

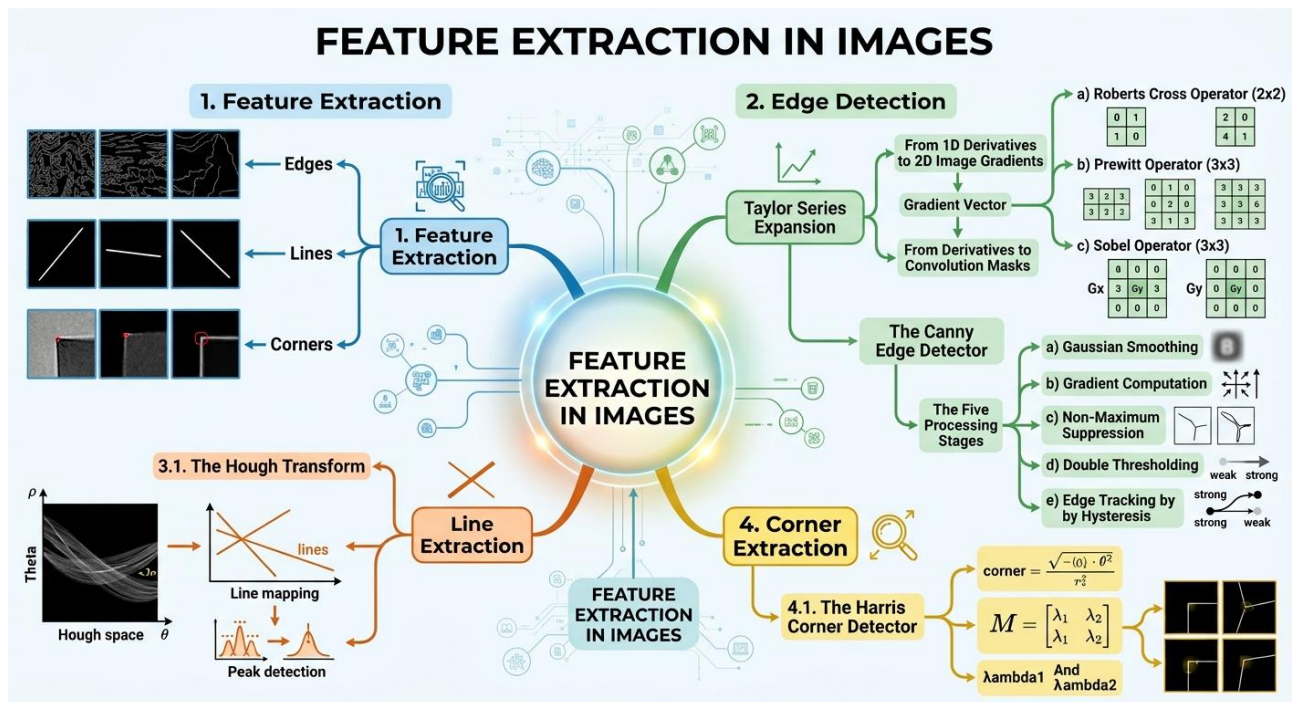
Chapter 3: FEATURE EXTRACTION IN IMAGES

Dr. MAAMAR BEKHOUCHE

m.bekhouche@centre-univ-mila.dz

2026-2027

1. Feature Extraction



1.1. Introduction and Definition

Feature extraction is the process of identifying and representing informative structures in an image. Instead of processing every pixel independently, feature extraction emphasizes patterns that are useful for interpretation, measurement, segmentation, recognition, and computer vision.

In a grayscale image, important local structures are often revealed by changes in intensity. A nearly constant region contains little structural information, whereas an abrupt change may indicate an object boundary. Other characteristic structures include elongated straight segments and points where several intensity changes meet.

1.2. Definition of Image Features

An image feature is a measurable property or local structure that provides useful information about the image content. Features can be described by intensity values, derivatives, geometric relationships, or local neighborhoods.

1.3. Main Types of Features Considered in This Chapter

- Edges: locations where image intensity changes significantly.
- Lines: elongated structures that can often be represented by straight geometric models.
- Corners: points where the local image structure changes significantly in more than one direction.

1.4. General Feature-Extraction Workflow

A typical workflow begins with a grayscale or intensity image, optionally reduces noise, computes a local measurement such as a derivative or gradient, and finally applies a decision rule or geometric model to identify the desired features.

Input image → Pre-processing → Local measurements → Feature decision →
Feature map

1.5. Chapter Objectives

By the end of this chapter, students should be able to explain image derivatives, compute a two-dimensional gradient, understand and compare Roberts, Prewitt, and Sobel operators, describe the complete Canny detector, explain the Hough transform for lines, and understand the mathematical basis of the Harris corner detector.

2. Edge Detection

2.1. Introduction and Definition

Definition – Edge

An **edge** is a set of connected pixels located on the boundary between two regions of an image, across which the intensity (or colour, or texture) changes abruptly. **Edge detection** is the process of locating these pixels and producing a binary *edge map*.

Edges are physically produced by four main phenomena in the scene:

- **depth discontinuity**: an object lies in front of a farther background;
- **surface-orientation discontinuity**: the edge of a cube, where two faces with different orientations meet;
- **change of material or reflectance**: a printed letter on a white sheet;
- **illumination change**: the boundary of a cast shadow.

Along a line perpendicular to the edge, the intensity profile takes one of the idealized shapes shown in Figure 1. A **step edge** is a perfect jump; in real images, optical blur and sampling transform it into a **ramp edge**. A **roof edge** and a **line (ridge)** correspond to thin structures (a crease, a wire, a road marking).

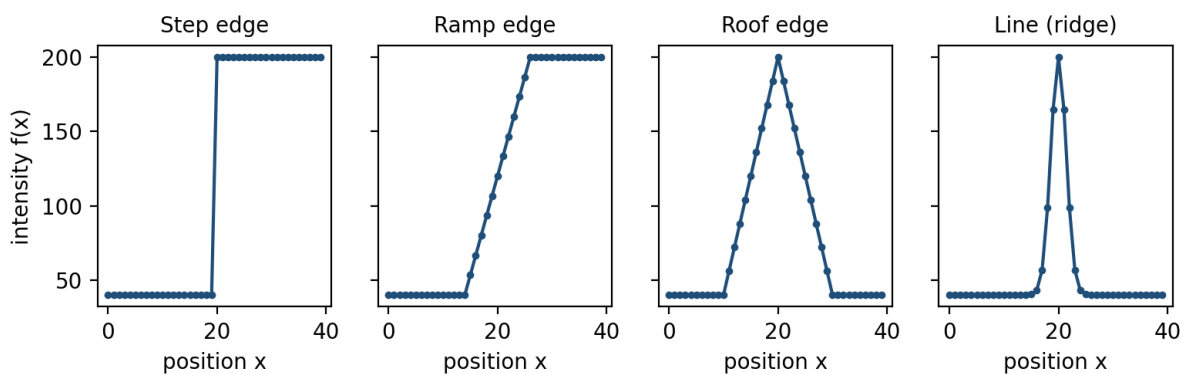


Figure 1: Idealized 1-D intensity profiles across the main types of edges.

