Image Segmentation :- Segmentation subdivides an image into its constituent regions or objects. The level of detail to which the subdivisions is carried depends on problem being solved. Segmentation should stop when the objects or regions of interest in an applicat^{n} have been detected.

Image Segmentation algorithms generally are based on one of the two basic properties of intensity values.

1. Discontinuity
2. Similarity.

In first category the approach is to partition an image based on abrupt changes in intensity such as edges in an image.

The principal approach in second category are based on partitioning an image into regions that are similar according to a set of pre defined criteria.

(1) Detection of Discontinuites :

Several techniques are proposed for detecting the three basic types of gray level discontinuites in a digital image,

(1) points (2) lines (3) Edges.

The most common way to look for discontinuites is to run a mask through the image.

* Detection of Isolated points

Point detection is based on the second derivative. The laplacian second order derivative is used.

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad \text{— Eq}^{n} (1).$$

Where the partials are obtained using the following eq^n.

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

Eq^n 1 becomes,

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y). \quad \rightarrow Eq(2)$$

This expression can be implemented as a mask. Using the laplacian mask, we say that a point has been detected at location (x,y) on which the mask is centered if the absolute value of the response of the mask at that point exceeds a specified threshold.

Such points are labelled as 1 in the output image f all others are labelled 0, thus producing a binary image.

In other words, the output is obtained using the following expression.

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

where g is the output image, T is a non negative threshold, and R is given by,

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k \quad \rightarrow \text{filter function.}$$

This formulation simply measures the weighted differences between a pixel & its 8-neighbors. The intensity of an isolated point will be quite different from its surroundings & can be easily detectable by this type of mask.

* Process of Line Detection

The masks for Line detectors are shown below,

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

Horizontal

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

+45°

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

Vertical

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

-45°

* If the horizontal mask were moved around an image, it would respond more strongly to line oriented horizontally.
* The co-efficients in each mask sum to zero, indicating a zero response from the masks in area of constant gray levels.
* Let R_1, R_2, R_3 & R_4 denote the response of the masks as shown above, Suppose four masks are run individually through an image.
* If a certain point in the image $|R_i| > |R_j|$ for all $j \neq i$, that point is said to be more likely associated with a line in the direction of mask.

Example:- If $|R_i| > |R_j|$ for $i = 1, 2, 3, 4$ that particular point is said to be more likely associated with a horizontal line.

* We may be interested in detecting lines in a specified direction. We use the mask associated with that direction & threshold its output.
* If we are interested in detecting all the lines in an image in the direction defined by a given mask, we run the mask throughout the image & threshold the absolute value of the result.

* The points that are left are the strongest responses which, for lines 1 pixel thick, correspond closest to the direction defined by the mask.

---

* Edge detection :

Edge detection is by far the most common approach for detecting meaningful discontinuities in gray level. For detecting edges in an image we have to use first & second order derivatives.

Basic formulation :

→ An Edge is a set of connected pixels that lie on the boundary between two regions.

→ An ideal edge has the properties of the model shown in the figure below,

Model of an ideal edge

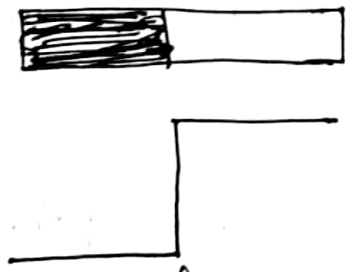


fig (a)

Model of a ramp edge

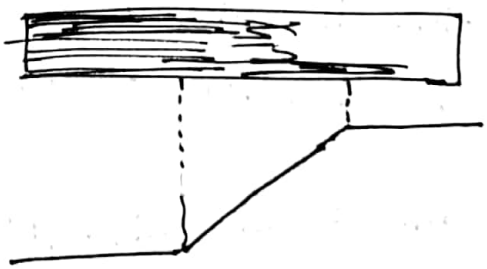


fig (b)

In practice optics, sampling & other image acquisition imperfections yield edges that are blurred as shown in fig(b). The edges are more closely modeled as having a 'ramp' like profile.

The slope of the ramp is inversely proportional to the degree of blurring in the edge.

