) Compute cumulative means, $m(k)$, for $k = 0, 1, 2 \cdots L-1$, using Eqn (6)

4) Compute the global intensity mean, m_G, using Eqn (7)

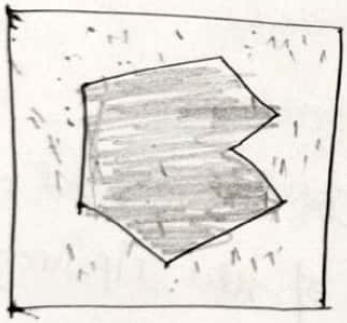
5) Compute the 'between-class variance', $\sigma_B^2(k)$, for $k = 0, 1, 2 \cdots L-1$ using Eqn (13)

6) obtain the Otsu threshold, k^*.

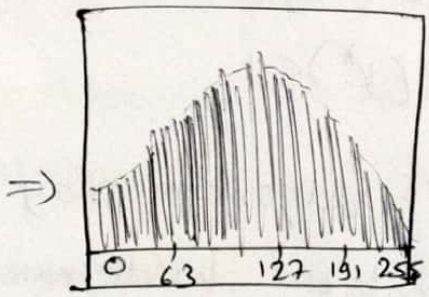
7) Obtain the Separability measure, η^*, by Evaluating Eqn (12) at $k = k^*$

* Using Image Smoothing to improve Global Thresholding:

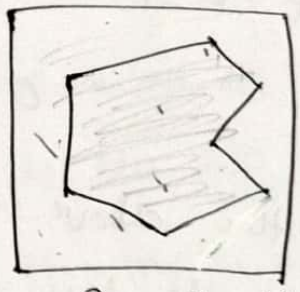
* As wkt, noise can turn a simple thresholding problem into an unsolvable one. When noise cannot be reduced at the source & if the thresholding is the Segmentation method of choice then a technique that often enhances performace is to Smooth the image prior to thresholding.
* It is Explained in the following Example:



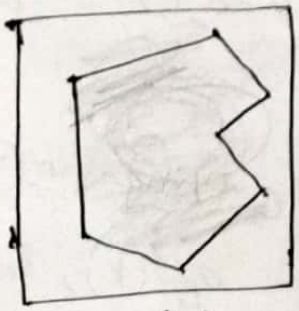
* Noisy image



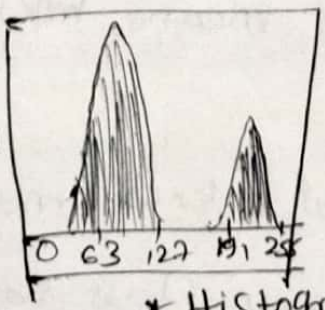
* Histogram of Noisy image. Difficult to choose threshold (T) Value due to noise presence.



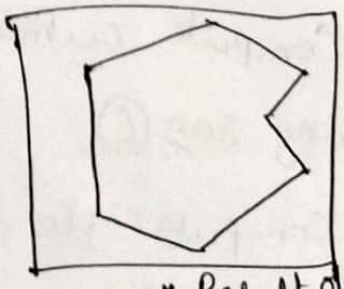
* Result obtained using otsu's method for noise image.



* Smoothed image using 5x5 averaging mask (i.e noise is removed)



* Histogram of Smoothed image. 'T' Value can be choosen accurately at valley.



* Result of thresholding using otsu's method. (Reduced Noise O/P).

* "Using Edges" to Improve Global Thresholding:

(5)

* One approach for improving the shape of histograms is to consider only those pixels that lie on (or) near the edges between objects & the background.
* An immediate and obvious improvement is that histograms would be less dependent on the relative sizes of objects & the background.
* The approach just discussed assumes that the edges between objects & background are known.
* An indication of whether a pixel is on an edge may be obtained by computing its gradient or Laplacian.
* The preceding discussion is summarized in the following algorithm, where $f(x,y)$ is the I/p image.

1) Compute an edge image as either the magnitude of the gradient, or absolute value of the Laplacian, of $f(x,y)$ using any of the methods discussed under the topic of → point, line & edge detection.
2) Specify a threshold value, T.
3) Threshold the image from Step1 using the 'T' from Step 2 to produce a binary image, $g_T(x,y)$. This image is used as a mask image in the following step to select pixels from $f(x,y)$ corresponding to "Strong" edge pixels.

4) Compute a histogram using only the pixels in $f(x,y)$ that correspond to the locations of the 1-valued pixels in $g_T(x,y)$.

5) use the histogram from Step4 to segment $f(x,y)$ globally using, for ex., otsu's method.

* Multiple thresholds :-

* So far, we have focused attention on image segmentation using a single global threshold. The thresholding method can be extended to an arbitrary number of thresholds.

* In the case of K classes $C_1, C_2, \cdots C_K$, the 'between class-variance' generalizes to the expression

$$\sigma_B^2 = \sum_{k=1}^{K} P_k (m_k - m_G)^2$$

where

$$P_k = \sum_{i \in C_k} P_i$$

$$m_k = \frac{1}{P_k} \sum_{i \in C_k} i P_i$$

* For 3-classes consisting of 3-intensity intervals, the 'between-class variance' is given by:

$$\sigma_B^2 = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2 + P_3 (m_3 - m_G)^2$$

Where $P_1 = \sum_{i=0}^{k_1} P_i$, $P_2 = \sum_{i=k_1+1}^{k_2} P_i$, $P_3 = \sum_{i=k_2+1}^{L-1} P_i$ (6)

& $m_1 = \frac{1}{P_1} \sum_{i=0}^{k_1} i P_i$, $m_2 = \frac{1}{P_2} \sum_{i=k_1+1}^{k_2} i P_i$, $m_3 = \frac{1}{P_3} \sum_{i=k_2+1}^{L-1} i P_i$

* we find the optimum thresholds by finding

$$\sigma_B^2(k_1^*, k_2^*) = \max_{0<k_1<k_2<L-1} \sigma_B^2(k_1, k_2)$$

* The thresholded image is given by

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) \le k_1^* \\ b & \text{if } k_1^* < f(x,y) \le k_2^* \\ c & \text{if } f(x,y) > k_2^* \end{cases}$$

where a, b & c are any 3-valid intensity values.

* finally, we note that the Separability measure to multiple thresholds:

$$\eta(k_1^*, k_2^*) = \frac{\sigma_B^2(k_1^*, k_2^*)}{\sigma_G^2}$$

where σ_G^2 is the total variance.