Why detect edges? Edges keep most of the structural information of an image (shape and position of objects) while discarding the large homogeneous regions. They also resemble the way the human visual system analyses a scene. The resulting edge map is much more compact than the original image and is the basis of segmentation and recognition.

General principle. Since an edge is a place of rapid intensity variation, it can be found by measuring the local rate of change of the image, that is, its **derivatives**. Most edge detectors follow the same three steps:

1. **Smoothing** (filtering): reduce noise, because derivatives amplify it;
2. **Enhancement**: compute a derivative-based quantity (gradient or Laplacian) that is large where edges are present;

Detection and localization: decide which pixels are edge points (thresholding) and where exactly the edge lies (thinning, zero-crossing).

2.2. Mathematical Foundations

2.2.1. Taylor Series Expansion

Let $f(x)$ be a sufficiently smooth one-dimensional intensity function. Around a point x , the Taylor expansion at $x+h$ is

$$f(x+h) = f(x) + h f'(x) + (h^2/2!) f''(x) + (h^3/3!) f'''(x) + \dots$$

Similarly, at $x-h$:

$$f(x-h) = f(x) - h f'(x) + (h^2/2!) f''(x) - (h^3/3!) f'''(x) + \dots$$

These expansions allow derivatives, which are continuous mathematical quantities, to be approximated from discrete neighboring pixel values.

2.2.2. Finite Difference Approximations

If $h = \Delta x = 1$, then:

The **forward difference** is obtained by subtracting the intensity at x from the intensity at $x+1$.

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

The **backward difference** is

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

Subtracting the two Taylor expansions gives the **central difference** approximation:

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

$$f''(x) = f(x+1) - 2f(x) + f(x-1)$$

2.3. Characterization of Edge Points Using Derivatives

Consider the profile of a ramp edge (Figure 2). The intensity goes from a low to a high plateau. The derivatives behave as follows:

- **First derivative:** it is zero on the plateaus and takes a large absolute value on the ramp. An edge point is located at a **local maximum of $|f'(x)|$** (the sign of f' tells whether the edge is dark-to-bright or bright-to-dark).
- **Second derivative:** it is positive on the dark side of the ramp, negative on the bright side and passes through zero at the middle. An edge point is located at a **zero-crossing of $f''(x)$** , that is, where f' changes sign.

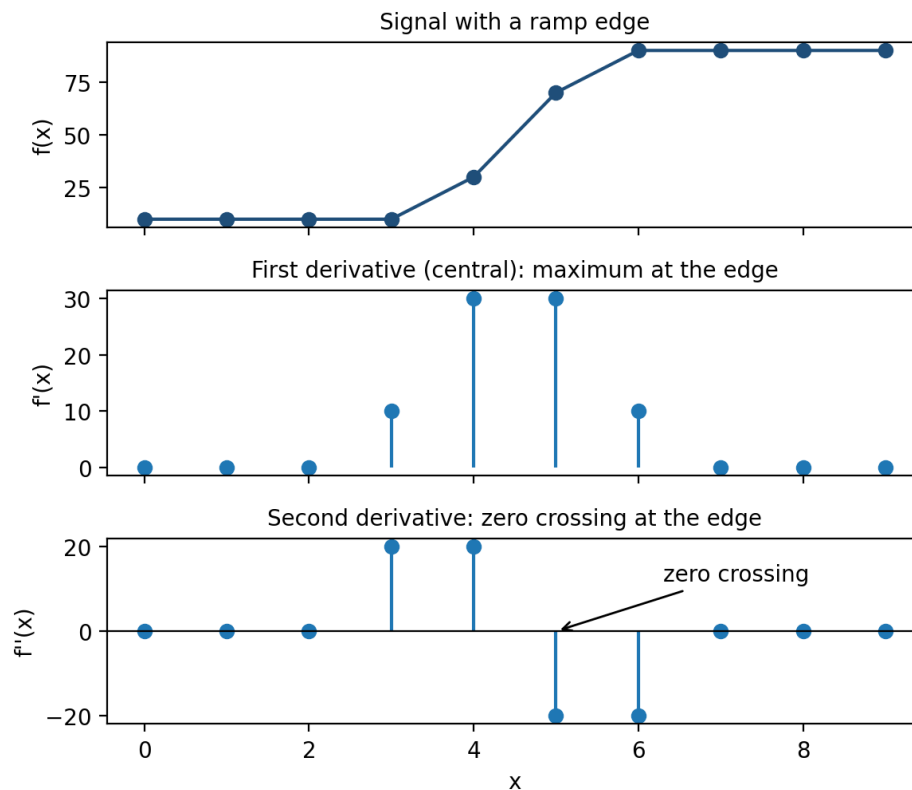


Figure 2: A 1-D ramp edge, its first derivative (maximum) and its second derivative (zero-crossing).

Example – Edge in a 1-D signal

Signal $f = [10, 10, 10, 10, 30, 70, 90, 90, 90, 90]$ (positions $x = 0$ to 9). Using the central formulas of Table 1:

x	0	1	2	3	4	5	6	7	8	9
f(x)	10	10	10	10	30	70	90	90	90	90
f'(x)	–	0	0	10	30	30	10	0	0	–
f''(x)	–	0	0	20	20	–20	–20	0	0	–

$|f'|$ reaches its maximum (30) at $x = 4$ and $x = 5$, and f'' changes sign between $x = 4$ (+20) and $x = 5$ (–20). Both criteria agree: the edge lies between pixels 4 and 5 (position ≈ 4.5).

Comparison of the two criteria. The first derivative gives a direct measure of the edge *strength* and is robust, but it produces a thick response (several pixels wide). The second derivative localizes the edge precisely (a single zero-crossing) but it is much more sensitive to noise, because each additional derivative amplifies the high frequencies. For this reason, noise filtering is essential before differentiating, especially for second-order operators.

2.4. From 1D Derivatives to 2D Image Gradients

2.4.1. The Gradient Vector in 2D

A grayscale image can be represented by a continuous intensity function $I(x,y)$. Its gradient contains the partial derivatives in the horizontal and vertical directions.

$$\nabla I(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

The gradient points in the direction of the greatest local increase in intensity. Therefore, it is perpendicular to an ideal edge.

2.4.2. Gradient Magnitude

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

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

2.4.3. Gradient Direction

$$\theta(x,y) = \arctan\left(\frac{g_y}{g_x}\right)$$

The atan2 function is preferred because it correctly determines the quadrant of the direction. The gradient direction is normal to the local edge direction.

Example

Suppose that, at a particular pixel of an image, the horizontal and vertical gradient components are:

$$G_x=30, G_y=40$$

Gradient magnitude

The gradient magnitude tells us **how strong the intensity variation is** at this pixel:

$$|\nabla I| = \sqrt{(I_x^2 + I_y^2)} = \sqrt{(30^2 + 40^2)} = 50$$

So, the **gradient magnitude is 50**. This indicates a relatively strong intensity change, which may correspond to an **edge**.

