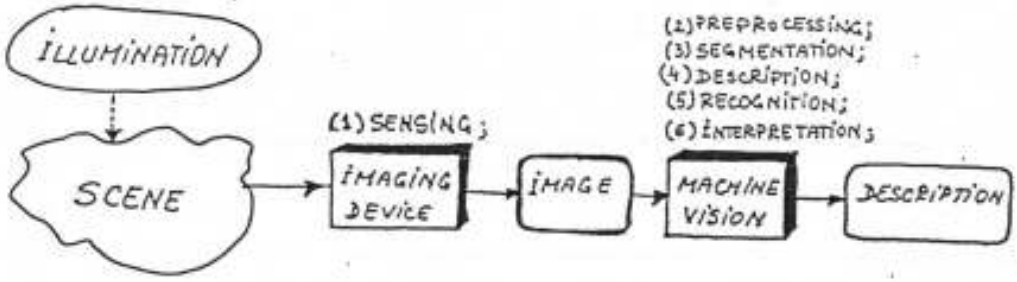
MACHINE VISION

THE PURPOSE OF A MACHINE VISION SYSTEM IS TO PRODUCE A SYMBOLIC DESCRIPTION OF WHAT IS BEING IMAGED.



(1) SENSING : THE PROCESS THAT YIELDS A VISUAL IMAGE;

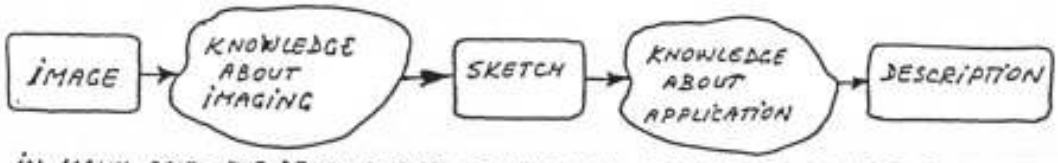
(2) PREPROCESSING : TECHNIQUES SUCH AS NOISE REDUCTION AND ENHANCEMENT OF DETAILS;

(3) SEGMENTATION: THE PROCESS THAT PARTITIONS AN IMAGE INTO OBJECTS OF INTEREST;

(4) DESCRIPTION : COMPUTATION OF FEATURES (e.g. SIZE, SHAPE) SUITABLE FOR DIFFERENTIATING ONE TYPE OF OBJECT FROM ANOTHER;

(5) RECOGNITION : THE PROCESS THAT IDENTIFIES THE OBJECTS (e.g. WRENCH, BOLT);

(6) INTERPRETATION: ASSIGNS MEANING TO AN ENSEMBLE OF RECOGNIZED OBJECTS;



IN MANY CASES, THE DEVELOPMENT OF A SYMBOLIC DESCRIPTION OF A SCENE FROM ONE ORE MORE IMAGES CAN BE BROKEN DOWN CONVENIENTLY INTO TWO STAGES. THE FIRST STAGE IS LARGELY GOVERNED BY OUR UNDERSTANDING OF THE IMAGE-FORMATION PROCESS; THE SECOND DEPENDS MORE ON THE NEEDS OF THE INTENDED APPLICATION.

USEFUL IMAGE FILTERING OPERATIONS

▶ AVERAGING

EACH PIXEL $f(i,j)$ IS REPLACED BY A WEIGHTED AVERAGE OF ITS NEIGHBOURING PIXELS:

$$f^*(i,j) = \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} a(k,l) \cdot f(i+k, j+l)$$

	$j-1$	j	$j+1$
$i-1$	1/9	1/9	1/9
i	1/9	1/9	1/9
$i+1$	1/9	1/9	1/9

3-BY-3 MASK

	$j-1$	j	$j+1$
$i-1$	0	1/8	0
i	1/8	1/2	1/8
$i+1$	0	1/8	0

5-POINT AVERAGE MASK

AVERAGING IS USED FOR NOISE SMOOTHING, LOW-PASS FILTERING, AND SUBSAMPLING.

▶ MEDIAN FILTERING

EACH PIXEL $f(i,j)$ IS REPLACED BY THE MEDIAN VALUE OF THE 9 PIXELS IN THE 3-BY-3 WINDOW:

$$f^*(i,j) = \text{MEDIAN}\{ f(i-1,j-1), f(i-1,j), f(i-1,j+1), f(i,j-1), f(i,j), f(i,j+1), f(i+1,j-1), f(i+1,j), f(i+1,j+1) \}$$

- THE MEDIAN VALUE OF A GIVEN SET IS SUCH THAT HALF THE VALUES IN THE SET ARE SMALLER THAN THAT MEDIAN AND ANOTHER HALF ARE GREAT THAN THE MEDIAN

- MEDIAN FILTERING FORCES PIXELS WITH VERY DISTINCT INTENSITIES TO BE MORE LIKE THEIR NEIGHBOURS, THUS ELIMINATING INTENSITY SPIKES THAT APPEAR ISOLATED IN THE WINDOW.

▶ MAGNIFICATION AND INTERPOLATION

- **REPLICATION** : "ZERO-ORDER HOLD" WHERE EACH PIXEL ALONG A SCAN LINE IS REPEATED ONCE AND THEN EACH SCAN LINE IS REPEATED.

  THE M-BY-N IMAGE IS INTERLACED BY ROWS AND COLUMNS OF ZEROS TO OBTAIN A $2M$-BY-$2N$ MATRIX WHICH IS THEN CONVOLVED WITH A MASK $H = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

- **LINEAR INTERPOLATION** : "FIRST-ORDER HOLD". THE $2M$-BY-$2N$ ZERO-INTERLACED IMAGE IS CONVOLVED WITH THE MASK : $H = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 5 & 6 \end{bmatrix} \xrightarrow{\text{ZERO INTERLACED}} \begin{bmatrix} 1 & 0 & 3 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 5 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{*H} \begin{bmatrix} 1 & 2 & 3 & 2.5 & 2 & 1 \\ 2.5 & 3.25 & 4 & 4 & 4 & 2 \\ 4 & 4.5 & 5 & 5.5 & 6 & 3 \\ 2 & 2.25 & 2.5 & 2.75 & 3 & 1.5 \end{bmatrix}$$

IMAGE SEGMENTATION

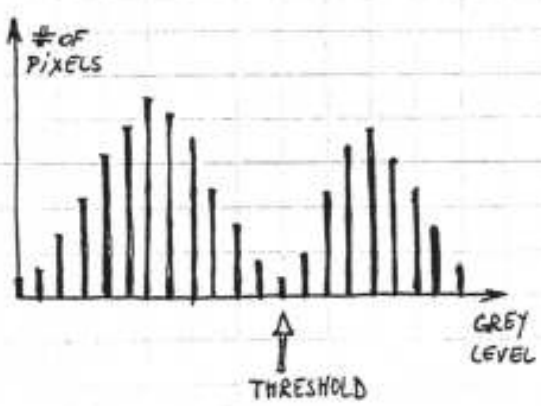
- PARTITIONING AN IMAGE INTO REGIONS OF INTEREST, MEANINGFULLY LABELLED

	PARALLEL	SEQUENTIAL
REGION PARTITIONING BASED ON "SIMILARITY"	THRESHOLDING	BLOB GROWING
REGION PARTITIONING BASED ON "DISCONTINUITY"	EDGE FINDING	EDGE TRACKING

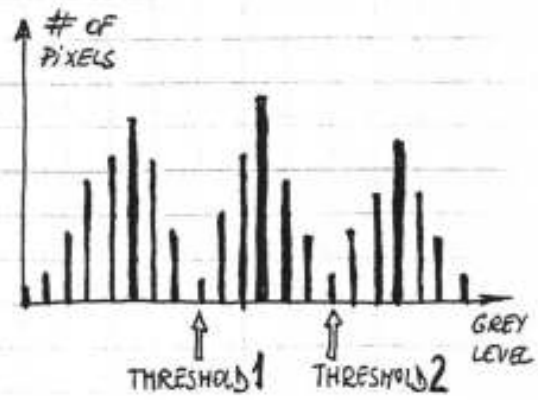
1. THRESHOLDING

GREY SCALE IMAGE → THRESHOLDING → BINARY IMAGE → FURTHER ANALYSIS

THE THRESHOLD LEVEL CAN BE SET TO DISCRIMINATE BETWEEN THE OBJECTS IN THE FOREGROUND AND THE BACKGROUND



THRESLOBING IN A BIMODAL HISTOGRAM



MULTILEVEL HISTOGRAM

- INCORRECT THRESHOLD VALUE → BINARY IMAGE WITH RAGGED EDGES

1▶ BLOB GROWING [McFerrow]