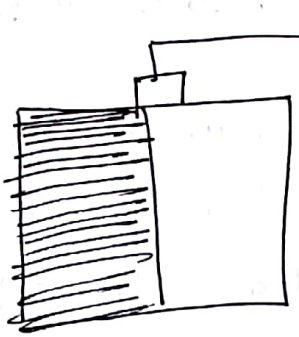


fig (a)

a → two regions separated by a vertical edge

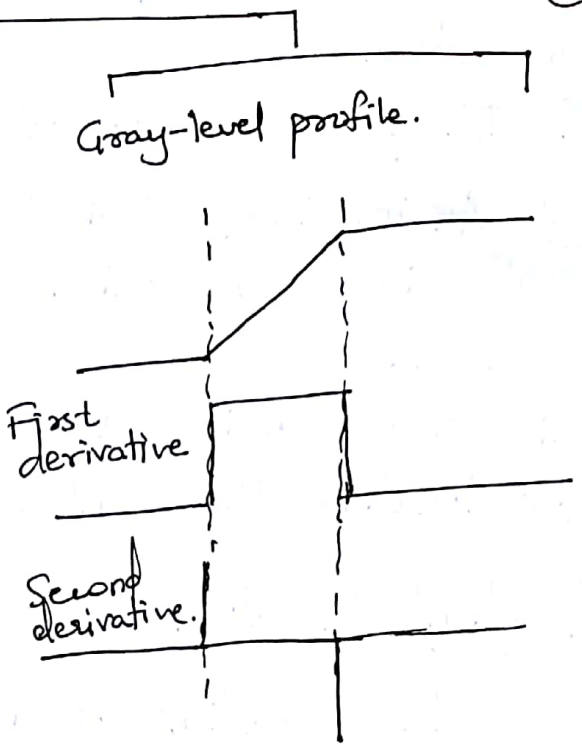


fig (b)

Detail near the edge, showing a gray level profile & the first & second derivatives of the profile.

→ The first derivative is positive at the points of transition into & out of the ramp as we move from left to right along the profile, it is constant for points in the ramp & is zero in areas of constant gray level.

→ The second derivative is positive at the transition associated with the dark side of the edge, negative at the transition associated with the right side of the edge, zero along the ramp & in areas of constant gray level.

→ The sign of the derivatives would be reversed for an edge that transitions from light to dark.

→ From these observations, the magnitude of the first derivative can be used to detect the presence of an edge at a point in an image.

→ Similarly the sign of the second derivative can be used to determine whether an edge pixel lies in the dark or light side of an edge.

Two additional properties of the second derivative around an edge.

(1) It produces two values for every edge in an image.

(2) An imaginary straight line joining the extreme positive & negative values which become zero near the midpoint of the edge.

This zero-crossing property of the second derivative is useful for locating the center.

Gradient Operators:-

The gradient of an image $f(x,y)$ at location (x,y) is defined as the vector,

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude of this vector denoted ∇f, where

$$\nabla f = \text{mag}(\nabla f) = \left[G_x^2 + G_y^2\right]$$

The direction of gradient vector is given as.

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

→ The direction of an edge at (x,y) is perpendicular to the direction of the gradient vector at that point.

→ For 3X3 area the simplest way to implement a first order partial derivative at point z_5 is using the Roberts gradient operators

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0
0	1

0	-1
1	0

Roberts

$G_x = z_9 - z_5$ $\quad G_y = z_8 - z_6$

Mask of size 2X2 are awkward to implement, because they do not have a clear center. So we prefer 3X3 mask. An approach using 3X3 masks is given by.

G_x

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

G_y

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

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$
$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

} Prewitt operators

The masks shown above are prewitt operators.

A slight variation of the above equation using a weight of 2 in the center co-efficient,

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

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

→ Sobel operators.

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

A weight value of 2 is used to achieve smoothing. The operators are called the Sobel operators.

→ The prewitt masks are simpler to implement.

→ Sobel masks have slightly superior noise suppression characteristics

→ An approach used frequently is to approximate the gradient by absolute values

$$\nabla f = |G_x| + |G_y|$$

→

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

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

Prewitt

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

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

Sobel

Fig Prewitt & Sobel masks for detecting diagonal edges.

* The Laplacian :-

The laplacian of a 2D function $f(x,y)$ is a second-order derivative defined as

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

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

→ Laplacian Masks.

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$
$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8 + z_9)$$

→ The Laplacian is generally not used in its original for edge detection for several reasons like sensitive to noise, produces double edges, unable to detect edge direction.

→ So the Laplacian is combined with smoothing as a precursor, to find edges via zero crossing.

Consider the function,

$$h(r) = -e^{\frac{-r^2}{2\sigma^2}} \quad \rightarrow (1)$$

where $r^2 = x^2+y^2$ & σ is standard deviation.

Convolving this function with an image, blurs the image with degree of blurring determined by value of σ.