Gradient orientation

The orientation of the gradient is calculated using:

$$\theta(x,y) = \arctan\left(\frac{g_y}{g_x}\right) = \arctan\left(\frac{40}{30}\right) = 53.1$$

This means that the **gradient vector points approximately 53.1°** from the positive horizontal x-axis.

Edge direction

An important point is that the **gradient direction and edge direction are not the same**.

The gradient points in the direction of the **maximum intensity change**, while an edge itself is **perpendicular to the gradient**.

Therefore, the edge direction is:

$$\theta_{\text{edge}} = \theta + 90^\circ = 53.1^\circ + 90^\circ = 143.1^\circ$$

So:

- **Gradient direction:** 53.1°
- **Edge direction:** 143.1°

A discrete derivative is a **weighted sum of neighbouring pixels**, so it can be implemented as a spatial filter. A mask h of size $(2a+1) \times (2b+1)$ is slid over the image and, at each position, the output is:

$$g(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b h(i, j) \cdot f(x+j, y+i) \quad (14)$$

Remark – Correlation and convolution

The formula above is a **correlation**. A true **convolution** uses the mask rotated by 180° . Derivative masks are antisymmetric, so the two operations only differ by a sign, which has no effect on the gradient magnitude. By convention, the masks of this chapter are written in correlation form, and a negative value on the left (or top) of the mask means “subtract the previous pixels”.

Step 1: Forward Difference Masks

From the forward difference $f(x+1) - f(x)$, we obtain a mask with the coefficients $[-1, 1]$, applied to the current pixel and its right neighbour for G_x , or to the current pixel and its lower neighbour for G_y :

$$G_x = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Drawback: the mask has an even size, so the result is attached to the middle of two pixels (half-pixel shift), and the estimation is asymmetric.

Step 2: Central Difference Masks

The central difference $[f(x+1) - f(x-1)]/2$ gives a symmetric 3-element mask, centred on the current pixel (the factor $\frac{1}{2}$ is often dropped or merged into a global normalization):

$$G_x = \frac{1}{2} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \frac{1}{2} \times \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

This mask uses a single row (or column) of pixels, so it remains very sensitive to noise. The idea of the classical operators is to **average the derivative along the direction perpendicular to the derivation**, which is obtained by stacking three shifted copies of the central-difference mask.

2.4.4. Standard Gradient-Based Edge Detection Operators

a) Roberts Cross Operator (2×2)

Proposed by Roberts in 1963, it computes the differences along the two **diagonals** of a 2×2 neighbourhood:

$$G_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad G_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The magnitude is $\sqrt{(G_1^2 + G_2^2)}$ or $|G_1| + |G_2|$. It is the fastest operator and gives thin edges, but it is very sensitive to noise and, having an even size, it is shifted by half a pixel.

b) Prewitt Operator (3×3)

The Prewitt operator (1970) averages the central difference over three adjacent rows (for G_x) or columns (for G_y) with a uniform weight:

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

The mask is **separable**: $G_x = [1 \ 1 \ 1]^T \times [-1 \ 0 \ 1]$, that is, a box smoothing in the vertical direction followed by a central difference in the horizontal direction. With the factor $1/6 (= 1/3 \times 1/2)$, it gives the true gradient value.

c) Sobel Operator (3×3)

The Sobel operator (1968) is similar but gives more weight to the row (or column) that contains the central pixel:

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

It is also separable: $G_x = [1 \ 2 \ 1]^T \times [-1 \ 0 \ 1]$. The smoothing part $[1 \ 2 \ 1]$ is a binomial approximation of a Gaussian, which suppresses noise better than the uniform box and makes the response closer to isotropic. The true gradient is obtained with the factor $1/8$ ($= 1/4 \times 1/2$). Sobel is the most widely used of the three operators.

Example – Gradient of a vertical edge

Consider the 3×3 patch with rows $[10, 10, 50]$ (the three rows are identical: a vertical edge between the 2nd and 3rd columns). Then:

- Sobel: $G_x = (50-10) + 2(50-10) + (50-10) = 160$; $G_y = 0$; $|\nabla I| = 160$; $\theta = 0^\circ$. With the $1/8$ factor: 20 grey levels per pixel.
- Prewitt: $G_x = 3 \times (50-10) = 120$; $G_y = 0$; $|\nabla I| = 120$ (20 with the $1/6$ factor).
- Roberts (on the top-left 2×2 block $[10 \ 10; 10 \ 10]$): $G_1 = 10 - 10 = 0$ and $G_2 = 0$: the edge is not seen by this block; it would be seen by the block shifted one column to the right.

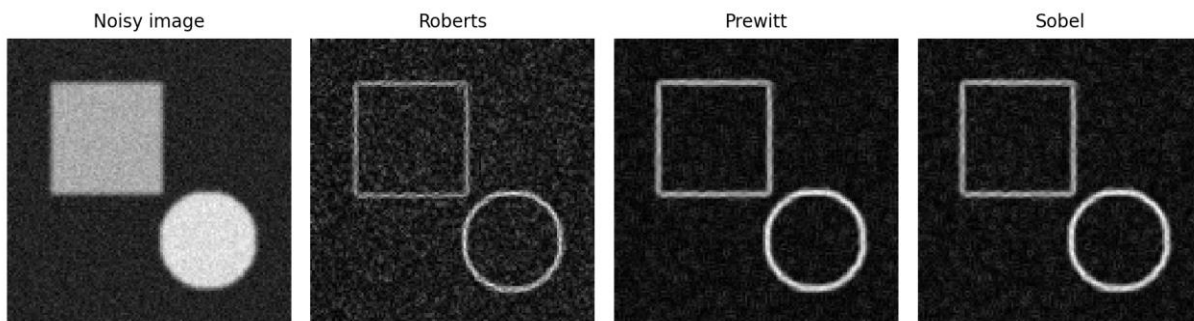


Figure 4: Edge maps (gradient magnitude) obtained with the Roberts, Prewitt and Sobel operators on a noisy image.

Figure 4 shows the practical difference: Roberts produces a noisy response, whereas Prewitt and Sobel, thanks to their smoothing component, give cleaner and thicker edges. Table 2 summarizes the comparison.

Table 2: Summary table of gradient kernels (comparison of first-order derivative operators).

Property	Roberts	Prewitt	Sobel
Mask size	2×2	3×3	3×3
Kernel G_x	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
Kernel G_y	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Smoothing	None	Low	Moderate
Noise sensitivity	High	Medium	Low to medium
Typical use	Fast edge detection	Simple edge detection	General-purpose edge detection

2.4.5. Gradient Magnitude Thresholding and Noise Reduction

A gradient image contains responses of different strengths. To obtain a binary edge map, a threshold T can be applied to the gradient magnitude.

$$E(x,y) = \{ 1, \text{ if } |\nabla I(x,y)| \geq T ; 0, \text{ otherwise } \}$$

A low threshold detects weak edges but also admits noise. A high threshold suppresses noise but may remove genuine weak edges. Choosing a suitable threshold is therefore important.