Algorithm 10.2 Growing a blob array

```
[Uses a pattern window to scan image array
 — value of pattern used to look up action table (Figure 10.61)
 — pattern implements an 8 connected background
   and a 4 connected foreground algorithm]
Allocate an m×n blob array the same size as the image array.
Initialize blob array to zero.
[assume that the image array includes a surrounding
 rectangle of background, at least one pixel wide]
Set the next blob descriptor to 1
For each row 1 to m in the image do
   For each column 1 to n in the image do
        Calculate the Window State [Figure 10.60].
        Use the Window State to select an action [Figure 10.61].
        Execute the action.
        Update the blob array.
   [If  blob statistics are being calculated on the fly calculate them here]
        If a new blob started then
            increment the next blob descriptor
        End
   End
End
```

BINARY WINDOW PATTERN

P_3	P_2
P_4	P_1

P_1 = target pixel

WINDOW STATE $= P_1 + 2^1 \cdot P_2 + 2^2 \cdot P_3 + 2^3 \cdot P_3$

WINDOW STATE	WINDOW PATTERN	ACTION
0	0 0 / 0 0	1 Connect to background blob
1	0 0 / 0 1	2 Start new object blob
2	0 1 / 0 0	1 Connect to background blob
3	0 1 / 0 1	3 Connect to object blob
4	1 0 / 0 0	4 End of object blob, connect to background blob, and merge background blobs
5	1 0 / 0 1	5 Start a new object blob and end old object blob (4-CONNECTED FOR OBJECTS)
6	1 1 / 0 0	1 Connect to background blob
7	1 1 / 0 1	3 Connect to object blob
8	0 0 / 1 0	1 Connect to background blob
9	0 0 / 1 1	3 Connect to ~~background~~ object blob
10	0 1 / 1 0	1 Connect to background blob (8-CONNECTED RULE FOR BACKGROUND)
11	0 1 / 1 1	6 End of background blob, connect to object blob and merge object blobs
12	1 0 / 1 0	1 Connect to background blob
13	1 0 / 1 1	3 Connect to object blob
14	1 1 / 1 0	7 Start a new background blob
15	1 1 / 1 1	3 Connect to object blob

Fig. 10.61 PATTERN LOOK UP TABLE FOR 8-CONNECTED BACKGROUND AND 4-CONNECTED OBJECTS, USED IN ALGORITHM 10.2

BLOB GROWING

Algorithm Grow [SNYDER]

This algorithm implements region growing by using a push down stack on which to temporarily keep the coordinates of pixels in the region.

1. Find an unlabeled black pixel; that is, $M(x, y) = 0$. Choose a new label number for this region, call it N. If all pixels have been labeled, stop.
2. If $P(x - 1, y)$ is black and $M(x - 1, y) = 0$, push $(x - 1, y)$ onto the stack.

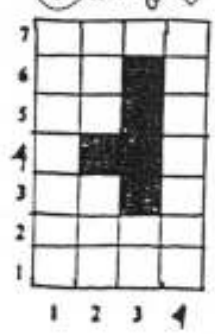
   If $P(x + 1, y)$ is black and $M(x + 1, y) = 0$, push $(x + 1, y)$ onto the stack.

   If $P(x, y - 1)$ is black and $M(x, y - 1) = 0$, push $(x, y - 1)$ onto the stack.

   If $P(x, y + 1)$ is black and $M(x, y + 1) = 0$, push $(x, y + 1)$ onto the stack.
3. $M(x, y) \leftarrow N$.
4. Choose a new (x, y) by popping the stack.
5. If the stack is empty, go to 1, else go to 2.

M labelled array

P binary image array



This labeling operation results in a set of connected regions, each assigned a unique label number. To find the region to which any given pixel belongs, the computer has only to interrogate the corresponding location in the M memory and read the region number.

2) ..P(x,y-1)= = BLACK; M(x,y-1)=0; PUSH <x,y-1>= =<3,3> ONTO THE STACK

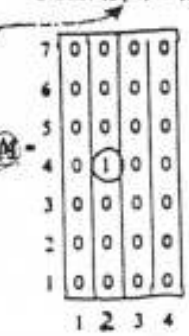
Example 13.3 Applying Region Growing

The figure below shows a 4 X 7 array of pixels. Assume the initial value of (x, y) is (2, 4). Apply algorithm "grow" and show the contents of the stack and M each time step 3 is executed. Let the initial value of N be 1.

Solution:

Pass 1: Immediately after execution of step 3.

Stack: (3, 4) ← top

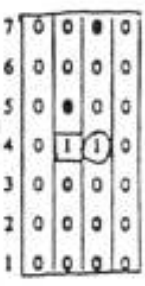


<x,y>=<2,4>
N=1

2)...P(x+1,y)=BLACK; M(x+1,y)=0; PUSH <x+1,y>= =<3,4> ONTO THE STACK;

Pass 2:

Stack: (3, 5) ← top
(3, 3)

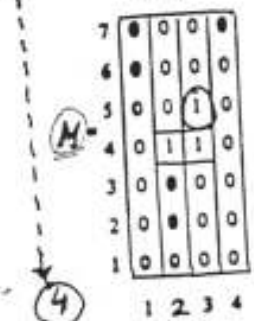


4 → <x,y>=<3,4>

2).... P(x,y+1)=BLACK; M(x,y+1)=0; PUSH <x,y+1> ONTO THE STACK;

Pass 3:

Stack: (3, 6) ← top
(3, 5)
(3, 3)



4 → <x,y>=<3,5>

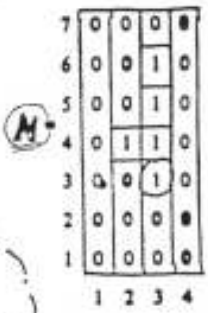
Pass 4:

Stack: (3, 3) ← top

4 → <x,y>=<3,6>

Pass 5:

Stack: Empty



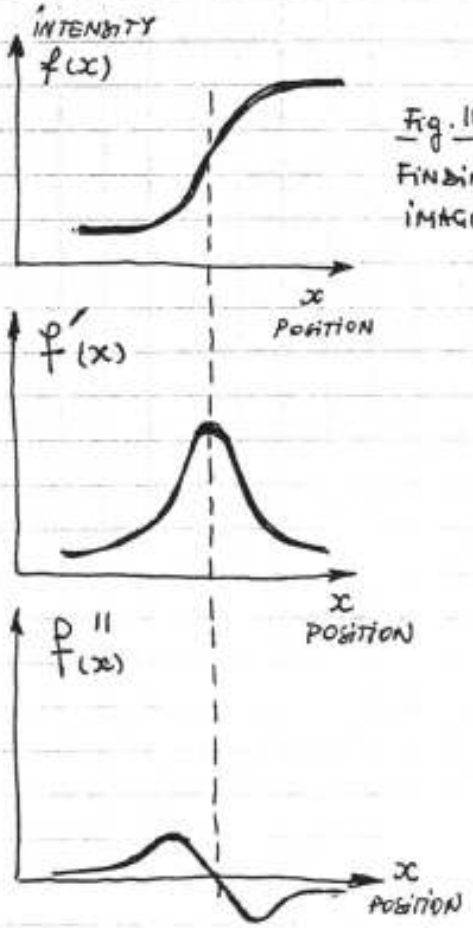
▶ EDGE FINDING

- IS THE PROCESS OF LOCATING THE BOUNDARIES BETWEEN REGIONS AND IMAGE.
- FINDING EDGES REDUCES THE IMAGE TO A SERIES OF LINES, LIKE A WIRE FRAME MODEL
- BASIC ASSUMPTION → OBJECTS IN IMAGE ARE CHARACTERIZED BY NEAR UNIFORM-INTENSITY REGIONS (SHADOWS CAN RESULT IN FALSE EDGES)

EDGE FINDING: (A TWO-STEP PROCESS)

(1) FIND PIXELS IN THE IMAGE THAT ARE LIKELY TO BE ON AN EDGE ← IMAGE GRADIENT

(2) JOIN THE EDGE PIXELS TOGETHER IN A COHERENT LINE ← HOUGH TRANSFORM (IF WE KNOW WHAT TYPE OF EDGE WE ARE LOOKING FOR (STRAIGHT, CIRCULAR, ...).