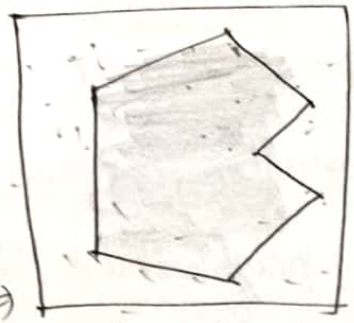
* Variable Thresholding :

* As wkt, factors such as noise & non-uniform illumination play a major role in the performance of a thresholding algorithm.
* We showed that image Smoothing & using Edge information can help significantly. However, it frequently is the case that this type of preprocessing is either impractical or simply ineffective in some situations.
* In such situations, the next level of thresholding complexity involves variable thresholding.
* In this section, we discuss various techniques for choosing variable thresholds.

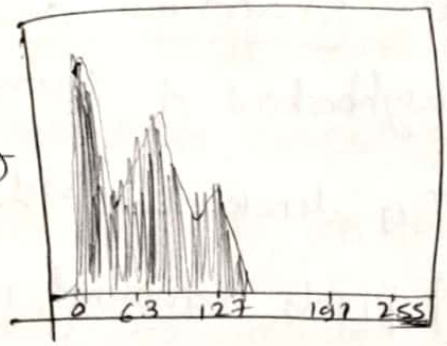
*i) Image Partitioning :-

* One of the simplest approaches to variable thresholding is to subdivide an image into non overlapping rectangles.
* This approach is used to compensate for non-uniformities in illumination and/or reflectance.
* The rectangles are chosen small enough so that the illumination of each is approximately uniform.
* The below fig. shows the original image subdivided into six rectangular regions & for each region Otsu's global

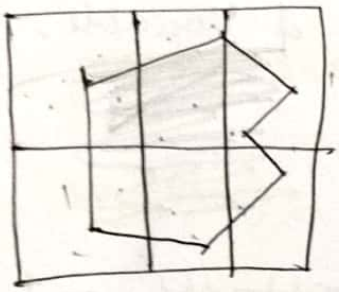
- thresholding approach applied to get final o/p. (7)



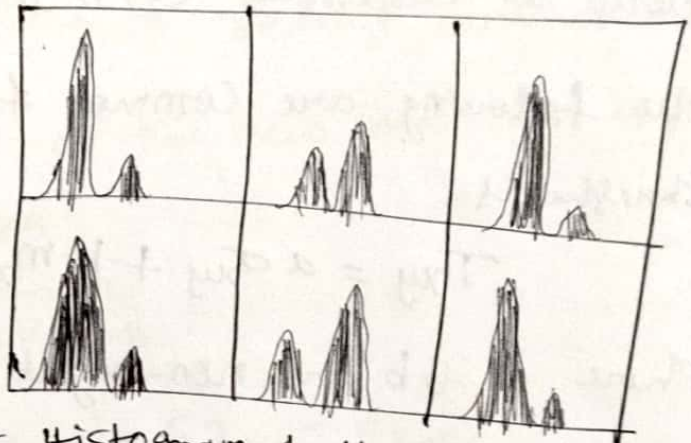
* Noisy, Shaded image



⇒ Histogram of Noisy image. Difficult to choose threshold value 'T'.

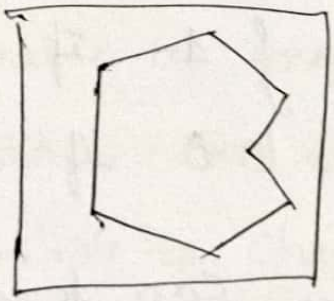


* Image subdivided into Six subimages



* Histogram of the six subimages

⇓ After applying otsu's thresholding method to Each Subimage individually, find o/p image.



⇒ reduced noise.

ii) * Variable thresholding based on local image properties:

* A more general approach than the image subdivision method discussed in the previous section is to Compute a threshold at Every point, (x, y), in the image based on one (or) more specified properties computed in a "neighborhood" of (x, y).

* We illustrate the basic approach to local thresholding using the standard deviation & mean of the pixels in a neighborhood of every point in an image.

* Let σ_{xy} & m_{xy} denote the std. deviation & mean value of the set of pixels contained in a neighborhood, S_{xy}, centered at coordinates (x,y) in an image.

* The following are common forms of variable, local thresholds:

$$T_{xy} = a\sigma_{xy} + b\, m_{xy}$$

where a & b are non-negative constants, and

$$T_{xy} = a\sigma_{xy} + b\, m_G$$

where m_G is the global image mean. The segmented image is computed as

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

(f(x,y) → I/p image.)

* Significant power can be added to local thresholding by using predicates on the parameters computed in the neighborhoods of (x,y):

$$g(x,y) = \begin{cases} 1 & \text{if } Q(\text{local parameter}) \text{ is true} \\ 0 & \text{if } Q(\text{local parameter}) \text{ is false} \end{cases}$$

⇒ where Q is a predicate based on parameters computed using the pixels in neighborhood S_{xy}.

iii) * Using moving averages : (8)

* A Special Case of the local thresholding method just discussed is based on computing a moving average along Scan lines of an image.
* The Scanning typically is carried out line by line in a zigzag pattern to reduce illumination bias.
* The moving average at the new point 'K+1' is given by:

$$m(K+1) = \frac{1}{n} \sum_{i=K+2-n}^{K+1} z_i$$

$$= m(K) + \frac{1}{n}(z_{K+1} - z_{K-n})$$

---

* Multivariable Thresholding :-

* So far, we have been concerned with thresholding - based on a single variable ie. gray-scale intensity.
* In Some cases, a sensor can make available for more than one variable to characterize each pixel in an image, & thus allow multivariable thresholding.
* A notable example is color imaging, where red(R), green (G) & blue (B) components are used to form a Composite color image.

* In this case, each "pixel" is characterized by three values, & can be represented as a 3-D vector, $z = (z_1, z_2, z_3)^T$, whose components are the RGB colors at a point.

* These 3-D points often are referred to as Voxels, to denote volumetric elements, as opposed to image elements.