2.4.6. Pre-filtering Before Edge Detection

Noise creates rapid intensity variations and can therefore produce large derivative responses. A common strategy is to smooth the image before differentiation. Mean and Gaussian filters are examples of such pre-filters.

$$I_s = I * H$$

where H is a smoothing kernel and $*$ denotes convolution.

2.4.7. Mean Filtering Followed by Sobel Detection

A simple cascade is:

original image $\rightarrow$ mean filter $\rightarrow$ Sobel gradients $\rightarrow$ gradient magnitude $\rightarrow$ threshold.
The mean filter reduces high-frequency fluctuations, while Sobel extracts first-order intensity changes.

2.4.8. Combined Pre-filtering and Gradient Operators

Some gradient operators already incorporate limited smoothing. Nevertheless, explicit Gaussian smoothing may be preferable when the image is strongly contaminated by noise or when edge localization must be balanced against noise suppression.

2.4.9. Limitations of Simple Gradient Thresholding

Problem	Explanation	Consequence
Noise responses	Noise can create large local derivatives	False edges
Threshold dependence	One threshold may not suit all image regions	Missed or excessive edges
Thick edges	A strong transition can occupy several pixels	Poor localization
Broken contours	Weak portions may fall below threshold	Disconnected edges
Texture	Repeated local variations create many responses	Cluttered edge map

Table 3. Limitations of simple gradient-based thresholding.

2.5. The Canny Edge Detector

The Canny detector was designed to obtain edge maps with good detection, localization, and single-response properties. It combines smoothing, gradient estimation, non-maximum suppression, and hysteresis thresholding.

2.5.1. The Five Processing Stages

1. Gaussian smoothing to reduce noise.
2. Gradient computation to estimate edge strength and direction.
3. Non-maximum suppression to thin broad gradient responses.
4. Double thresholding to classify strong, weak, and non-edge pixels.
5. Edge tracking by hysteresis to retain weak pixels connected to strong edges.

a) Gaussian Smoothing

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

The smoothed image is obtained by convolving the input image with the Gaussian kernel:

$$I_s(x, y) = I(x, y) * G_{\sigma}(x, y)$$

where:

- $I(x, y)$: original image
- $G_{\sigma}(x, y)$: Gaussian kernel
- σ : standard deviation controlling the amount of smoothing
- $I_s(x, y)$: smoothed image
- $*$: convolution operation.

b) Gradient Computation

After smoothing, horizontal and vertical derivatives are computed. The magnitude and direction are then calculated:

$$M(x, y) = \sqrt{(G_x^2 + G_y^2)}$$
$$\theta(x, y) = \arctan\left(\frac{g_y}{g_x}\right)$$

c) Non-Maximum Suppression

Non-maximum suppression examines each candidate pixel along the gradient direction. The pixel is retained only if its magnitude is a local maximum relative to its two neighboring samples along that direction. This converts broad responses into thin contours.

d) Double Thresholding

Two thresholds are used: a high threshold T_H and a low threshold T_L . Pixels above T_H are strong edges. Pixels between T_L and T_H are weak candidates. Pixels below T_L are rejected.

Strong: $M \geq T_H$

Weak: $T_L \leq M < T_H$

Non-edge: $M < T_L$

e) Edge Tracking by Hysteresis

Weak pixels are retained when they are connected to strong edges through neighboring edge pixels. This prevents isolated weak noise responses from being accepted while preserving meaningful contour segments.

2.5.2. Numerical Example

Example canny

matrice I d'origine (avec padding de zeros)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 255 & 255 & 255 \end{bmatrix}$$

Etape 01 Lissage gaussien (Gaussian Smoothing)

Ligne 1

Pixel (1,0) et (1,1) : Tous les pixels recouverts sous le filtre sont 0. $\Rightarrow 0$

Pixel (1,2) : Le bas-droite du filtre touche $I(2, 2) = 255$ avec le poids 1.

$$\frac{255 \times 1}{16} = \frac{255}{16} \approx \mathbf{15.9375}$$

Pixel (1,3) : Le bas du filtre touche $I(2, 2) = 255$ (poids 2) et $I(2, 3) = 255$ (poids 1).

$$\frac{255 \times (2 + 1)}{16} = \frac{765}{16} \approx \mathbf{47.8125}$$

Pixel (1,4) : Similaire à (1,3) en raison de la répétition des valeurs 255 à droite. $\Rightarrow \mathbf{47.8125}$

Ligne 2

Pixel (2,0) : Tous les pixels de l'image recouverts sont 0. $\Rightarrow 0$

Pixel (2,1) : Le filtre couvre $I(2, 2) = 255$ (poids 2), $I(3, 2) = 255$ (poids 4), $I(4, 2) = 255$ (poids 2).

$$\frac{255 \times (2 + 4 + 2)}{16} = \frac{255 \times 8}{16} = \mathbf{127.5}$$

Pixel (2,2) : Le filtre couvre les pixels non nuls $I(2, 2..3)$, $I(3, 2..3)$, $I(4, 2..3)$.

Somme des poids concernés = $2 + 1 + 4 + 2 + 2 + 1 = 12$

$$\frac{255 \times 12}{16} = \mathbf{191.25}$$

Pixel (2,3) et (2,4) : Les 3 colonnes couvertes par le filtre contiennent des 255 sur les lignes 2, 3 et 4. La somme de tous les poids des 2ème et 3ème lignes du filtre vaut

$$2 + 4 + 2 + 1 + 2 + 1 = 12.$$

$\Rightarrow$ Similaire à (2,2) = $\mathbf{191.25}$

Ligne 3

Pixel (3,0) : Tous les pixels sous le filtre sont 0. $\Rightarrow 0$

Pixel (3,1) : Similaire à (2,1) car les lignes 2, 3, 4 contiennent 255 en colonne 2. $\Rightarrow \mathbf{127.5}$

Pixel (3,2) : Le filtre est entièrement couvert par des 255 sur ses colonnes de droite (colonnes 2, 3, 4 de l'image).