* IMAGE GRADIENT

Fig. 10.71 [McKetrow]
FINDING THE EDGE POINT IN AN IMAGE BY TAKING THE FIRST- AND SECOND-ORDER DERIVATIVES OF THE INTENSITY FUNCTION IN THE x DIRECTION

IMPORTANT PROPERTIES FOR GRADIENT OPERATORS:

(1) ACCURACY IN ESTIMATING GRADIENT AMPLITUDE;
(2) ACCURACY IN ESTIMATING GRADIEN ORIENTATION;
(3) ACCURACY IN ESTIMATING STEP EDGE CONTRAST;
(4) ACCURACY IN ESTIMATING STEP EDGE DIRECTION;

- FINDING THE IMAGE GRADIENT USING THE "SOBEL" OPERATOR

	$x-1$	x	$x+1$
$y-1$	-1	0	1
y	-2	0	2
$y+1$	-1	0	1

"SOBEL" MASK TO FIND VERTICAL EDGES (DUE TO GRADIENT CHANGE IN HORIZONTAL DIRECTION): dx

	$x-1$	x	$x+1$
$y-1$	-1	-2	-1
y	0	0	0
$y+1$	1	2	1

"SOBEL" MASK TO FIND HORIZONTAL EDGES (DUE TO GRADIENT CHANGE IN THE VERTICAL DIRECTION: dy

$$dx = [p(x+1,y-1) + 2\cdot p(x+1,y) + p(x+1,y+1)] - [p(x-1,y-1) + 2\cdot p(x-1,y) + p(x-1,y+1)]$$

$$dy = [p(x-1,y+1) + 2\cdot p(x,y+1) + p(x+1,y+1)] - [p(x-1,y-1) + 2\cdot p(x,y-1) - p(x+1,y-1)]$$

"SOBEL" GRADIENT MAGNITUDE $= \sqrt{dx^2 + dy^2}$

"SOBEL" GRADIENT DIRECTION $= \arctan(dy/dx)$

- IF THE GRADIENT MAGNITUDE AT A GIVEN PIXEL $p(x,y)$ IS BIGGER THAN THE THRESHOLD THAN IT IS CONSIDERED THAT THE PIXEL BELONGS TO AN EDGE
- THE GRADIENT DIRECTION IS USED FOR EDGE ORGANIZATION SELECTION, AND LINKING.

- OTHER EDGE FINDING ALGORITHMS

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

P1

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

P2

◄ "PREWITT" EDGE DETECTOR MASKS

"PREWITT" GRADIENT MAGNITUDE $= \sqrt{P1^2 + P2^2}$

"PREWITT" GRADIENT DIRECTION $= \arctan(P1/P2)$

-1	$-\sqrt{2}$	-1
0	0	0
1	$\sqrt{2}$	1

F1

-1	0	1
$-\sqrt{2}$	0	$\sqrt{2}$
-1	0	1

F2

◄ "FREI & CHEN" GRADIENT MASKS

"FREI & CHEN" GRADIENT MAGNITUDE $= \sqrt{F1^2 + F2^2}$

"FREI & CHEN" GRADIENT DIRECTION $= \arctan(F1/F2)$

► THE CHOICE OF THE "SOBEL", "PREWITT", OR "FREI & CHEN" MASKS FOR EDGE FINDING IS BASED ON A NOISE MODEL.

- COMPASS GRADIENT OPERATORS

COMPASS GRADIEN OPERATORS HAVE MAXIMAL RESPONSE FOR EDGES WITH MORE ORIENTATIONS THAN THE BASIC HORIZONTAL AND VERTICAL ONES.

-3	-3	5
-3	0	5
-3	-3	5

K0

-3	5	5
-3	0	5
-3	-3	-3

K1

5	5	5
-3	0	-3
-3	-3	-3

K2

5	5	-3
5	0	-3
-3	-3	-3

K3

5	-3	-3
5	0	-3
5	-3	-3

K4

-3	-3	-3
5	0	-3
5	5	-3

K5

-3	-3	-3
-3	0	-3
5	5	5

K6

-3	-3	-3
-3	0	5
-3	5	5

K7

"KIRSCH" COMPASS MASKS

"KIRSCH" GRADIENT MAGNITUDE $= \max_{i=0,1,\dots,7} K_i$

"KIRSCH" GRADIENT DIRECTION $= \arg(\max K_i) \cdot 45^\circ$

$$R0 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad R1 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad R2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad R3 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$R4 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad R5 = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \quad R6 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad R7 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

"ROBINSON" COMPASS MASKS

"ROBINSON" GRADIENT AMPLITUDE $= \max_{i=0,1,\dots,7} R_i$

"ROBINSON" GRADIENT DIRECTION $= \arg(\max R_i) \cdot 45°$

▷ THE MAJOR PROBLEM WITH GRADIENT EDGE OPERATORS IS THAT THEY GENERALLY PRODUCE NOISY RESULTS ⟹ REDUCE NOISE BY USING NOISE-CLEANING (SMOOTHING) TECHNIQUES: GAUSSIAN FILTERING, MEDIAN FILTERING, ...

▷ A SECOND PROBLEM WITH GRADIENT EDGE OPERATORS IS THAT THEY MAY PRODUCE THICK EDGES (WIDER THAN ONE PIXEL). THIS HAPPENS WHEN EDGES ARE NOT ABRUPT OR WHEN SOME KIND OF RUNNIG-AVERAGE NOISE-CLEANING STEP IS APPLIED PRIOR TO THE EDGE DETECTION.

⇓

INCORPORATE NON MAXIMA SUPPRESSION IN THE EDGE DETECTION

NON MAXIMA SUPPRESSION: CONDITIONS FOR A PIXEL TO BE CONSIDERED AN EDGE PIXEL

NOT ONLY MUST THE GRADIENT MAGNITUDE BE HIGH ENOUGH TO DETECT AN EDGE, BUT AS ONE CROSSES THE PIXEL IN THE DIRECTION OF THE GRADIENT, THE GRADIENT MAGNITUDE OF THE PIXEL MUST BE HIGHER THAN THE GRADIENT MAGNITUDE OF THE PREDEDING AND SUCCEEDING NEICHBORING PIXELS.

↳ ZERO-CROSSING EDGE DETECTORS: THE PLACE WHERE THE FIRST DERIVATIVE OF THE STEP IS MAXIMUM IS EXACTLY THE PLACE WHERE THE SECOND DERIVATIVE HAS A ZERO-CROSSING.