* Multivariable thresholding may be viewed as a distance computation. One way to segment a color image based on this parameter is to compute a distance measure, $D(z, a)$ between an arbitrary color point 'z' & avg. color 'a'. Then we segment the I/p image as follows:

$$g = \begin{cases} 1 & \text{if } D(z,a) < T \\ 0 & \text{otherwise.} \end{cases}$$

* This $D(z,a)$ using n-dimensional Euclidean distance is

$$D(z,a) = \|z-a\| = [(z-a)^T (z-a)]^{1/2}$$

* A more powerful distance measure is the so-called Mahalanobis distance, defined as

$$D(z,a) = [(z-a)^T \cdot C^{-1} (z-a)]^{1/2}$$

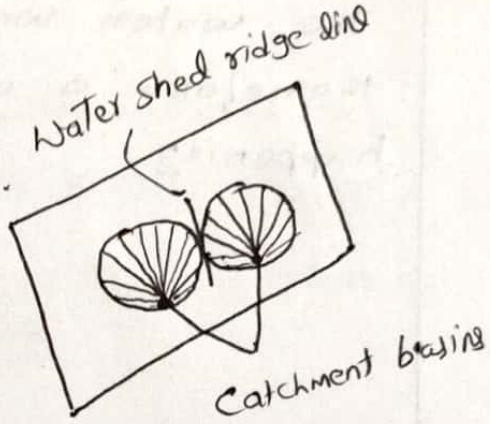
where C is the Co-variance matrix of the zs.

Segmentation using Morphological Watershed

Watershed is useful for segmenting objects that are touching each other

To understand the watershed transform - we view a grayscale image as a topological surface, where the values of $f(x,y)$ corresponds to heights

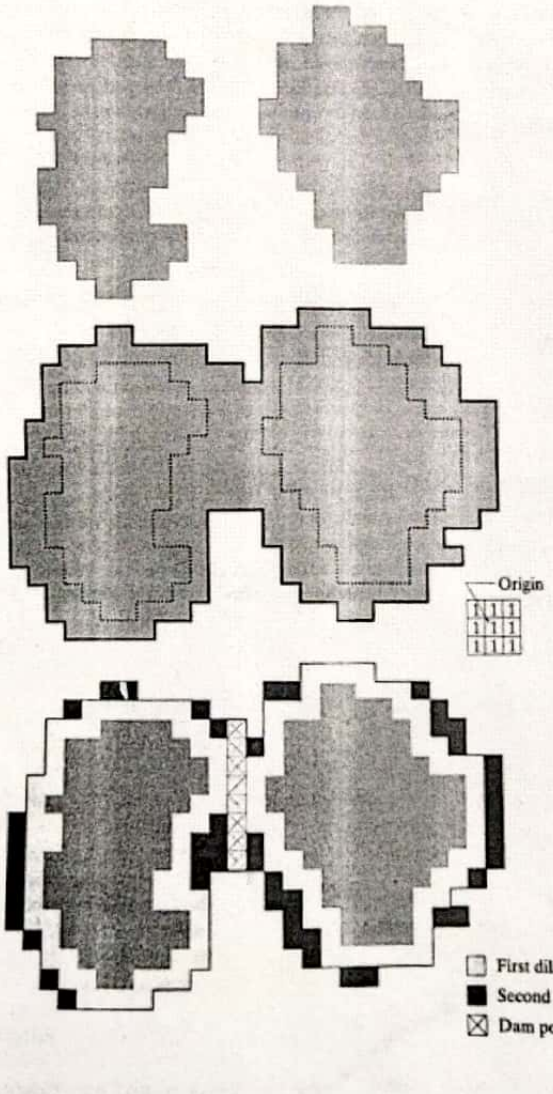
Consider the topographic surface shown in the figure. Water falling on watershed ridge line separating the two catchment basins would be equally likely to collect into either of the two catchment basins. watershed algorithms then find the catchment basins and the ridge lines in an image.



Suppose a hole is punched at each regional local minimum and the entire topography is flooded from below by letting the water rise through the holes at uniform rate. Pixels below the water level at a given time are marked as flooded. when we rise the water level incrementally, the flooded regions will grow in size. Eventually the water will rise to a level where two flooded region from seperate catchment basins will merge. when this occurs the algorithm constructs a one pixel thick dam that seperates the two region. The flooding continues until the entire image is segmented into seperate catchment basins divided by watershed ridge lines.

Dam construction

The simplest way to construct the dams or watershed lines is by using — morphological dilation. Fig (a) shows partitions of two catchment basins at flooding step n-1 and fig (b) shows the result at next flooding step n. the water was spilled from one basin to other therefore a dam must be built to keep this from happening.



a
b c
d

FIGURE (a) Two partially flooded catchment basins at stage $n-1$ of flooding. (b) Flooding at stage n, showing that water has spilled between basins. (c) Structuring element used for dilation. (d) Result of dilation and dam construction.

Watershed segmentation Algorithm

Let $M_1, M_2, \cdots M_R$ be sets denoting coordinates of the points in the regional minima of a image $g(x,y)$. Let $C(M_i)$ be a set denoting the coordinates of the points in the catchment basin associated with regional minimum M_i. Let $T[n]$ represent the set of coordinates (s,t) for which $g(s,t) < n$

i.e $T[n] = \{(s,t) \mid g(s,t) < n\}$

Geometrically, $T[n]$ is a set of coordinates of points in $g(x,y)$ lying below the plane $g(x,y) = n$

Let $C[n]$ denote the union of the flooded catchment basins at stage n

$$C[n] = \bigcup_{i=1}^{R} C_n(M_i)$$

then $C[max+1] = \bigcup_{i=1}^{R} C(M_i)$

The algorithm for finding the watershed lines is initialized with $C[min+1] = T[min+1]$. The procedure for finding $C[n]$ from $C[n-1]$ is as follows. Let Q denote set of connected component in $T[n]$ then for each connected component $q \in Q(n)$ there are 2 possibilities.

1. $q \cap C[n-1]$ is empty
2. $q \cap C[n-1]$ contains one connected component of $C[n-1]$
3. $q \cap C[n-1]$ contains more than one connected component of $C[n-1]$

condition 1 occurs when a new minimum is encountered. condition 2 occurs when g lies within the catchment basin of some regional minimum. condition 3 occurs when all or part of ridge seperating two or more catchment basin is encountered.

Use of Markers

Direct application of watershed segmentation algorithm leads to oversegmentation due to noise and other local irregularities of gradient. To avoid this marker is used. A marker is a connected component belonging to an image.