Somme des poids des colonnes de droite = $2 + 4 + 2 + 1 + 2 + 1 = 12$

$$\frac{255 \times 12}{16} = \mathbf{191.25}$$

Pixel (3,3) et (3,4) : Similaires à (3,2) $\Rightarrow \mathbf{191.25}$

Ligne 4

Pixel (4,0) : Seuls des 0 sous le filtre. $\Rightarrow 0$

Pixel (4,1) : Le filtre dépasse en bas (padding 0). Seules les lignes 3 et 4 de l'image sont couvertes pour la colonne 2 (poids $2 + 4 = 6$).

$$\frac{255 \times (2 + 4)}{16} = \frac{255 \times 6}{16} = \mathbf{95.625}$$

• • • • •
• • • • •
• • • • •

Image résultante $I_{\text{lissé}}$ (arrondie à 1 décimale) :

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15.9 & 47.8 & 47.8 \\ 0 & 127.5 & 191.3 & 191.3 & 191.3 \\ 0 & 127.5 & 191.3 & 191.3 & 191.3 \\ 0 & 95.6 & 191.3 & 191.3 & 191.3 \end{bmatrix}$$

Etaape 02 Calcul du gradient (Gradient Computation)

Voici la sous-région 3×3 centrée autour du pixel $(2, 1)$ (correspondant aux lignes 1 à 3 et colonnes 0 à 2 de $I_{\text{lissé}}$) :

$$\text{Région } 3 \times 3 = \begin{bmatrix} 0 & 0 & 15.9 \\ 0 & 127.5 & 191.3 \\ 0 & 127.5 & 191.3 \end{bmatrix}$$

1. Calcul des Composantes du Gradient (g_x et g_y)

On utilise les masques de Sobel S_x et S_y :

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

A. Gradient Horizontal g_x (Variations en x) :

$$g_x = (-1 \times 0) + (0 \times 0) + (1 \times 15.9) + (-2 \times 0) + (0 \times 127.5) + (2 \times 191.3) + (-1 \times 0) + (0 \times 127.5) + (1 \times 191.3)$$
$$g_x = 15.9 + 382.6 + 191.3 = \mathbf{589.8}$$

B. Gradient Vertical g_y (Variations en y) :

$$g_y = (-1 \times 0) + (-2 \times 0) + (-1 \times 15.9) + (0 \times 0) + (0 \times 127.5) + (0 \times 191.3) + (1 \times 0) + (2 \times 127.5) + (1 \times 191.3)$$
$$g_y = -15.9 + 255.0 + 191.3 = \mathbf{430.4}$$

2. Calcul de la Norme du Gradient (M)

La norme représente l'intensité / force du contour au pixel $(2, 1)$:

$$M(2, 1) = \sqrt{g_x^2 + g_y^2} = \sqrt{(589.8)^2 + (430.4)^2}$$
$$M(2, 1) = \sqrt{347,864.04 + 185,244.16} = \sqrt{533,108.2} \approx \mathbf{730.14}$$

3. Calcul de la Direction du Gradient (θ)

La direction donne l'angle de la plus forte variation lumineuse :

$$\theta(2, 1) = \arctan\left(\frac{g_y}{g_x}\right) = \arctan\left(\frac{430.4}{589.8}\right)$$
$$\theta(2, 1) = \arctan(0.7297) \approx \mathbf{36.12^\circ}$$

En pratique dans Canny, cet angle de 36.12° sera arrondi à la direction principale la plus proche, soit 45° , pour l'étape de suppression des non-maxima.

Les angles sont quantifiés selon les 4 directions principales du réseau de pixels :

Plage d'angle	Direction quantifiée	Voisins à comparer
$[-22.5^\circ, 22.5^\circ]$ ou $[157.5^\circ, 202.5^\circ]$	0° (Horizontal)	Gauche et Droite
$[22.5^\circ, 67.5^\circ]$	45° (Diagonale /)	Bas-Gauche et Haut-Droit
$[67.5^\circ, 112.5^\circ]$	90° (Vertical)	Haut et Bas
$[112.5^\circ, 157.5^\circ]$	135° (Diagonale \)	Haut-Gauche et Bas-Droit

Comme $\theta = 36.12^\circ$, il appartient à l'intervalle $[22.5^\circ, 67.5^\circ]$.

$$\Rightarrow \theta_{\text{quantifié}} = 45^\circ$$

Etape 03 suppression de non-maxima

Comparaison avec les deux voisins orthogonaux

Puisque la direction du gradient est à 45° , la ligne du contour lui est orthogonale. On compare donc la norme de $P(2, 1)$ avec ses deux voisins situés le long de la direction à 45° :

- Voisin 1 (V_1) : Haut-Droit $\rightarrow$ Pixel $(1, 2)$
- Voisin 2 (V_2) : Bas-Gauche $\rightarrow$ Pixel $(3, 0)$

Calcul de la règle de décision

Imaginons les normes calculées à l'étape 1 pour ces deux voisins :

- $M(2, 1) = 730.14$ (notre pixel)
- $M(1, 2) \approx 520.0$ (voisin Haut-Droit)
- $M(3, 0) = 0$ (voisin Bas-Gauche, car région plate à 0)

Règle :

Si $M(2, 1) \geq M(V_1)$ ET $M(2, 1) \geq M(V_2) \implies$ CONSERVER $M(2, 1)$

Sinon $\implies M(2, 1) = 0$

Application :

$$730.14 \geq 520.0 \quad \text{et} \quad 730.14 \geq 0 \quad (\text{VRAI})$$

Résultat :

Le pixel $(2, 1)$ est un **maximum local**. Sa valeur $M(2, 1) = 730.14$ est donc **conservée** pour l'étape suivante, tandis que ses voisins non-maximaux sont remis à 0.

Étape 4 : Seuillage par hystérésis (*Hysteresis Thresholding*)

On fixe deux seuils globaux pour l'ensemble de l'image :

- Seuil haut (T_{high}) : par exemple 150
- Seuil bas (T_{low}) : par exemple 50

On compare la valeur du pixel $M(2, 1)$ à ces deux seuils :

1. Test du Seuil Haut ($M \geq T_{\text{high}}$) :

$$730.14 \geq 150 \quad (\text{VRAI})$$

2. Classification :

Puisque sa valeur (730.14) dépasse largement le seuil haut T_{high} , le pixel $P(2, 1)$ est classé comme **PIXEL FORT (Strong Pixel)**.

Note :

- Si sa valeur avait été comprise entre 50 et 150, il aurait été classé pixel faible (*Weak Pixel*).
- Si sa valeur avait été inférieure à 50, il aurait été éliminé immédiatement ($M = 0$).