➤ THE ISOTROPIC GENERALIZATION OF THE SECOND ORDER DERIVATIVE IN 2-D IS THE "LAPLACIAN"

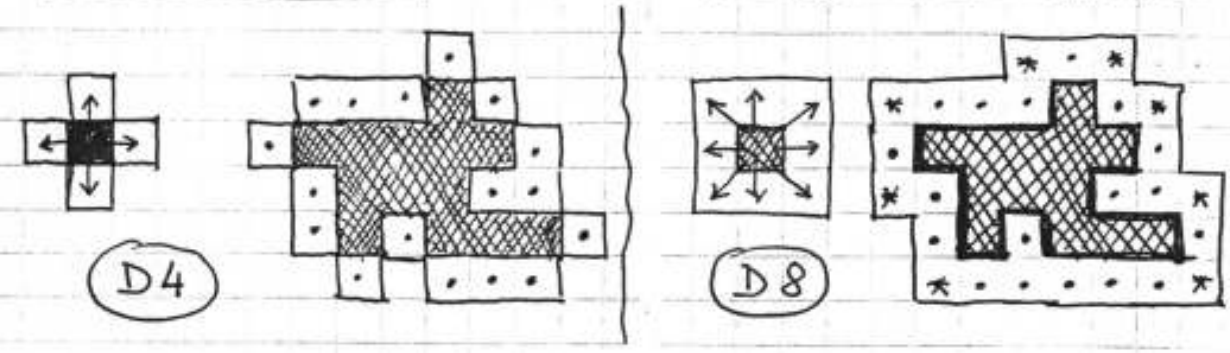
$$L[f(x,y)] = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{3} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

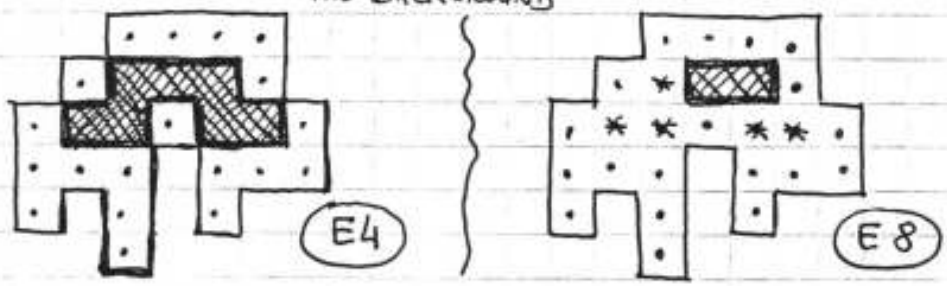
TWO COMMON "LAPLACIAN" MASKS

▶ EDGE (CONTOUR) FINDING IN BINARY IMAGES USING BINARY NEIGHBOURHOOD OPERATORS

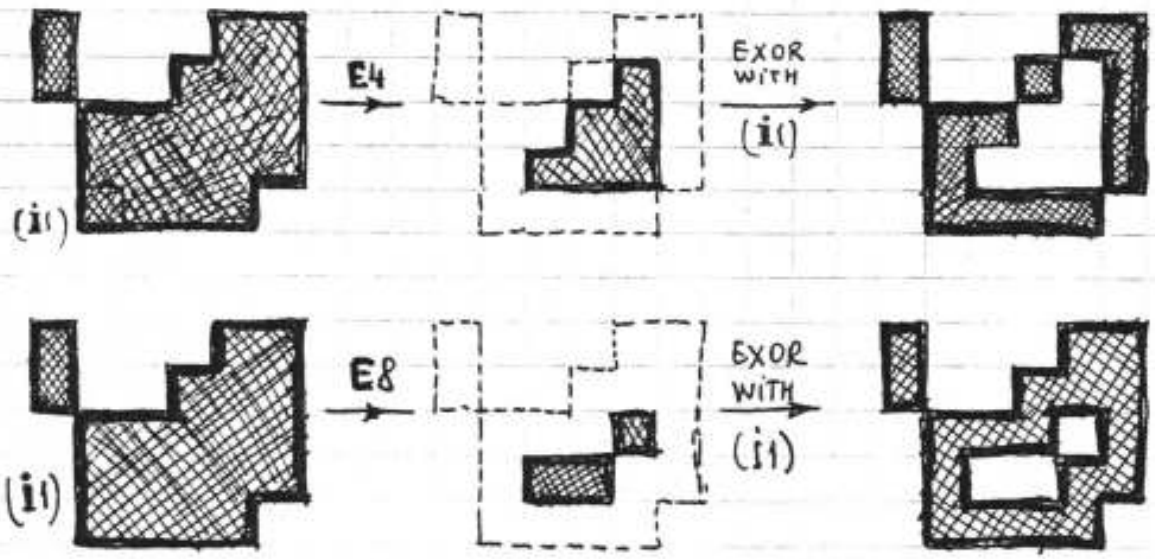
- **DILATION (EXPANSION)** EXPANDS EACH OBJECT PIXELS WITH ITS NEIGHBOURS



- **EROSION (SHRINKING)** REMOVES EACH OBJECT PIXEL HAVING A NEIGHBOUR IN THE BACKGROUND



- **CONTOUR FINDING**



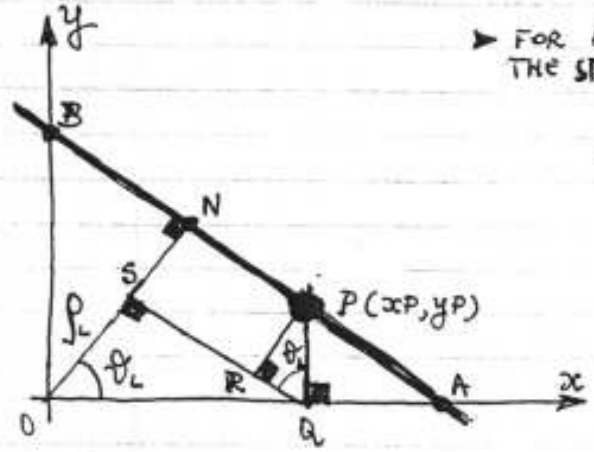
"HOUGH" TRANSFORM

- IT IS A FAST ALGORITHM TO DETECT LINES IN IMAGES

- BASIC IDEAS: (i) A STRAIGHT LINE DEFINED IN THE CARTESIAN SPACE (x, y) BY TWO PARAMETERS (LET'S SAY ρ = THE DISTANCE FROM THE ORIGIN TO THE LINE, AND ϑ = THE ANGLE OF THE NORMAL FROM THE ORIGIN TO THE LINE) IS REPRESENTED BY A UNIQUE POINT IN THE PARAMETER PLANE (ϑ, ρ)

  (ii) A POINT IN THE CARTESIAN SPACE (x, y) IS REPRESENTED BY A CURVE IN THE PARAMETER PLANE (ϑ, ρ)

→ (iii) IF WE HAVE MORE POINTS LAYING ON THE SAME STRAIGHT LINE IN THE CARTESIAN SPACE (x, y), THEN ALL THE CURVES WHICH REPRESENT THESE POINTS IN THE PARAMETER SPACE (ϑ, ρ) INTERSECT IN A UNIQUE POINT THAT REPRESENTS THAT STRAIGHT LINE IN THE PARAMETER SPACE.



▶ FOR A GIVEN POINT $P(x_P, y_P)$ LAYING ON THE STRAIGHT LINE AB:

$$\rho_L = \overline{ON} = \overline{OS} + \overline{SN} = \overline{OQ} \cdot \cos\vartheta_L + \overline{PR} = \overline{OQ} \cdot \cos\vartheta_L + PQ \cdot \sin\vartheta_L = x_P \cdot \cos\vartheta_L + y_P \cdot \sin\vartheta_L$$