Procedure for marker selection :-

(i) Preprocessing (ii) Definition of a set of criteria that marker must satisfy.

There are 2 markers

(i) Internal marker → Associated with objects of interest

(ii) External marker → Associated with background.

Image Representation

*(i) Boundary following
*(ii) Chain code
* (iii) Polygon approximation using MPP

} Refer Part A notes.

(iv) Other polygon approximation Approaches.

Representation :-

From segmentation, image features or image attributes are extracted. But, these extracted features and attributes cannot be used by a computer/machines. For that purpose, we use descriptors, which makes image ready for recognition.

The output of segmentation can be applied directly to descriptors. But, it is better to use representation before description. By description, image compactness is increased.

1) Boundary following (Border following) :-

The result of segmentation is a set of regions. These regions should be represented and described. Two ways of representing a region are,

1) External characteristics. (its boundary : focus on shape).
2) Internal characteristics. (its internal pixels : focus on colour, textures.).

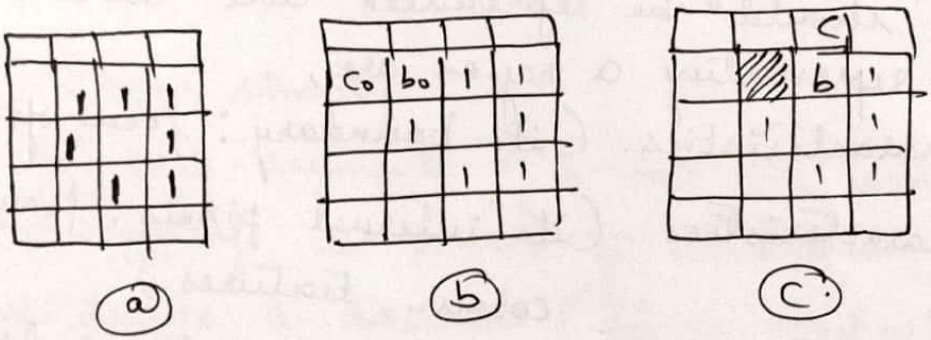
In boundary following algorithm, the output is an ordered sequence of points.

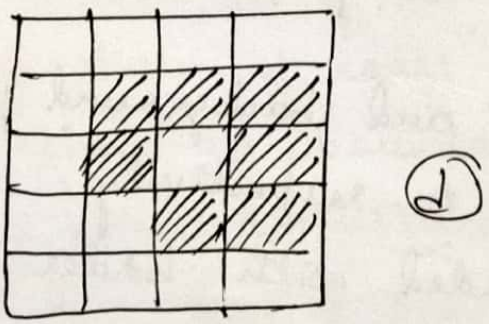
Assume,

1) In image, object and background points are labelled 1 and 0 respectively.
2) Images are padded with border of zero's to eliminate the possibility of an object merging with image border.

Boundary following algorithm :-

1) Let the starting point be the leftmost point (b_0) in the image. Denote c_0 the west neighbour of b_0. Start at c_0 and proceed in clockwise direction. Let b_1 is the first neighbour and c_1 be the background point in the sequence. Store the locations b_0 and b_1.
2) Let $b = b_1$ and $c = c_1$. Fig (c)
3) Let the 8 neighbour of b start at c and proceed in clockwise direction. $(n_1, n_2, \ldots n_k)$.
4) Let $b = n_k$ and $c = n_k - 1$.
5) Repeat steps 3 and 4 until $b = b_0$ and next boundary point is b_1.



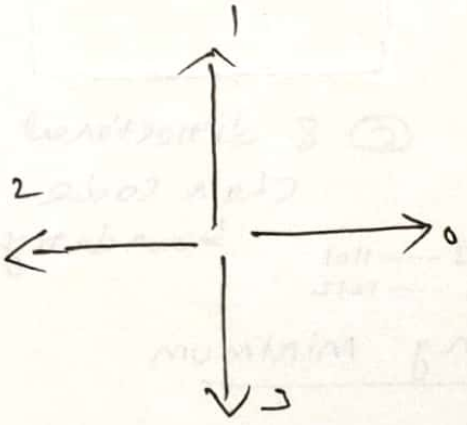


(d)

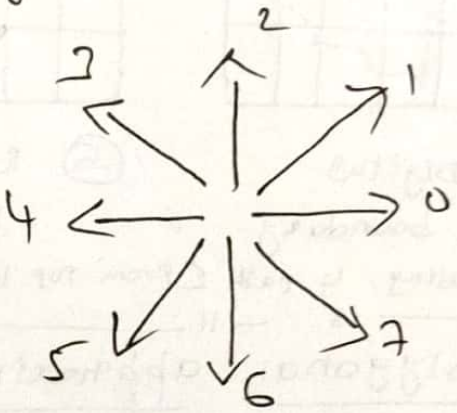
Fig :- Illustration of boundary filling algorithm.

2) Chain codes :-

These are used to represent a boundary by a connected sequence of straight line segments of specific length and direction. Typically, this representation is based on 4 or 8 connectivity of the segment.



a) Four directional chain code



b) 8 directional chain code

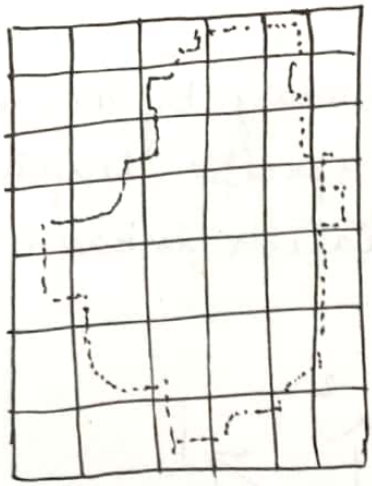
Now, simply

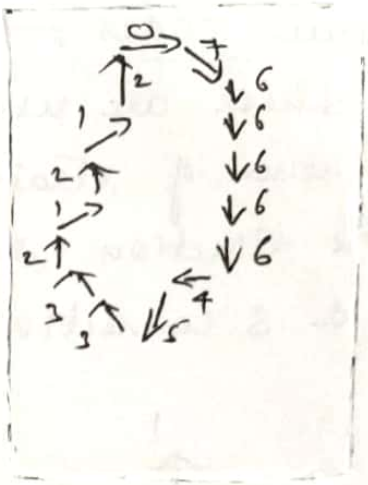
1) Divide each segment of image in small grids with equal spacings in x and y directions.
2) Now, decide a direction (for eg; clockwise). Assign codes for every pixel. (x, y).

