Chapter 3: Feature Extraction

Computer Vision – Unit 03

Computer Vision Course

February 2026

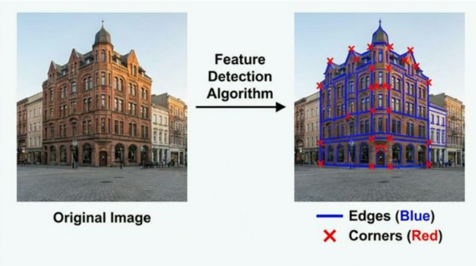
Table of Contents

What is Feature Extraction?

- **Feature extraction** is the process of identifying and isolating meaningful patterns, structures, or descriptors from raw image data.
- Goal: Reduce data dimensionality while retaining discriminative information.
- Common features: **Edges, Corners, Blobs, Ridges, Textures**.

Why Feature Extraction?

- Enables object recognition, tracking, image stitching, 3D reconstruction.
- Forms the foundation of many computer vision pipelines.



Taxonomy of Features

- **Edges**: Boundaries between regions (Canny, LOG, DOG)
- **Lines**: Straight structures (Hough Transform)
- **Corners / Interest Points**: Harris, Hessian Affine
- **Descriptors**: SIFT, SURF, HOG, GLOH
- **Multi-scale**: Scale-Space, Image Pyramids
- **Frequency-based**: Gabor Filters, DWT

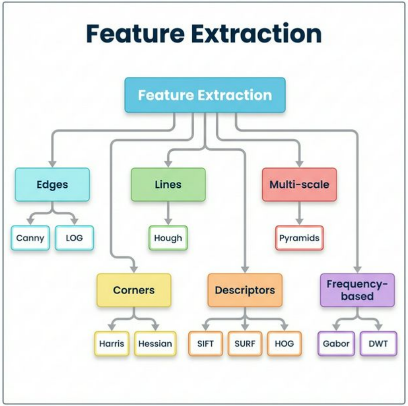


Figure: Hierarchy chart showing feature types branching from “Feature Extraction”

Edge Detection – Overview

- An **edge** is a significant local change in image intensity.
- Edges correspond to object boundaries, surface markings, shadows, or depth discontinuities.
- Mathematically:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

- Magnitude: $|\nabla I| = \sqrt{I_x^2 + I_y^2}$
- Direction: $\theta = \arctan(I_y / I_x)$

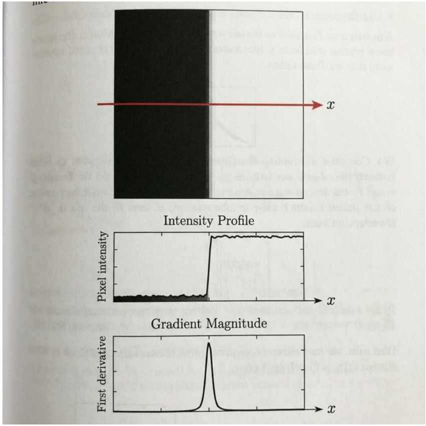


Figure: Image showing intensity profile across an edge with gradient plot

Canny Edge Detector

Canny (1986) – Optimal Edge Detector

Designed to satisfy three criteria: **Good detection**, **Good localisation**, **Single response**.

Steps of the Canny Algorithm:

1. **Gaussian Smoothing**: Convolve with G_σ to reduce noise.

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

2. **Gradient Computation**: Compute $|\nabla I|$ and θ using Sobel or similar operators.
3. **Non-Maximum Suppression (NMS)**: Thin edges by keeping only local maxima along gradient direction.
4. **Double Thresholding**: Use T_{high} and T_{low} to classify strong, weak, and non-edges.
5. **Edge Tracking by Hysteresis**: Weak edges connected to strong edges are kept.

Canny Edge Detector – Illustration

Key Parameters:

- σ – controls smoothing level.
- $T_{\text{high}}, T_{\text{low}}$ – thresholds for hysteresis.
- Typical ratio: $T_{\text{high}}/T_{\text{low}} \approx 2:1$ or $3:1$.

Important

Larger $\sigma \Rightarrow$ fewer edges detected (more smoothing).
Smaller $\sigma \Rightarrow$ more edges but also more noise.

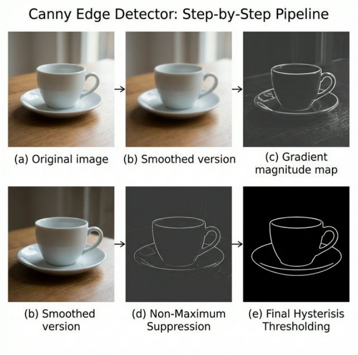


Figure: Step-by-step Canny pipeline – (a) Original (b) Smoothed (c) Gradient (d) NMS (e) Final edges

Laplacian of Gaussian (LOG)

- The **Laplacian** $\nabla^2 I$ is a second-order derivative operator:

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

- Direct Laplacian is very sensitive to noise $\Rightarrow$ first smooth with Gaussian.
- **LOG** = Laplacian applied to Gaussian-smoothed image:

$$\mathrm{LOG}(x, y) = \nabla^2 \left[G_\sigma * I \right] = \left[\nabla^2 G_\sigma \right] * I$$

- The LOG kernel ("Mexican Hat"):

$$\nabla^2 G_\sigma(x, y) = \frac{1}{\pi \sigma^4} \left(\frac{x^2 + y^2}{2\sigma^2} - 1 \right) \exp\left(- \frac{x^2 + y^2}{2\sigma^2} \right)$$

- Edges are located at **zero-crossings** of the LOG response.