▶ THE FOUR CURVE EQUATIONS REPRESENTING THE POINTS A, P, N AND B WHICH LAY ON THE SAME STRAIGHT LINE IN THE CARTESIAN SPACE ARE:

$$\rho_L = x_A \cdot \cos\vartheta_L + y_A \cdot \sin\vartheta_L$$

$$\rho_L = x_P \cdot \cos\vartheta_L + y_P \cdot \sin\vartheta_L$$

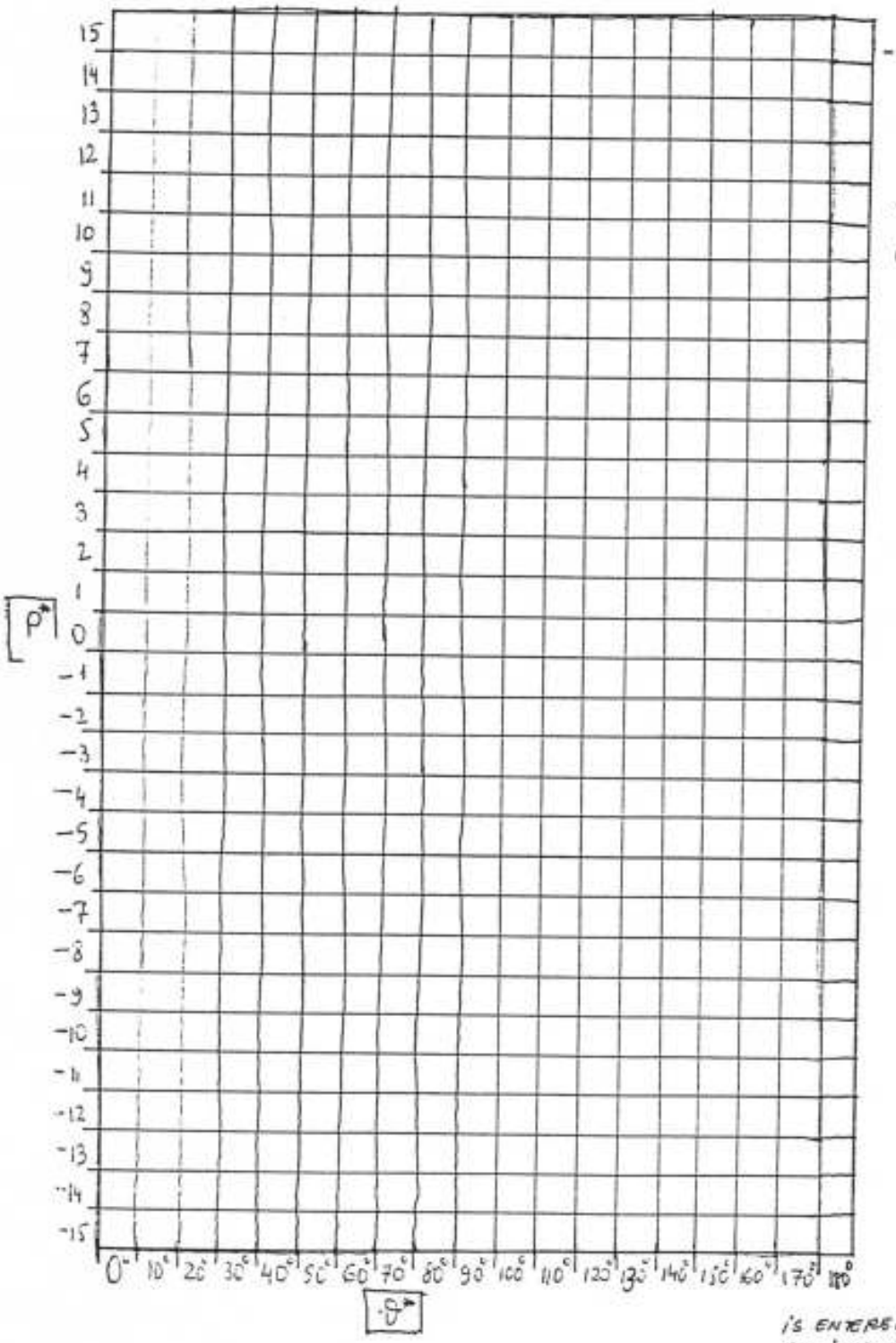
$$\rho_L = x_N \cdot \cos\vartheta_L + y_N \cdot \sin\vartheta_L$$

$$\rho_L = x_B \cdot \cos\vartheta_L + y_B \cdot \sin\vartheta_L$$

→ ALL THESE EQUATIONS INTERSECT IN THE SAME POINT (ϑ_L, ρ_L) IN THE PARAMETER SPACE (ϑ, ρ)

AN ACCUMULATOR ATTACHED TO EACH COORDINATE (ϑ, ρ) IN THE PARAMETER SPACE WILL COUNT ALL THE CURVES WHICH INTERSECT IN THAT COORDINATE

ACCUMULATOR ARRAY FOR THE HOUGH TRANSFORMATION USING NORMAL (ρ, ϑ) PARAMETERS FOR LINES



$-15 \leq \rho \leq 15$, $\Delta\rho = 1$

$0° \leq \vartheta \leq 180°$, $\Delta\vartheta = 10°$

- FOR A GIVEN POINT (x_i, y_i) IN THE CARTESIAN SPACE THERE IS A SINUSOIDAL CURVE IN THE PARAMETER SPACE:

$$\rho = x_i \cdot \cos\vartheta + y_i \cdot \sin\vartheta$$

- FOR m POINTS LYING ON THE SAME STRAIGHT LINE IN THE CARTESIAN SPACE THERE ARE m SINUSOIDAL CURVES WHICH HAVE A COMMON POINT OF INTERSECTION (ρ_k, ϑ_k) IN THE PARAMETER SPACE

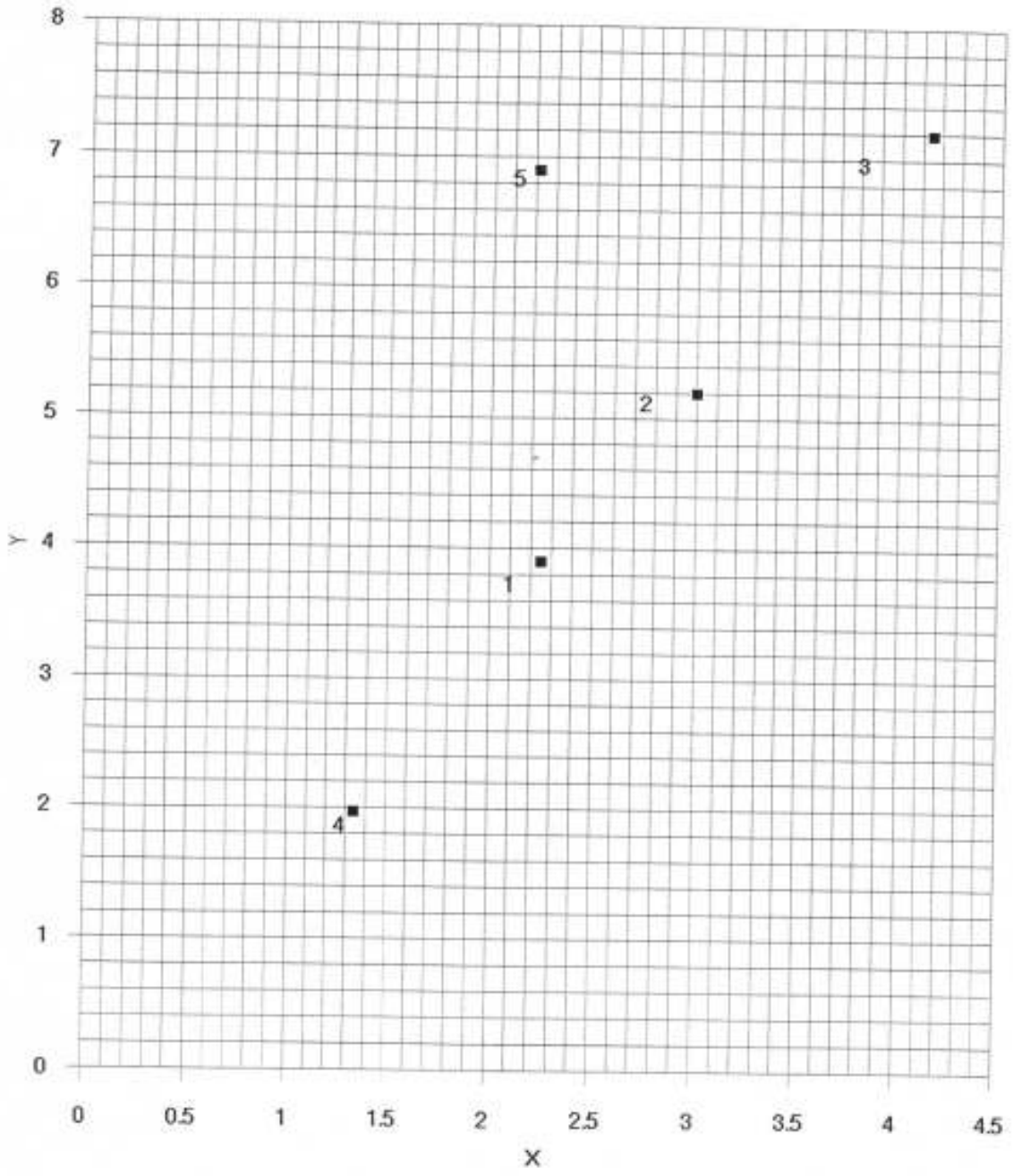
$$\rho_k = x_i \cdot \cos\vartheta_k + y_i \cdot \sin\vartheta_k$$