But, the disadvange of this chain code is that,

1) The resulting chain of codes is very long.
2) Any small change in boundary with respect to noise or any other disturbance, code will be changed.

To avoid this type of problem, we resample the boundary by selecting a large grid spacing.





fig(a): Digital boundary

(b) Result of Resampling

(c) 8 directional chain code boundary

Code using 4 path (From top left corner) = 0033 1101

— 11 — 8 — 11 — = 0766 --- 1212

3) Polygonal approximation using minimum perimeter polygon

Consider the perimeter (boundary) as shown in below figure

* Enclose the boundary using sampling grid
* Assume the boundary as rubber band which is placed in between 2 walls
* Now image that the rubber band is shrinked so that it touches the inside wall i.e cell strips

The shape taken after shrinking gives the minimum perimeter polygon. This kind of approximation is also known as rubber band approximation

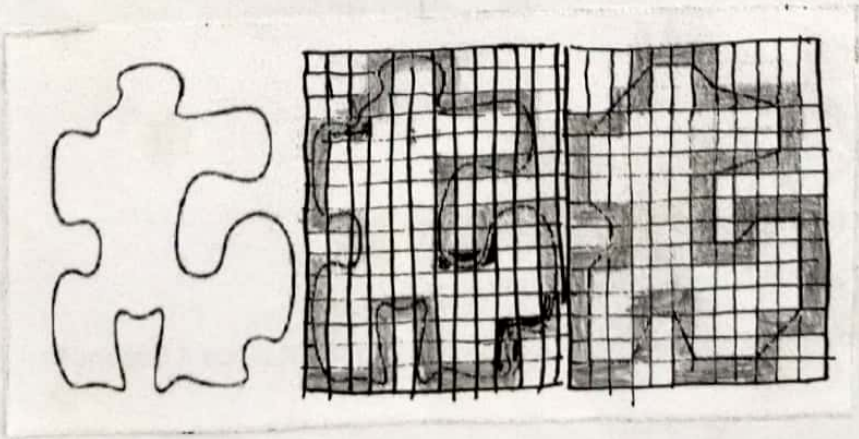


Fig. Polygon approximation using minimum perimter polygon

(iv) Other polygon approximation approaches

(1) Merging techniques (11)

* Set a threshold for least square error
* Merge some points along boundary. This is done until the least square error line fit the points merged so far exceeds the threshold.
* When it exceeds, store the paramets of the line and set the error to zero.
* Repeat the steps (2) and (3) by merging new points until the error again exceeds the threshold. Thus many lines are created around the boundary.
* At last, the intersection produced by adjacent line segments gives vertices i.e corner points of the polygon and joining the vertices will form the polygon.

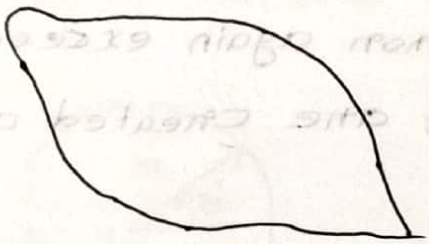
Drawback

The vertices resulting from this method are not exactly related with corners because a new line is started only after the error threshold is exceeded.

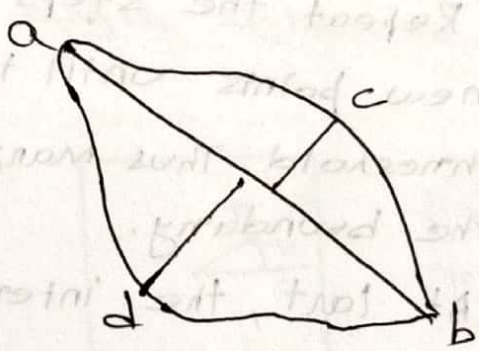
(2) Splitting techniques.

* Join two far points on the boundary (line ab)

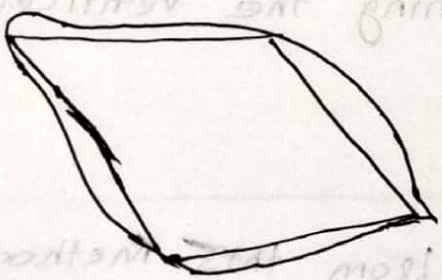
* obtain a point on the upper segment, i.e c and a point on lower segment i.e d, such that perpendicular distance from there to ab is as large as possible
* Now obtain the polygon by joining c and d with a and b
* Repeat until the perpendicular distance is less than some predefined fraction of ab



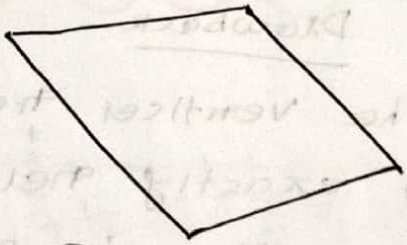
(a) original boundary



(b) Boundary divided into segments based on extreme points



(c) Joing of vertices



(d) Resulting polygon

(3) Signature

A signature is a 1D representation of a boundary. It is used to reduce the original 2D boundary to 1D function so that the description becomes easier.

There are many ways to generate a signature one way is plot the distance from centroid to the boundary as a function of angle. It is called distance verses angle signature.