The Laplacian of h is,

$$\nabla^2 h(r) = -\left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{\frac{-r^2}{2\sigma^2}} \quad \rightarrow (2)$$

This function is referred to as Laplacian of Guassian (LOG) because the eq^n (1) is in the form of a Guassian function.

* More Advanced Techniques for Edge detection

The Marr Hildreth Edge Detector

Marr & Hildreth argued that,

(1) Intensity changes are not independent of image scale & so their detection requires the use of operators of different size, &

(2) A sudden intensity change will give rise to a peak or trough in the first derivative or equivalently zero crossing in the second derivative.

Marr & Hildreth argued that the most satisfactory operator fulfilling these conditions is the filter $\nabla^2 G$ where ∇^2 is the Laplacian operator, $(\delta^2/\delta x^2 + \delta^2/\delta y^2)$, & G is the 2D Gaussian function.

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

To find an expression for $\nabla^2 G$ we perform following differentiations.

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

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

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

Collecting terms gives the final expression:

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \rightarrow \text{eq}^n (1)$$

This eq^n is Laplacian of Gaussian.

→ The Marr-Hildreth algorithm consists of convolving the LoG filter with an input image, $f(x,y)$,

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

→ And then finding the zero crossings of $g(x,y)$ to determine the locations of edges in $f(x,y)$.

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

indicating that we can smooth the image first with a guassian filter & then compute the Laplacian of the result. The steps can be summarised as follows,

1. Filter the input image with an NXN Gaussian Low pass filter obtained by sampling.
2. Compute the laplacian of the image
3. Find the zero crossings of the image.

---

* The Canny edge detector

Canny's approach is based on three basic objectives.

1. Low error rate
2. Edge points should be well localised
3. Single edge point response.

Aig. → Let $f(x,y)$ denote the input image & $G(x,y)$ denote the Gaussian function.

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

A smoothed image can be formed by convolving $f_s(x,y)$ by G. → $f_s(x,y)$ is input image

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

→ This operation is followed by computing the gradient's magnitude & direction using,

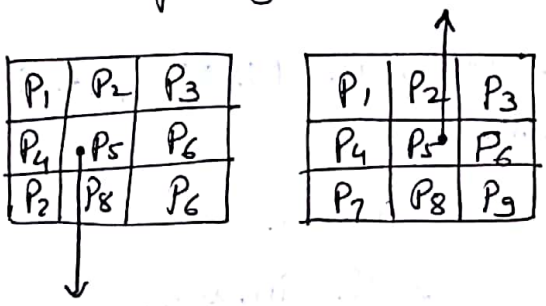
$$M(x,y) = \sqrt{g_x^2 + g_y^2}$$

&

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

→ Perform non maxima suppression.

Specify a number of discrete orientations,



→ Two possible orientations of an horizontal edge in a 3x3 neighborhood.

Non maxima suppression steps :-

1) Find the direction d_k that is closest to $\alpha(x,y)$.
2) If the value of $M(x,y)$ is less than at least one of its two neighbors along d_k, let $g_N(x,y) = 0$ (suppression); otherwise, let $g_N(x,y) = M(x,y)$. where $g_N(x,y)$ is the non maxima suppressed image.

→ The final operation is to threshold $g_N(x,y)$ to reduce false edge points.

Summarising, the Canny edge detector,

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

* Edge Linking & Boundary Detection

The methods discussed till now yield pixels only on edges.

→ But due to noise, breaks in the edges form non uniform illumination & other effects that introduce intensity discontinuities

→ To remove these intensity discontinuities the edge detection algorithms are followed by linking procedures to assemble edge pixels into meaningful edges.

Local processing:-

One of the simplest approaches for linking edge point is to analyse the characteristics of pixels in a small neighborhood (say 3x3 or 5x5) about every point (x,y) in an image that has been labelled as edge point.

→ All points that are similar according to a set of predefined criteria are linked forming an edge of pixels that share those criteria.

* The two principal properties, used for establishing similarity of edge pixels.

(1) The strength of the response of the gradient operator used to produce the edge pixel (magnitude)

(2) The direction of the gradient vector. (angle)

→ From first property the given values are of ∇f,

→ Thus an edge pixel with co-ordinates (x_0, y_0) in a predefined neighborhood of (x,y) is similar in magnitude to the pixel at (x,y) if,

$$|\nabla f(x,y) - \nabla f(x_0,y_0)| \leq E$$

where E is non negative threshold.