FOR $i = 1, 2, \ldots, m$

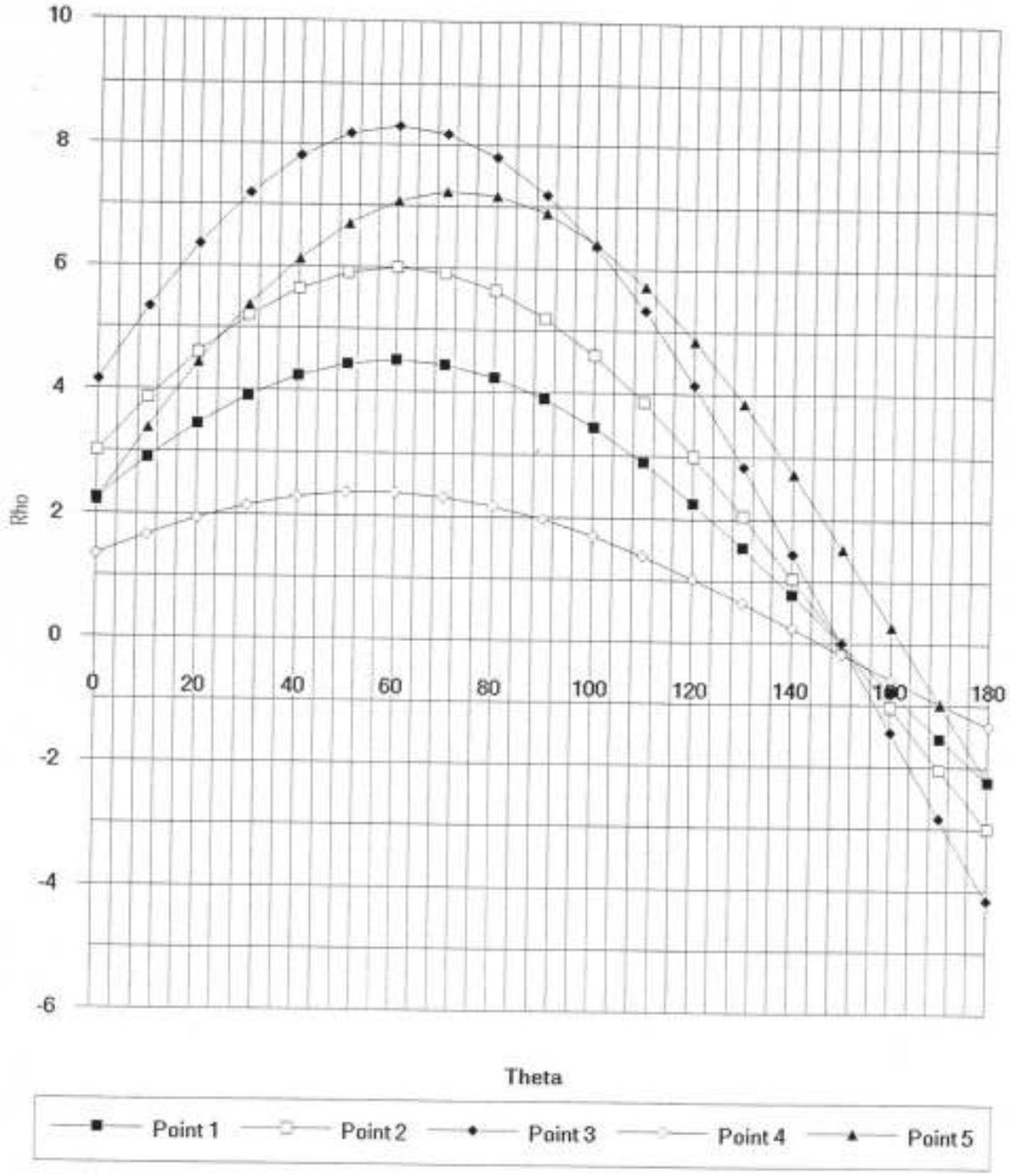
- FOR EACH POINT (x_i, y_i) IN THE CARTESIAN SPACE THE QUANTIZED VERSION OF THE SINUSOIDAL CURVE $\rho^* = x_i \cdot \cos\vartheta^* + y_i \cdot \sin\vartheta^*$ IS ENTERED IN THE ACCUMULATOR ARRAY BY INCREMENTING THE COUNT IN EACH (QUANTIZATION) CELL ALONG THE CURVE.

N.B. ρ^* AND ϑ^* ARE THE QUANTIZED VALUES OF THE SINUSOIDAL CURVE

Data Points



Hough Transform of Data Points



Graphics and Image Processing

W. Newman
Editor

Use of the Hough Transformation To Detect Lines and Curves in Pictures

Richard O. Duda and Peter E. Hart
Stanford Research Institute,
Menlo Park, California

Hough has proposed an interesting and computationally efficient procedure for detecting lines in pictures. This paper points out that the use of angle-radius rather than slope-intercept parameters simplifies the computation further. It also shows how the method can be used for more general curve fitting, and gives alternative interpretations that explain the source of its efficiency.

Key Words and Phrases: picture processing, pattern recognition, line detection, curve detection, colinear points, point-line transformation, Hough transformation

CR Categories: 3.63

This work was supported by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NAS12-2221.

1. Introduction

A recurring problem in computer picture proces is the detection of straight lines in digitized image: the simplest case, the picture contains a number of crete, black figure points lying on a white backgro The problem is to detect the presence of groups of linear or almost colinear figure points. It is clear the problem can be solved to any desired degree o curacy by testing the lines formed by all pairs of po However, the computation required for n points is proximately proportional to n^2, and may be prohib for large n.

Rosenfeld [1] has described an ingenious method to Hough [2] for replacing the original problem of ing colinear points by a mathematically equivalent p lem of finding concurrent lines. This method inv transforming each of the figure points into a str line in a *parameter space*. The parameter space is de by the parametric representation used to describe in the picture plane. Hough chose to use the fan slope-intercept parameters, and thus his param space was the two-dimensional slope-intercept p Unfortunately, both the slope and the intercept ar bounded, which complicates the application of the nique. In this note we suggest an alternative param zation that eliminates this problem. We also give tw ternative interpretations of Hough's method, or which reveals plainly the source of its efficiency. Fi we show how the method can be extended to find general classes of curves in pictures.

2. Fundamentals

The set of all straight lines in the picture plan stitutes a two-parameter family. If we fix a paramet

tion for the family, then an arbitrary straight line can be represented by a single point in the parameter space. For reasons that become obvious, we prefer the so-called *normal parameterization*. As illustrated in Figure 1, this parameterization specifies a straight line by the angle θ of its normal and its algebraic distance ρ from the origin. The equation of a line corresponding to this geometry is

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

If we restrict θ to the interval $[0, \pi]$, then the normal parameters for a line are unique. With this restriction, every line in the x-y plane corresponds to a unique point in the θ-ρ plane.

Suppose now, that we have some set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n figure points and we want to find a set of straight lines that fit them. We transform the points (x_i, y_i) into the sinusoidal curves in the θ-ρ plane defined by

$$\rho = x_i \cos \theta + y_i \sin \theta. \tag{1}$$

It is easy to show that the curves corresponding to colinear figure points have a common point of intersection. This point in the θ-ρ plane, say (θ_0, ρ_0), defines the line passing through the colinear points. Thus the problem of detecting colinear points can be converted to the problem of finding concurrent curves.