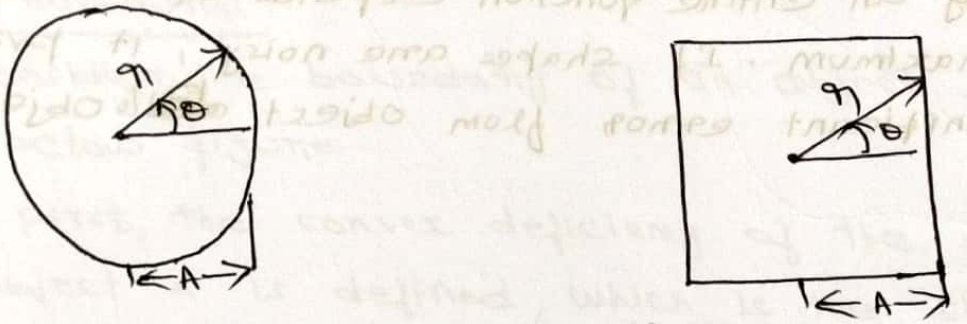
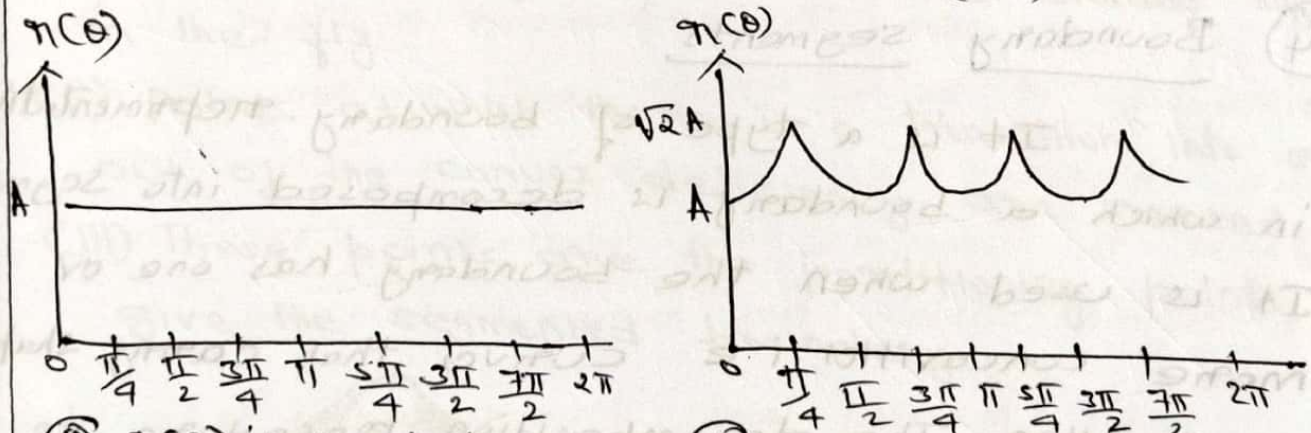


fig: Distance verses angle signature.



(a) r(θ) is constant (b) signature consist of repetation

* signatures are invarient to translation & Rotation i.e starting point can be far from center
* Scaling varies the amplitude of signature.

Normalization for Rotation

(1) Choose the starting point as a far point from the centroid or

(2) Choose the starting point as the point on the major axis that is far from center

Sampling is taken at each intervals of θ. Change in ~~shape~~ size of a shape results in changes in amplitude values of corresponding signature.

Advantage

* This Method is very simple

Disadvantage

* scaling an entire function depends on minimum and maximum. If shapes are noisy, it produces a significant error from object ~~to~~ to object.

(4) Boundary segments

It is a type of boundary representation in which a boundary is decomposed into segments It is used when the boundary has one or more concavities i.e curves that carry shape information. The decomposition procedure is used to

(i) Reduce the complexity of boundary

(ii) simplify the description process

Convex Hull, (H)

A set A is said to be convex if the straight line joining any 2 points in A lies entirely within A. The convex hull H of an arbitrary set S is the smallest convex set containing S. The set difference H − S is called convex deficiency of S. The convex hull of the region enclosed by the boundary is used as a tool for efficient decomposition of the boundary.

Procedure

Consider the boundary of an object S shown in in below figure

(i) First, the convex deficiency of the set i.e object S is defined, which is the shaded region in the fig

(ii) Points at which there is a transition into or out of the convex deficiency are marked

(iii) These points are the partitioning points that give the segmented boundary

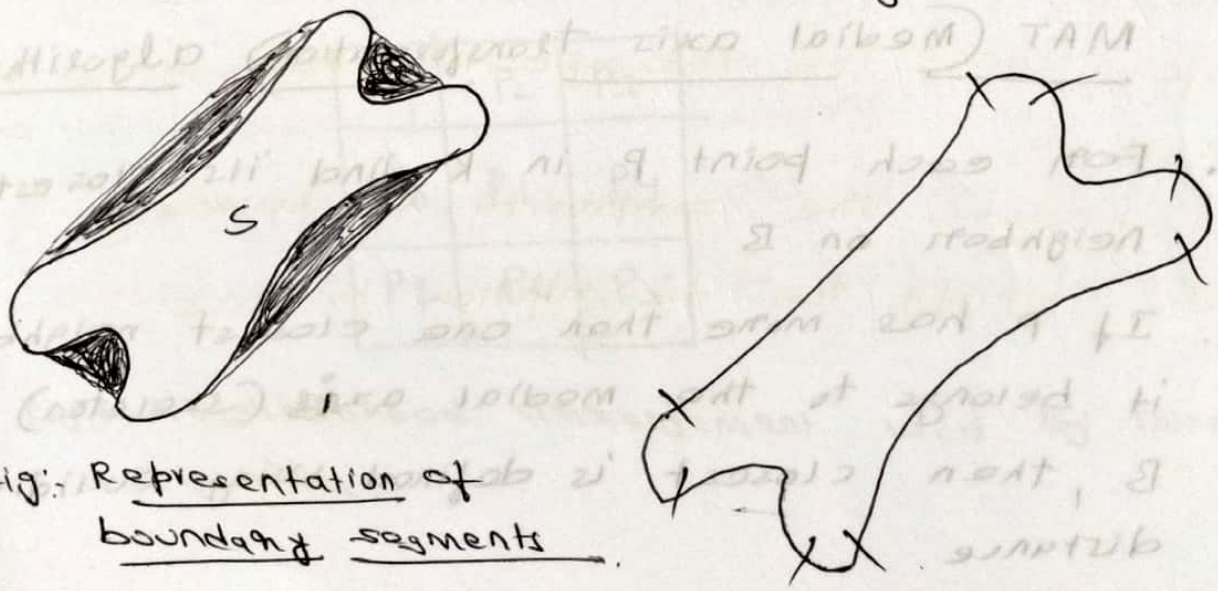


Fig: Representation of boundary segments.

Advantage :

The advantage of segmenting a boundary using convex deficiency is that it is independent of the size and orientation of the given region. The convex hull and its deficiency can also be used for describing a region and its boundary.

Drawback :

Noise and variations in segmentation may lead to irregular boundaries. Such boundaries produce convex deficiencies with small meaningless components scattered randomly throughout the boundary. This results in an insufficient decomposition process.