→ The direction (angle) of the gradient vector is given by,

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

→ An edge pixel at (x_0,y_0) in the predefined neighborhood of (x,y) has an angle similar to the pixel at (x,y) if,

$$|\alpha(x,y) - \alpha(x_0,y_0)| < A$$

where A is non negative angle threshold.

→ A point in the predefined neighborhood of (x_0,y_0) is linked to the pixel at (x,y) if both magnitude & direction criteria are satisfied.

This process is repeated at every location in the image.

Local processing involves the following steps,

1. Compute the gradient magnitude & angle arrays $M(x,y)$ & $\alpha(x,y)$ of the input image, $f(x,y)$.

2. Form a binary image, g, whose value at any pair of co-ordinates (x,y) is given by,

$$g(x,y) = \begin{cases} 1 & \text{if } M(x,y) > T_M \text{ \& } \alpha(x,y) = A \pm T_A \\ 0 & \text{otherwise.} \end{cases}$$

where T_M is a threshold. A is a specified angle direction & $\pm T_A$ defines a band of acceptable directions.

3. Scan the rows of g & fill (set to 1) all gaps (sets of 0s) in each row that do not exceed a specified length K.

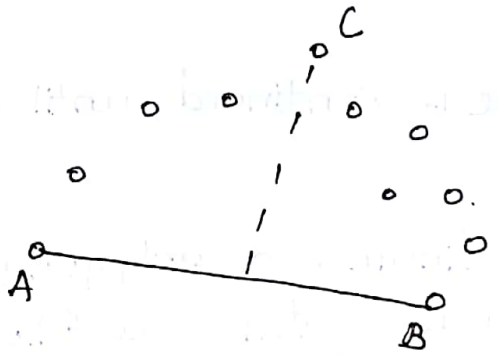
4. To detect gaps in any other direction θ, rotate ~~by~~ g by this angle & apply the horizontal scanning procedure in step 3. Rotate the result back by $-\theta$.

* Regional processing

Regional processing requires points on the boundary of a region to be known.

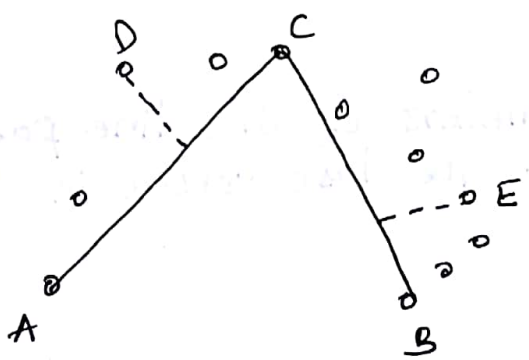
One approach to this type of processing is functional approximation, where we fit a 2-D curve to the known points. Polygonal approximations are particularly attractive because they can capture the essential shape features of a region while keeping the representation of the boundary, relatively simple.

Figure below shows a set of points representing an open curve in which the end points have been labeled A & B.



These two points are by definition vertices of a polygon. We begin by computing the parameters of a straight line passing through A & B. Then, we compute the perpendicular distance from all other points in the curve to this line & select the point that yielded maximum distance. If distance exceeds a specified threshold, T, the corresponding point, labeled 'C' in above fig is declared as vertex.

→ Lines from A to B & from C to B are established, & distances from all points between A & C to line AC are obtained.



The point corresponding to the maximum distance is declared as vertex D, if distance exceeds T; otherwise no new vertices are declared for that segment, as shown in the previous figure.

→ A similar procedure is applied for points between B & C.

→ Fig 3 shows next step.

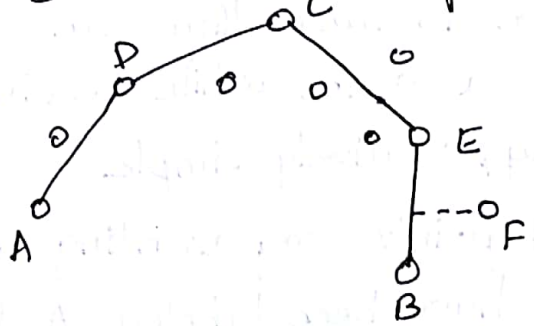


fig 3

→ This iterative procedure is continued until no points satisfy threshold test.

* An algorithm for finding a polygonal fit to open & closed curves may be stated as follows:

1. Let P be a sequence of ordered, distinct, 1-valued ~~object~~ points of a binary image. Specify two starting points. A & B. These are the two starting vertices of the polygon.