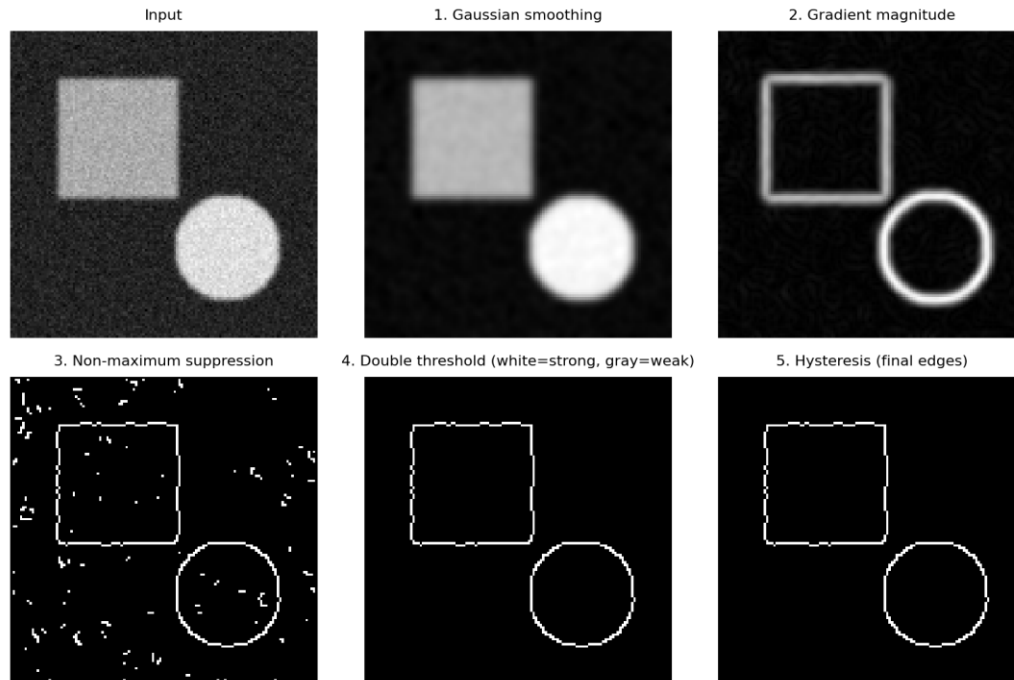


Figure 7: Intermediate results of the Canny detector

3. Line Extraction

3.1. Introduction

An edge detector provides a set of unconnected edge pixels, but many applications need **geometric primitives**: straight lines or segments. Examples are lane detection for autonomous driving, document skew correction, building and road extraction in aerial images, or the measurement of industrial parts.

Finding the line that passes through a set of edge points is difficult because the edge map contains gaps (broken contours), noise points and points belonging to other structures. A least-squares fit is very sensitive to such outliers. The **Hough transform** solves this problem by a *voting* principle: each edge pixel votes for all the lines that could pass through it, and the lines that receive many votes are retained.

3.2. Mathematical Representation of a Line

A conventional slope-intercept representation is $y = mx + b$. However, it cannot represent vertical lines using a finite slope. The Hough transform therefore commonly uses a polar representation.

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

Here, ρ is the perpendicular distance from the origin to the line, and θ is the angle of the line's normal with the x-axis.

3.3. The Hough Transform

The Hough transform was introduced by P. Hough in 1962 for bubble-chamber images using the slope-intercept form $y = mx + b$. This form cannot represent vertical lines ($m \rightarrow \infty$), which led Duda and Hart (1972) to propose the polar parameterization used today.

3.3.1. Polar Parameterization (Duda & Hart, 1972)

A line is described by two parameters: ρ , the signed distance from the origin to the line, and θ , the angle between the x-axis and the normal to the line (Figure 8). A point (x, y) lies on this line if and only if:

$$\rho = x \cos \theta + y \sin \theta \quad (19)$$

with $\theta \in [0^\circ, 180^\circ[$ and $\rho \in [-D, D]$, where $D = \sqrt{W^2 + H^2}$ is the image diagonal (W and H are the width and height).

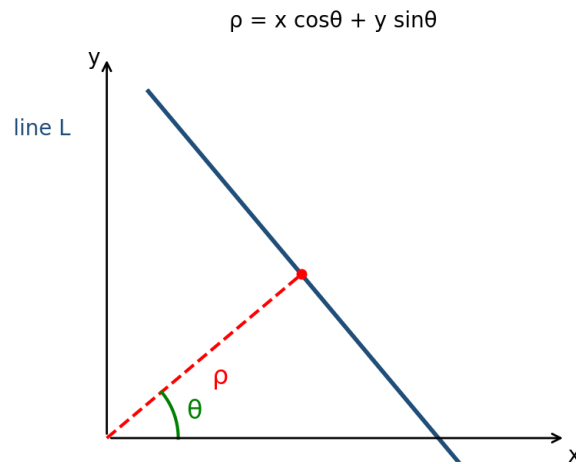


Figure 8: Polar parameterization of a line: distance ρ and normal angle θ .

This representation has two important consequences (a duality between the image space and the parameter space):

- a **point** (x_0, y_0) of the image corresponds to a **sinusoid** $\rho = x_0 \cos \theta + y_0 \sin \theta$ in the (θ, ρ) space (all the lines passing through the point);
- **collinear points** of the image give sinusoids that all intersect at the same point (θ_0, ρ_0) , which are the parameters of their common line.

3.3.2. Hough Line Detection Algorithm

- **Edge map:** compute a binary edge image (for example with Canny).
- **Accumulator:** quantize the parameter space with the steps $\Delta\theta$ (for example 1°) and $\Delta\rho$ (for example 1 pixel) and create a 2-D array $A(\rho, \theta)$ initialized to 0.
- **Voting:** for each edge pixel (x, y) and for each θ_j , compute $\rho = x \cos \theta_j + y \sin \theta_j$ and increment the corresponding cell.
- **Peak detection:** select the cells whose value exceeds a threshold and are local maxima of A (non-maximum suppression in the accumulator).
- **Line extraction:** each peak (ρ, θ) defines a line; optionally, it is cut into segments by looking at the edge pixels that support it (minimum length, maximum gap).

```
A[:, :] = 0
for each edge pixel (x, y):
    for j = 0 ... N_theta - 1:
        theta = j * d_theta
        rho = x*cos(theta) + y*sin(theta)
        i = round((rho + D) / d_rho)
        A[i, j] = A[i, j] + 1
peaks = cells of A with A[i, j] >= T_vote (local maxima)
```

The cost is proportional to $N_{\text{edge}} \times N_\theta$. The choice of $\Delta\rho$ and $\Delta\theta$ is a compromise: very fine steps give a precise but noisy accumulator (votes are dispersed over neighbouring cells), while coarse steps merge nearby lines. The Hough principle extends to other parametric shapes: circles need a 3-D accumulator (a, b, r) , and the *probabilistic Hough transform* uses a random subset of the edge points to reduce the computation time.

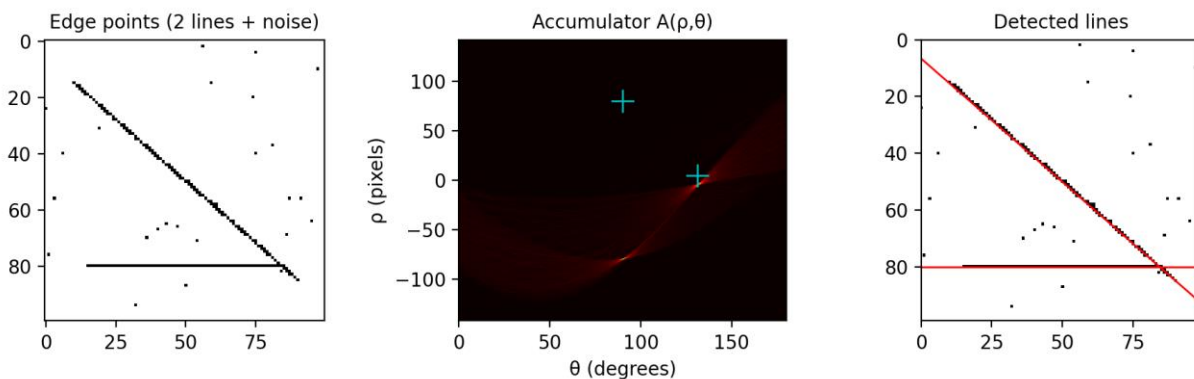


Figure 9: Hough transform: edge points, accumulator $A(\rho, \theta)$ with its two peaks, and the corresponding detected lines.