(5) Skeleton

One way to represent a shape is to reduce it to a graph, by obtaining its skeleton via thinning (skeletonization). Skeletons are used to analyze the geometric structure of a region which has bumps and arms

MAT (Medial axis transformation) algorithm

1. For each point P in R find its closest neighbor on B
2. If P has more than one closest neighbor it belongs to the medial axis (skeleton) of B, then closest is defined using Euclidian distance

MAT calculates the distance from every interior point to every point on the boundary of a region. Numerous algorithms have been proposed for improving computational efficiency and also produce a medial axis representation of a region. These are the thinning algorithms that iteratively delete boundary points and (i) It does not remove end points (ii) does not break connectivity (iii) does not cause excessive erosion of the region

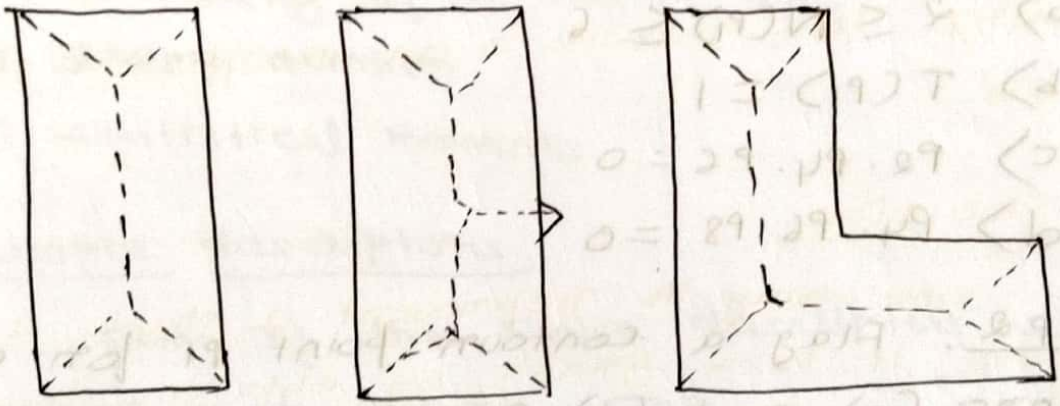


Fig a:- Medial axes (dashed) of three simple regions

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Fig b:- Neighborhood arrangement used by thinning algorithm

Let $N(P_1) = P_2 + P_3 + \cdots + P_9$ where $N(P_1)$ is the number of nonzero neighbors of P_1.

Let $T(P_1)$ is the no of 0–1 transitions in the ordered sequence $P_2, P_3, \cdots P_9, P_2$

0	0	1
1	P_1	0
1	0	1

} Here $N(P_1) = 4$, $T(P_1) = 3$

Fig (c)

<u>step 1</u> :- Flag (indicate) a contour point P_1 for deletion if the following conditions are satisfied

a) $2 \le N(P_1) \le 6$

b) $T(P_1) = 1$

c) $P_2 \cdot P_4 \cdot P_6 = 0$

d) $P_4 \cdot P_6 \cdot P_8 = 0$

<u>step 2</u> : Flag a contour point P_1 for deletion, where (a) and (b) remains same but (c) and (d) are changed to

$P_2 \cdot P_4 \cdot P_8 = 0$ and

$P_2 \cdot P_6 \cdot P_8 = 0$

This procedure is applied until no further points are deleted. Application of skeletonization is character recognition.

Boundary descriptors

Boundary descriptors describe the boundary of a region using the feature of the boundary.

Types of boundary descriptors

1. simple descriptors
2. Fourier descriptors

Some qualities which are used to describe the boundary of a region are

1. shape number
2. statistical moments

simple descriptors

Some of the simple descriptors used to describe a boundary are

(i) Length :- It is given by the number of pixels along the boundary. For a chain coded curve which has unit spacing in both directions, the length is obtained by

Length = (Number of horizontal and vertical components) + $\sqrt{2}$ (Number of diagonal components)

(ii) Diameter :- The diameter of a boundary B is described as

$$\text{Diam}(B) = \max [D(P_i, P_j)]$$

Where D = Distance measure

P_i, P_j = Points on boundary

(iii) Major axis :- It is defined as line segment connecting the two extreme points of its diameter.

(iv) Minor axis :- It is defined as the line perpendicular to major axis.

(v) Eccentricity :- The ratio of major axis to minor axis is called eccentricity of a boundary.

(vi) Basic Rectangle :- The major axis intersects with the boundary at two points and the minor axis also intersects with the boundary at two points. A box which completely enclose the boundary by passing through these four outer points is called the basic rectangle

(vii) Curvature :- Curvature is defined as rate of change of its slope. Measuring the curvature of a point on a digital boundary is very difficult.

Fourier descriptors

Fourier descripton describe a digital boundary by considering it as a complex sequence. The xy coordinates of the boundary are treated as the real and imaginary parts of a complex number. The the list of coordinates is fourier transformed using the DFT. The Fourier coefficients are called Fourier descriptors.

Consider a digital boundary shown in below fig which has K number of points in xy plane

Starting from point $(x_0, 0)$ and following the boundary in anticlockwise direction, its coordinate pairs are $(x_0, y_0), (x_1, y_1), (x_2, y_2) \cdots (x_{K-1}, y_{K-1})$

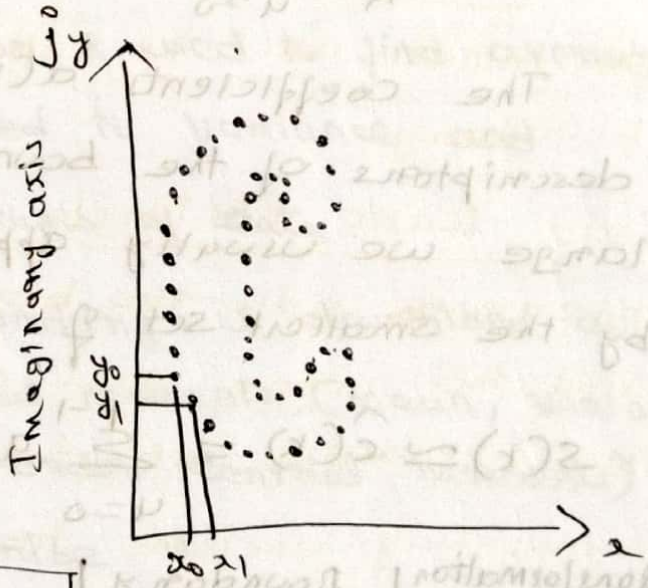


Fig: Fourier descripton

The coordinates can be represented as

$x(k) = x_k$ and
$y(k) = y_k$ → (1)

Now the coordinate pairs represented as a complex number in the form

$$s(k) = x(k) + jy(k) \quad \text{for } k = 0, 1, 2, \cdots K-1 \quad \text{— (2)}$$

The sequence of coordinates of boundary are

$$s(k) = [x(k), y(k)] \text{ for } k = 0, 1, 2 \cdots K-1 \quad \text{---(3)}$$

where x axis is considered as real axis and y axis is considered as imaginary axis

The Forward DFT of the complex sequence $s(k)$ is expressed as

$$a(u) = \sum_{k=0}^{K-1} s(k)\, e^{-j\frac{2\pi uk}{K}} \quad \text{for } u = 0, 1, 2 \cdots K-1 \quad \text{---(4)}$$

$a(u)$ are called fourier descriptor of boundary.