2. Specify a threshold T, & two empty stacks OPEN & CLOSED

3. If the points in P corresponds to a closed curve, put A into OPEN & put B into OPEN & into CLOSED. If the points correspond to an open curve, put A into OPEN & B in closed.

4. Compute the parameters of the line passing from last vertex in CLOSED to the last vertex in OPEN

5. Compute the distances from the line in step 4 to all points in P. Select the point, V_{max}, with the maximum distance, D_{max}.

6. If $D_{max} > T$, place V_{max} at the end of the OPEN stack as a new vertex. Go to step 4.

7. Else, remove the last vertex from OPEN & insert it as the last vertex of CLOSED.

8. If OPEN is not empty, go to step 4.

9. Else Exit. The vertices in CLOSED are the vertices of the polygonal fit to the points in P.

---

*Global processing using the Hough Transform

Global processing is the approach that uses or work with an entire edge image.

Hough transform: is the way of finding edge points in an image that lie along a straight line.

~~Method~~:

Given 'n' points in an image, suppose that we want to find subsets of these points that lie on straight lines.

→ First find all lines determined by every pair of points and then find all the subsets of points that are close to particular lines.

→ The problem with this procedure is that, it involves finding $n(n-1)/2 \approx n^2$ lines & then performing $\frac{n((n-1)n)}{2} \approx n^3$ comparison of every point to all lines.

(P.T.O)

Method:-

Hough proposed an alternative approach commonly refered to as hough transform.

→ Consider a point (x_i, y_i)

The general equation of straight line in slope-intercept form $\boxed{y_i = ax_i + b}$ → (1)

→ Infinite lines will pass through (x_i, y_i) satisfying $y_i = ax_i + b$ for varying values of a & b.

→ Writing eqn (1) as $\boxed{b = -x_i a + y_i}$ → (2) (using ab-plane also called as parameter space rather than xy plane).

→ Consider a second point (x_j, y_j) also has a line in parameter space associated with it & this line intersects the line associated with (x_i, y_i) at (a', b') where a' is the slope & b' is the intercept, of the line containing both (x_i, y_i) & (x_j, y_j) in the xy-plane.

→ The computational attractiveness of the Hough transform arises from subdividing the parameter space into so-called accumulator cells

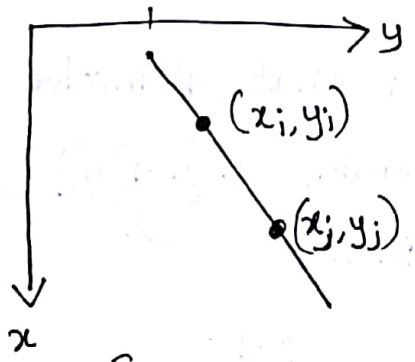


Fig:- xy plane

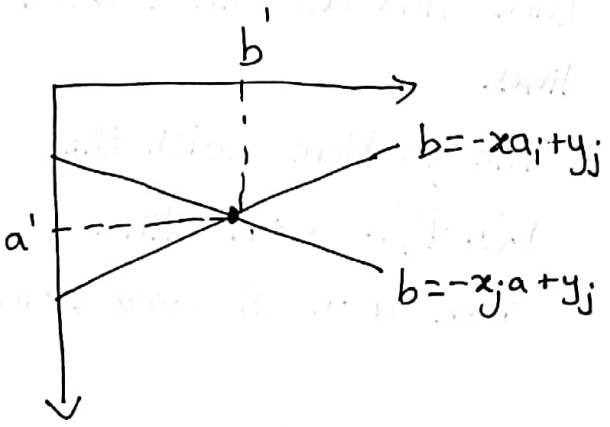


Fig: parameter space.

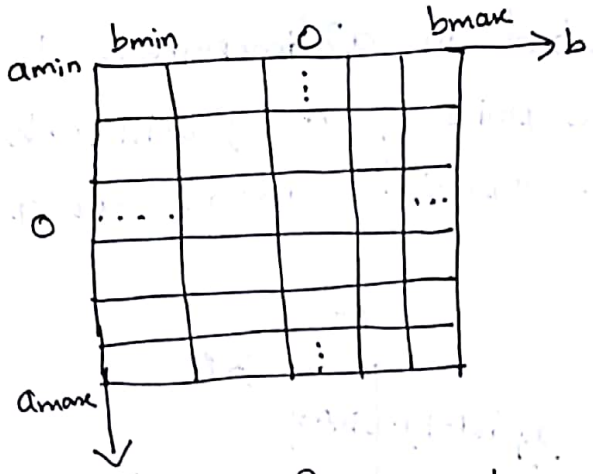


Fig :- Subdivision of parameter space for use of Hough Transform.