A dual property of the point-to-curve transformation can also be established. Suppose we have a set $\{(\theta_1, \rho_1), \ldots, (\theta_k, \rho_k)\}$ of points in the θ-ρ plane, all lying on the curve

$$\rho = x_0 \cos \theta + y_0 \sin \theta.$$

Then it is easy to show that all these points correspond to lines in the x-y plane passing through the point (x_0, y_0). We can summarize these interesting properties of the point-to-curve transformation as follows:

Property 1. A point in the picture plane corresponds to a sinusoidal curve in the parameter plane.

Property 2. A point in the parameter plane corresponds to a straight line in the picture plane.

Property 3. Points lying on the same straight line in the picture plane correspond to curves through a common point in the parameter plane.

Property 4. Points lying on the same curve in the parameter plane correspond to lines through the same point in the picture plane.

In Section 3 we apply these results to the problem of detecting colinear points in the picture plane and show how significant computational economies can be realized in certain situations.

Fig. 1. The normal parameters for a line.

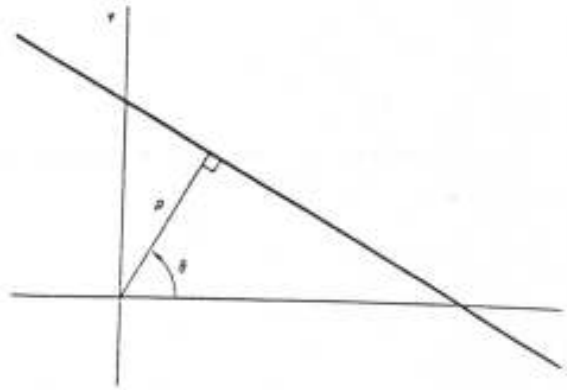
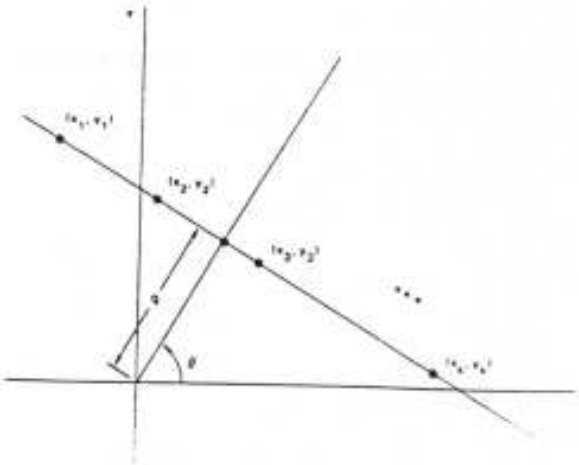


Fig. 2. Projection of colinear points onto a line.



3. Applications and Alternative Interpretations

Suppose we map all of the points in the picture plane into their corresponding curves in the parameter plane. In general, these n curves will intersect in $n(n-1)/2$ points corresponding to the lines between all pairs of

figure points. Exactly colinear subsets of figure points can be found, at least in principle, by finding coincident points of intersection in the parameter plane. Unfortunately, this approach is essentially exhaustive, and the computation required grows quadratically with the number of picture points.

When it is not necessary to determine the lines exactly, the computational burden can be reduced considerably. Following Hough's basic proposal, we specify the acceptable error in θ and ρ and quantize the θ-ρ plane into a quadruled grid. This quantization can be confined to the region $0 \leq \theta < \pi$, $-R \leq \rho \leq R$, where R is the size of the retina, since points outside this rectangle correspond to lines in the picture plane that do not cross the retina. The quantized region is treated as a two-dimensional array of accumulators. For each point (x_i, y_i) in the picture plane, the corresponding curve given by (1) is entered in the array by incrementing the count in each cell along the curve. Thus, a given cell in the two-dimensional accumulator eventually records the total number of curves passing through it. After all figure points have been treated, the array is inspected to find cells having high counts. If the count in a given cell (θ_i, ρ_j) is k, then precisely k figure points lie (to within quantization error) along the line whose normal parameters are (θ_i, ρ_j).

An alternative interpretation of the point-curve transformation can be obtained by recognizing that the ρ computed by (1) locates the projection of the point (x_i, y_i) onto a line through the origin with slope angle θ. Thus, if a number of figure points lie close to some line l, their projections onto the line normal to l are nearly coincident (see Figure 2). A given column in the θ-ρ accumulator array is just a histogram for these projections, so a high count in a given cell clearly corresponds to a nearly colinear subset of figure points. A variation of this approach was used by Griffith [3] to find long lines in a picture.

Let us investigate how the computation required by the accumulator implementation varies with the number of figure points. To be more specific about the quantization, suppose that we restrict our attention to d_1 values of θ uniformly spaced in the interval $[0, \pi)$. Suppose further that the ρ axis in the interval $[-R, R]$ is quantized into d_2 cells. For each figure point (x_i, y_i), we use (1) to compute the d_1 different values of ρ corresponding to the d_1 possible values of the independent variable θ. Since there are n figure points, we need to carry out this computation nd_1 times. When these computations are complete, the d_1d_2 cells of the two-dimensional accumulator are inspected to find high counts. Thus the computation required grows linearly with the number of figure points. Clearly, when n is large compared to d_1, this approach is preferable to an exhaustive procedure that requires considering the lines between all $n(n-1)/2$ pairs of figure points.

Another alternative interpretation exposes the source of this efficiency. Consider again Property 4 in Section 2: Points lying on the same curve in the θ-ρ plane cor respond to lines through the same point in the pictur plane. When the curve corresponding to figure poir (x_i, y_i) is "added" to the accumulator, we are reall computing and recording the parameters of the d_1 lin in the picture plane passing through (x_i, y_i), ar because θ is quantized, these are "all the lines in th plane" passing through (x_i, y_i). Should a given paran eter pair ever recur as a result of computing the lines through some other figure point, the recurren will be reflected in an increased count in the appr priate accumulator cell. Roughly speaking, then, f each figure point the quantized transform method co siders only the set of all d_1 lines through that poir whereas more exhaustive methods consider all $(n -$ lines between the given point and all other figu points.

4. Example

The following example illustrates some of the featu of the transform approach. Figure 3(a) shows a tele sion monitor view of a box, and Figure 3(b) show digitized version of that view. A simple differenc operation locates significant intensity changes and p duces the binary picture shown in Figure 3(c). T 120 × 120 picture contains many nearly colinear fig points that can be fit well by a few straight lines.

Sampling θ at $d_1 = 9$ twenty-degree increments θ and, quantizing ρ into $d_2 = 86$ two-element cells, obtain the two-dimensional accumulator array shc in Table I. If the array entry at (θ_0, ρ_0) is k_0, ther figure points lie on parallel lines for which $\theta = \theta_0$, ar lies between ρ_0 and $\rho_0 + 2$. When many points nearly colinear, the entry for the line that fits them b is large. The largest entry in Table I occurs at (0°, – and corresponds to the middle vertical edge of the b The nine circled entries correspond to locally maxim values that exceed the arbitrary threshold of 35. corresponding nine groups of nearly colinear fi points are shown in Figure 3(d). In this exampl happens that every group corresponds to some p ically meaningful line in the picture. However, significant lines on the top of the box were not four one, because it contained very few points, and the o because it fell between the lines at $\theta = 80°$ and $\theta = 1$ The 20° angular quantization interval was chose keep the accumulator array small. Clearly, we fortunate to have found as many lines as we did, a smaller quantization interval would have to be us practice.

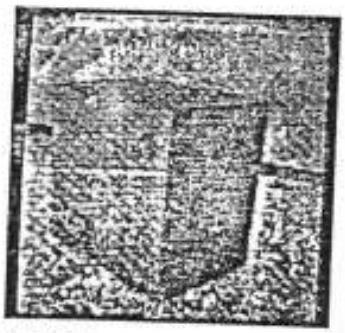
A few remarks concerning some limitations o transform approach are in order. First, the result sensitive to the quantization of both θ and ρ. quantization gives better resolution, but increase computation time and exposes the problem of clus entries corresponding to nearly colinear points. Se

the technique finds colinear points without regard to contiguity. Thus the position of a best-fit line can be distorted by the presence of unrelated figure points in another part of the picture. A related problem is that of meaningless groups of colinear points being detected. In our example, a false line would be detected if the threshold were reduced from 35 to 24, the value needed to detect the top left-hand edge of the box.