3.3.3. Numerical Walkthrough Example: Hough Transform

Consider the five edge points $P_1(1, 1)$, $P_2(2, 2)$, $P_3(3, 3)$, $P_4(4, 4)$ and $P_5(1, 4)$: four of them are aligned on the line $y = x$, and the last one is an outlier. To keep the example small, we quantize θ in four values (0° , 45° , 90° , 135°) and round ρ to the nearest integer ($\Delta\rho = 1$). Table 4 gives the computation of $\rho = x \cos \theta + y \sin \theta$ for each point.

Table 4: Hough accumulator voting matrix calculations (exact $\rho \rightarrow$ rounded ρ).

Point (x, y)	$\theta = 0^\circ$	$\theta = 45^\circ$	$\theta = 90^\circ$	$\theta = 135^\circ$
P₁ (1, 1)	1.00 $\rightarrow$ 1	1.41 $\rightarrow$ 1	1.00 $\rightarrow$ 1	0.00 $\rightarrow$ 0
P₂ (2, 2)	2.00 $\rightarrow$ 2	2.83 $\rightarrow$ 3	2.00 $\rightarrow$ 2	0.00 $\rightarrow$ 0
P₃ (3, 3)	3.00 $\rightarrow$ 3	4.24 $\rightarrow$ 4	3.00 $\rightarrow$ 3	0.00 $\rightarrow$ 0
P₄ (4, 4)	4.00 $\rightarrow$ 4	5.66 $\rightarrow$ 6	4.00 $\rightarrow$ 4	0.00 $\rightarrow$ 0
P₅ (1, 4)	1.00 $\rightarrow$ 1	3.54 $\rightarrow$ 4	4.00 $\rightarrow$ 4	2.12 $\rightarrow$ 2

Each row of Table 4 gives one vote per column. Counting, for each angle, how many points fall into the same rounded ρ gives the accumulator:

Accumulator matrix $A(\rho, \theta)$

$\rho \setminus \theta$	0°	45°	90°	135°
0	0	0	0	4
1	2	1	1	0
2	1	0	1	1
3	1	1	1	0
4	1	2	2	0
5	0	0	0	0
6	0	1	0	0

The maximum is $A(\rho = 0, \theta = 135^\circ) = 4$ votes: the four aligned points have voted for the same cell. The other cells contain at most 2 votes (the outlier P_5 only shares a cell with one other point). The detected line is:

$$x \cos 135^\circ + y \sin 135^\circ = 0 \Leftrightarrow -x/\sqrt{2} + y/\sqrt{2} = 0 \Leftrightarrow y = x$$

which is the expected line. Figure 10 shows the five sinusoids: four of them intersect at $(\theta = 135^\circ, \rho = 0)$, while P_5 passes through that point nowhere.

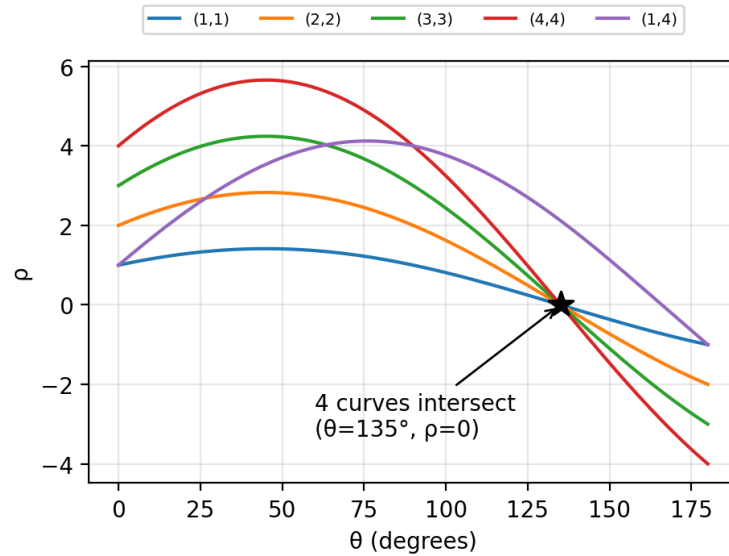


Figure 10: Sinusoids $\rho(\theta)$ of the five points of the example. The four curves of the aligned points meet at $(135^\circ, 0)$.

3.3.4. Advantages and Limitations of the Hough Transform

- It can detect lines even when contours are partially broken.
- It is relatively robust to missing points and moderate noise.
- It provides an explicit geometric representation of detected lines.
- Its computational cost can increase with image size and parameter resolution.
- The choice of accumulator resolution affects both accuracy and memory requirements.

3.3.5. numerical example

Step 1: Input Binary Edge Image

Suppose edge detection gives us a tiny 3×3 binary image containing **three edge points (pixels)**:

Points: $P_1(0,0)$, $P_2(1,1)$, $P_3(2,2)$

Visualizing the grid (where 1 is an edge pixel):

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

Goal: Use Hough Transform to find the line passing through these points.

Step 2: Define the Parameter Space (θ , r)

A line in polar coordinates is defined by:

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

Where:

- r : Perpendicular distance from the origin $(0, 0)$ to the line.
- θ : Angle of the normal line relative to the horizontal axis.

Angle (Degrees)	Angle (Radians)	$\sin(\theta)$	$\cos(\theta)$	$\tan(\theta)$
0°	0	0	1	0
45°	$\pi / 4$	$\sqrt{2} / 2$ (≈ 0.707)	$\sqrt{2} / 2$ (≈ 0.707)	1
90°	$\pi / 2$	1	0	Undefined
135°	$3\pi / 4$	$\sqrt{2} / 2$ (≈ 0.707)	$-\sqrt{2} / 2$ (≈ -0.707)	-1

Step 3: Vote in the Accumulator Matrix

For every edge point (x, y) , calculate r for each candidate θ and cast a vote in an accumulator table.