→ A problem with using the eq^n $y = ax + b$ to represent a line is that the slope approaches infinity as the line approaches the vertical.

→ One way to solve this difficulty is to use the normal representation of line

$$x \cos\theta + y \sin\theta = \rho$$

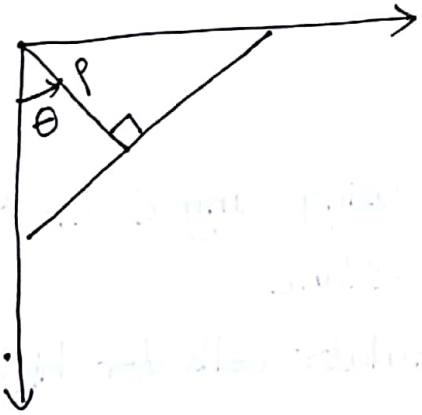


Fig a

a. Normal representation of a line

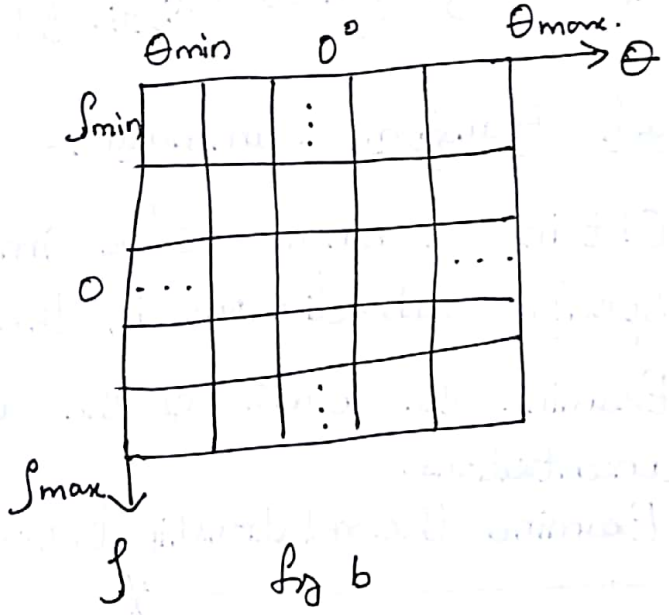


Fig b

b. Subdivision of the $\rho\theta$-plane

→ The Hough transform consists of finding all pairs of values of θ & ρ which satisfy the equations, that pass through (x, y).

(P.T.O)

→ These are accumulated in a 2dimensional histogram

→ When plotted these pairs of θ & ρ will look like a sine wave. The process is repeated for all appropriate (x,y) locations.

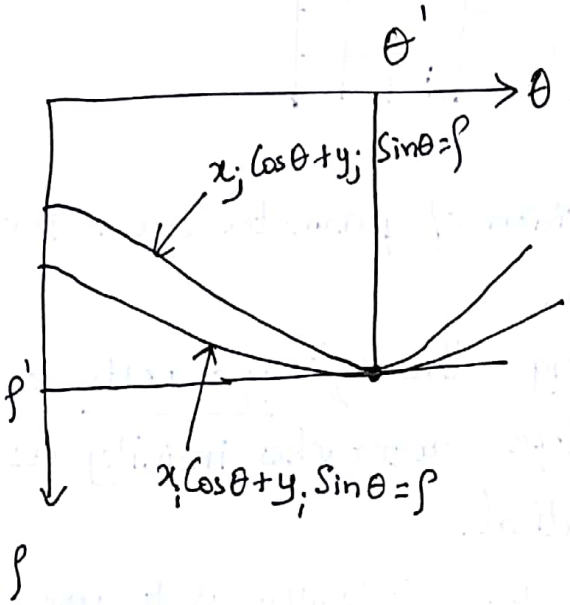


Fig c. Sinusoidal curves in ρθ plane.

→ The point of intersection (ρ', θ') in fig c corresponds to the line passing through points (x_i, y_i) & (x_j, y_j) in the xy plane.

* Hough transform Summary :-

1. Obtain a binary edge image using any of the techniques
2. Specify sub-divisions in the ρθ-plane
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship between pixels in a chosen cell.

Imp Questions :-

* Process of: Isolated Point detection, Line Detection, Marr-Hildreth Method, Canny Edge detector

* Process of :- Local processing, Regional processing & Global processing using Hough transform.