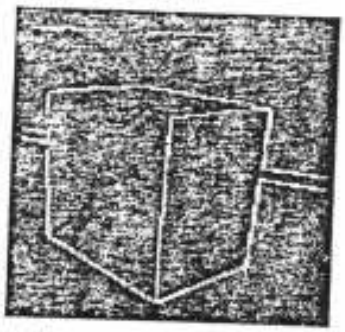
The transform approach does successfully find groups of colinear or nearly colinear figure points. If the minimum size of a significant group is known, all such groups can be detected. If additional properties such as contiguity are known, they can be used to reject meaningless results. In general, the transform approach should be viewed primarily as a computationally efficient way of accomplishing a conceptually simple step in scene analysis.

Fig. 3 An illustrative example.

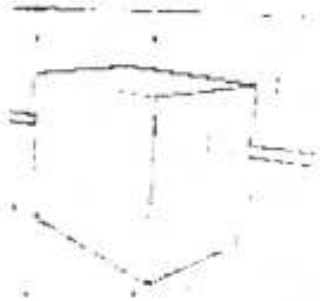
(a) Monitor



(b) Digitized



(c) Gradient



(d) Lines

5. Extensions and Conclusions

The transform method can be generalized and specialized in several ways. We note immediately that any parametrization of the family of straight lines can be used. As we have mentioned, Hough used the slope-intercept parameterization. However, this parameterization has the disadvantage of being sensitive to the choice of coordinate axes in the picture plane. If several figure points lie on a nearly vertical line, for example, both the slope and the intercept may be arbitrarily large. Thus the entire two-dimensional parameter plane must be considered. As Rosenfeld [1] has pointed out, one could do the entire problem twice, interchanging the x- and y-axes, but this would introduce additional complications. The normal parametrization avoids these disadvantages, fundamentally for the same reasons that make it useful in integral geometry: It allows us to place an invariant measure on the set of all straight lines.

An important special use of the transform method is to detect the occurrence of figure points lying on a straight line and possessing some specified property. For example, suppose we want to find whether a significant number of figure points lie on a line through the point (x_0, y_0) in the picture plane. As we have seen from Property 4, the normal coordinates of any such line must lie on (or, in practice at least near) the curve $\rho = x_0 \cos\theta + y_0 \sin\theta$. Hence, the transform process can be carried out in the usual way, but attention can be restricted to the region of the θ-ρ plane near this curve. If we find a cell with count k near this curve, then we are assured that k figure points lie on a line passing (nearly) through the point (x_0, y_0). Similarly, suppose we are interested only in lines having a given direction, say θ_0. Again, we carry out the process in the usual way, but restrict our attention to a subset of the θ-ρ plane in the vicinity of $\theta = \theta_0$.

It is clear that the general transform approach can be

Table I. Accumulator Array for Figure 3(c)

ρ \ θ	0°	20°	40°	60°	80°	100°	120°	140°	160°
85								2	
83							1	2	
81							4	4	
79			2				6	5	
77				2			8	4	2
75							6	6	2
73							4	3	3
71		2		1		1	4	2	3
69			1	4		12	4	3	5
67			3		2	14	2	3	4
65			1			11	1	2	4
63					5	2		2	4
61						1		3	9
59	4	1			11	9	1	8	12
57	4	3		3	10	12	3	10	15
55	9	5		4	5	4	5	11	12
53	6	6		4	10		11	9	14
51	4	9		4	20	2	11	10	8
49	5	6		2	10	3	11	13	8
47	8	4	4	4		2	13	10	10
45	4	7	14	3			11	6	8
43	4	18	21	5		1	12	10	8
41	9	17	21	15		25	18	7	8
39	8	20	21	13		22	11	11	7
37	12	17	22	17		9	10	9	10
35	38	14	17	17	38	8	7	9	6
33	37	16	22	21	42	10	5	9	9
31	35	11	21	23	23	8	11	9	10
29	13	18	18	23	20	14	13	9	9
27	7	16	12	30	20	20	7	9	6
25	7	18	12	22	19	27	8	7	8
23	8	12	11	20	17	52	11	6	7
21	7	17	12	23	8	11	15	11	10
19	9	14	12	16	7	7	14	6	7
17	9	12	12	16	6	9	16	12	7
15	8	13	13	11	7	10	16	14	10
13	10	9	15	11	7	10	16	13	6
11	12	11	13	14	40	10	16	13	13
9	10	10	16	14	8	9	14	21	22
7	10	8	22	12	41	6	7	12	21
5	11	12	15	11	23	6	11	14	14
3	13	15	15	8	18	7	11	16	15
1	10	14	17	11	7	8	9	10	12

ρ \ θ	0°	20°	40°	60°	80°	100°	120°	140°	160°
−85									
−83									
−81									
−79			3	1					
−77			1	3					
−75									
−73									
−71		2							
−69		2							
−67									
−65			1	2					
−63		1			2				
−61	5								
−59	7		2	1		1			
−57	6					2	1		2
−55	0	10	6			1	1		4
−53	16	13	12		18	4	1		6
−51	40	15	11		15	16			6
−49	32	18	11	57	15	23	1	1	5
−47	10	16	11	22	14	16	21	9	5
−45	7	17	11	11	16	18	41	21	6
−43	8	12	14	10	13	17	12	17	6
−41	6	7	14	11	14	14	7	19	12
−39	7	10	9	8	12	8	11	20	23
−37	7	7	14	8	17	9	12	18	24
−35	8	9	17	8	10	7	10	23	23
−33	6	12	15	8	12	9	11	22	26
−31	5	9	19	9	8	11	16	18	15
−29	9	10	12	9	8	9	18	18	15
−27	7	12	10	8	6	9	18	19	19
−25	5	10	8	8	7	7	22	9	14
−23	6	11	9	9	6	11	19	12	9
−21	7	15	9	7	10	10	16	10	11
−19	6	13	8	16	9	11	17	9	10
−17	7	17	9	15	7	11	16	14	13
−15	6	15	10	17	8	13	10	14	9
−13	10	15	9	15	9	17	11	13	12
−11	10	13	10	7	8	17	9	11	15
−9	7	14	8	7	8	23	8	12	15
−7	9	15	12	7	8	21	7	13	12
−5	27	13	15	9	7	14	10	12	15
−3	26	14	14	6	8	12	9	11	18
−1	10	13	18	9	8	8	11	12	15

extended to curves other than straight lines. For example, suppose we want a method to detect circular configurations of figure points. We can choose a parametric representation for the family of all circles (within a retina) and transform each figure point in the obvious way. If, as a parametric representation, we describe a circle in the picture plane by

$$(x - a)^2 + (y - b)^2 = c^2,$$

then an arbitrary figure point (x_i, y_i) will be transformed into a surface in the a-b-c parameter space defined by

$$(x_i - a)^2 + (y_i - b)^2 = c^2.$$

In this example, then, each figure point will be transformed into a right circular cone in a three-dimensional parameter space. If the cones corresponding to many figure points intersect at a single point, say the point (a_0, b_0, c_0), then all the figure points lie on the circle defined by those three parameters. As in the preceding case of straight lines, no saving is effected if the entire process is performed analytically. However, the process can be implemented efficiently by using a three-dimensional array of accumulators representing the thr dimensional parameter space.

In principle, then, the transform method extends arbitrary curves. We need only pick a convenient para eterization for the family of curves of interest and th proceed in the obvious way. A parameterization havi bounded parameters is obviously preferable, althou this is not essential. It is much more important to ha a small number of parameters since the accumulat implementation requires quantization of the entire p rameter space and the computation grown exponentia with the number of parameters.

Received November 1970, revised April 1971

References

1. Rosenfeld, A. *Picture Processing by Computer*. Academic Pre New York, 1969.
2. Hough, P.V.C. Method and means for recognizing complex patterns. U.S. Patent 3,069,654, Dec. 18, 1962.
3. Griffith, A.K. Computer recognition of prismatic solids. Ph Th., Dep. of Math., MIT, June 1970.