1. Point $P_1(0, 0)$:

- $\theta = 0^\circ \implies r = 0(1) + 0(0) = \mathbf{0}$
- $\theta = 45^\circ \implies r = 0(0.707) + 0(0.707) = \mathbf{0}$
- $\theta = 90^\circ \implies r = 0(0) + 0(1) = \mathbf{0}$
- $\theta = 135^\circ \implies r = 0(-0.707) + 0(0.707) = \mathbf{0}$

2. Point $P_2(1, 1)$:

- $\theta = 0^\circ \implies r = 1(1) + 1(0) = \mathbf{1}$
- $\theta = 45^\circ \implies r = 1(0.707) + 1(0.707) = \mathbf{1.414}$
- $\theta = 90^\circ \implies r = 1(0) + 1(1) = \mathbf{1}$
- $\theta = 135^\circ \implies r = 1(-0.707) + 1(0.707) = \mathbf{0}$

3. Point $P_3(2, 2)$:

- $\theta = 0^\circ \implies r = 2(1) + 2(0) = \mathbf{2}$
- $\theta = 45^\circ \implies r = 2(0.707) + 2(0.707) = \mathbf{2.828}$
- $\theta = 90^\circ \implies r = 2(0) + 2(1) = \mathbf{2}$
- $\theta = 135^\circ \implies r = 2(-0.707) + 2(0.707) = \mathbf{0}$

Step 4: Accumulator Matrix (Voting Table)

Count the number of votes each (r, θ) pair receives:

$r \setminus \theta$	0°	45°	90°	135°
$r = 0$	$1(P_1)$	$1(P_1)$	$1(P_1)$	$3(P_1, P_2, P_3)$
$r = 1$	$1(P_2)$	0	$1(P_2)$	0
$r = 1.414$	0	$1(P_2)$	0	0
$r = 2$	$1(P_3)$	0	$1(P_3)$	0
$r = 2.828$	0	$1(P_3)$	0	0

Step 5: Peak Detection & Line Reconstruction

Look for the peak (highest vote count) in the table:

Peak at $r = 0$, $\theta = 135^\circ$ (Votes = 3)

Reconstructing the Cartesian Line Equation:

Substituting $r = 0$ and $\theta = 135^\circ$ into $r = x \cos \theta + y \sin \theta$:

$$0 = x(-0.707) + y(0.707)$$

$$0.707x = 0.707y \implies \mathbf{y = x}$$

The algorithm correctly identifies that the three points belong to the diagonal line $y = x$.

4. Corner Extraction

4.1. Introduction

A corner is a local image point at which intensity changes significantly in more than one direction. Unlike an ideal straight edge, which mainly changes along one direction, a corner produces strong local variations in two approximately independent directions.

4.2. What Is a Corner?

Consider a small window centered on a candidate pixel. If the window can be shifted in one direction without producing a large change, the region behaves like an edge. If

large changes occur for shifts in several directions, the region is likely to contain a corner or another highly textured structure.

4.3. The Harris Corner Detector

4.3.1. Local Image Structure

The Harris detector analyzes how much the image changes when a small window is displaced. Let $I(x,y)$ be the image and let $w(x,y)$ be a window function. For a small displacement (u,v) , the change can be approximated by

$$E(u,v) = \sum w(x,y) [I(x+u,y+v) - I(x,y)]^2$$

Using a first-order Taylor approximation:

$$I(x+u,y+v) \approx I(x,y) + I_x u + I_y v$$

Substitution gives a quadratic form involving the image derivatives.

4.3.2. The Structure Tensor Matrix M

$$E(u,v) \approx [u \ v] M [u; v]$$

$$M = \begin{bmatrix} \sum w I_x^2 & \sum w I_x I_y \\ \sum w I_x I_y & \sum w I_y^2 \end{bmatrix}$$

The matrix M is called the second-moment matrix or structure tensor. Its eigenvalues describe the amount of intensity variation along two principal directions.

4.3.3. Harris Response Measure R

The Harris response combines the determinant and trace of M :

$$R = \det(M) - k[\text{trace}(M)]^2$$

$$\det(M) = \lambda_1 \lambda_2$$

$$\text{trace}(M) = \lambda_1 + \lambda_2$$

The parameter k is an empirical constant, commonly chosen in a small range such as 0.04–0.06.

4.3.4. Interpretation of the Harris Response

Local structure	Eigenvalue interpretation	Harris response tendency
Flat region	λ_1 small, λ_2 small	R small
Edge	One large, one small	R typically negative or small
Corner	Both large	R large positive

Table 6. Interpretation of local structures using the Harris detector.

4.3.5. Harris Corner Detection Procedure

- Convert the image to an intensity representation if necessary.
- Compute horizontal and vertical image derivatives.
- Form the products I_x^2 , I_y^2 , and $I_x I_y$.
- Smooth these products with a window, often Gaussian.
- Construct M for each pixel.
- Compute the Harris response R.
- Threshold R and apply local non-maximum suppression.

4.3.6. Numerical Example

Consider the following 5×5 image:

$$I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The bright square creates a corner around the transition between the dark and bright regions.

Step 1: Compute image gradients

Using simple derivative filters:

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

At pixels around the corner, both G_x and G_y are significant.

For example, suppose at the corner we obtain approximately:

$$G_x = 1, \quad G_y = 1$$

Step 2: Construct the Harris matrix

For a small neighborhood around the pixel:

$$M = \begin{bmatrix} \sum G_x^2 & \sum G_x G_y \\ \sum G_x G_y & \sum G_y^2 \end{bmatrix}$$

For simplicity, assume the neighborhood gives:

$$\sum G_x^2 = 4, \quad \sum G_y^2 = 4, \quad \sum G_x G_y = 1$$

Therefore:

$$M = \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}$$

Step 3: Calculate the Harris response

The Harris response is

$$\text{where typically } R = \det(M) - k(\text{trace}(M))^2$$

$$\text{First: } k = 0.04$$

$$\text{and } \det(M) = (4)(4) - (1)(1) = 15$$

$$\text{trace}(M) = 4 + 4 = 8$$

$$\text{Thus: } R = 15 - 0.04(8)^2$$

$$R = 15 - 2.56 = 12.44$$

Step 4: Interpret the result

$$R = 12.44 > 0$$

A large positive Harris response indicates a corner.

Harris response R	Interpretation
Large positive	Corner
Small	Flat region
Negative	Edge

Simple idea

At a corner, the image intensity changes significantly in both horizontal and vertical directions:

$$\text{Corner} \Rightarrow G_x \text{ large and } G_y \text{ large}$$

4.3.7. Advantages and Limitations of Harris Detection

- It is computationally efficient and mathematically well founded.
- It responds strongly to image structures that vary in two directions.
- It is reasonably robust to moderate image transformations.
- The detected position depends on scale, window size, smoothing, and threshold.
- The original Harris detector is not inherently scale invariant.

4.3.8. Applications of Corner Detection

Corner features are useful in image matching, registration, object tracking, motion estimation, panorama construction, and visual localization. Corners are particularly valuable because they provide localized structures that can be distinguished from long, repetitive edges.

5. References

R. C. Gonzalez and R. E. Woods, Digital Image Processing, Pearson.

J. Canny, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986.

C. G. Harris and M. Stephens, "A Combined Corner and Edge Detector," Alvey Vision Conference, 1988.

P. V. C. Hough, "Method and Means for Recognizing Complex Patterns," U.S. Patent, 1962.

R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Communications of the ACM, 1972.