The Inverse fourier transform is given by

$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u)\, e^{\frac{j2\pi uk}{K}} \quad \text{---(5)}$$

The coefficients $a(u)$ are the Fourier descriptors of the boundary. since K can be large we usually approximate the boundary by the smallest set of points i.e p so that

$$s(k) \simeq \hat{s}(k) = \sum_{u=0}^{P-1} a(u)\, e^{\frac{j2\pi uk}{K}}$$

Transformation	Boundary	Fourier descriptor
1) Identity	$s(k)$	$a(u)$
2) Rotation	$s_R(k) = s(k)e^{j\theta}$	$a_R(u) = a(u)e^{j\theta}$
3) Translation	$s_R(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\,\delta(u)$
4) Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha\, a(u)$
5) Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j}$

Table: Some basic properties of Fourier descriptors

Statistical Moments

Statistical moments can be used to describe shape of boundary segment. A boundary segment can be represented by 1D function $g(n)$. The amplitude of g can now be treated as a discrete random variable v so that a histogram $p(v_i), i = 0, 1, 2, \cdots A-1$ is formed, where A is the number of amplitude increments.

Moments are statistical measures of data. They come in integer orders.

Order 0 is the number of points in the data
order 1 is the sum & used to find average
order 2 is related to variance and
order 3 is the skew of data.

once a boundary is described as a 1D function, statistical moments (mean, variance and a few higher order central moments) can be used to describe it.

The n^{th} moment of v about its mean is

$$\mu_n(v) = \sum_{i=0}^{A-1} (v_i - m)^n p(v_i)$$

where $$m = \sum_{i=0}^{A-1} v_i p(v_i)$$

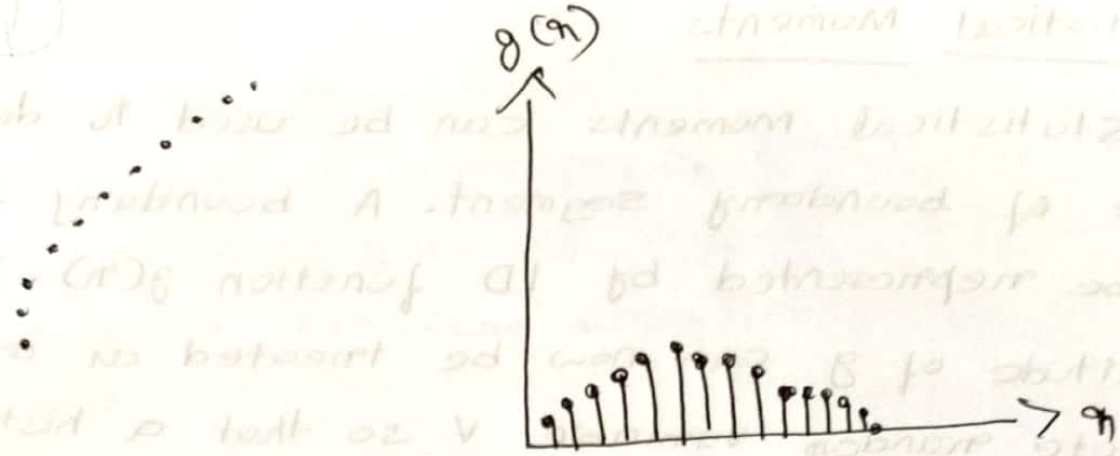


Fig ⓐ: Boundary segment Fig ⓑ: Representation as a 1D function.

An alternative approach is to normalize g(r) to unit area and treat it as a histogram. i-e g(ri) is now treated as the probablity of value ri occuring. In this case r is treated as random variable and the moments are

$$\mu_n(r) = \sum_{i=0}^{k-1} (r_i - m)\, g(r_i)$$

where

$$m = \sum_{i=0}^{k-1} r_i\, g(r_i)$$

where K = number of points on boundary

$\mu_n(r)$ = It is related to shape of g(r)

Shape number

The shape number of a boundary is defined as first difference of smallest magnitude.

The order n of a shape number is defined as the number of digits in its representation.

Procedure

To find a shape number, the steps are

(i) First find a Rectangle of order n

(ii) Establish a grid size using this new Rectangle

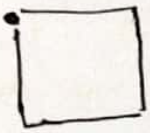
(iii) Allign a chain code direction for the resulting grid

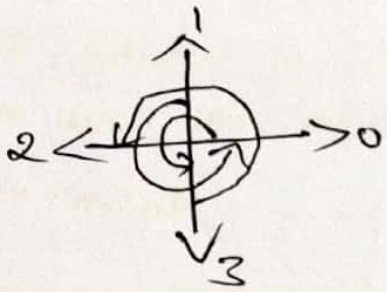
(iv) Obtain the chain code

(v) Find first difference and use it to compute the shape number

Ex: Shape number of order 4 (dot indicate the starting point)

(i)





Chain code → 0 3 2 1

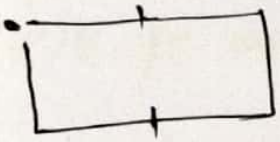
Difference → 3 3 3 3

Shape number → 3 3 3 3

Difference from 0 to 3 is 3
—11— 3 to 2 is 3
—11— 2 to 1 is 3

For shape number start from min value & write in order Here all diff value are 3
∴ shape no' is also 3 3 3 3

(ii)



order 6

Chain code → 0 0 3 2 2 1

Difference → 3 0 3 3 0 3

Shape number → 0 3 3 0 3 3