LOG – Visualization

Properties of LOG:

- Isotropic (rotationally symmetric).
- Detects edges at zero-crossings.
- Scale controlled by σ.
- Sensitive to blobs at scale σ.

Typical 5×5 LOG Mask ($\sigma = 1.4$)

$$
\begin{bmatrix}
0 & 0 & -1 & 0 & 0 \\
0 & -1 & -2 & -1 & 0 \\
-1 & -2 & 16 & -2 & -1 \\
0 & -1 & -2 & -1 & 0 \\
0 & 0 & -1 & 0 & 0
\end{bmatrix}
$$

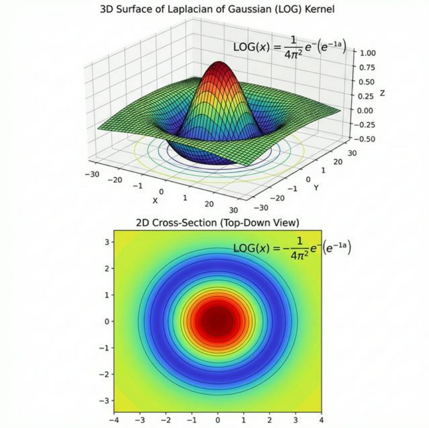


Figure: 3D surface plot of LOG kernel (Mexican Hat shape) and its 2D cross-section

Difference of Gaussians (DOG)

- **DOG** approximates the LOG and is computationally cheaper:

$$\mathrm{DOG}(x, y) = G_{\sigma_1} - G_{\sigma_2}, \quad \sigma_2 > \sigma_1$$

- Typically $\sigma_2 = k\sigma_1$ with $k \approx 1.6$.
- DOG $\approx$ LOG because:

$$G_{\sigma_1} - G_{\sigma_2} \approx (\sigma_2 - \sigma_1)\nabla^2 G_\sigma$$

- **Advantages**: Faster computation (subtract two blurred images), directly gives scale information.
- Used extensively in **SIFT** for keypoint detection.

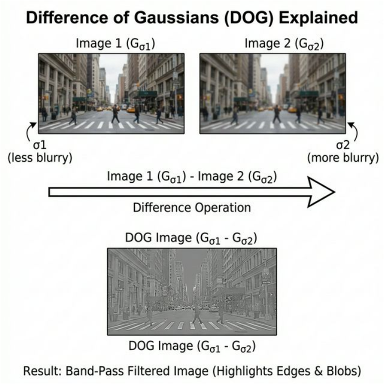


Figure: Two Gaussian-blurred images and their difference showing blob-like edge response

Edge Detectors – Comparison

Property	Canny	LOG	DOG
Derivative Order	1st	2nd	$\approx$ 2nd
Edge Criterion	Gradient max	Zero-crossing	Zero-crossing
Noise Handling	Gaussian smooth	Gaussian smooth	Inherent
Computational Cost	Moderate	High	Low
Thin Edges	Yes (NMS)	Yes	Approximate
Multi-scale	Manual σ	σ param	Natural

Key Takeaway

Canny is the most widely used general-purpose edge detector. LOG gives precise zero-crossings. DOG is a fast approximation used in scale-space frameworks like SIFT.

Hough Transform – Concept

- **Hough Transform** detects parametric shapes (lines, circles, ellipses) via a **voting scheme** in parameter space.
- For a line $y = mx + c$, each edge point votes for all (m, c) pairs passing through it.
- Problem: $m \to \infty$ for vertical lines.

Normal (Polar) Parameterization

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

where ρ = perpendicular distance from origin, θ = angle of normal.
Parameter space: (ρ, θ) with $\rho \in [-D, D]$, $\theta \in [0, \pi)$.

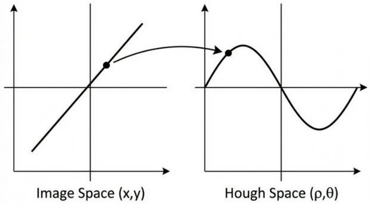


Figure: Line in image space mapped to sinusoidal curve in (ρ, θ) Hough space

Hough Transform – Algorithm

Algorithm Steps:

1. Run edge detection (e.g., Canny) on input image.
2. Create an **accumulator array** $A[\rho][\theta]$ initialized to 0.
3. For each edge pixel (x_i, y_i):
   - For each θ from 0 to π (quantized):
   - Compute $\rho = x_i \cos\theta + y_i \sin\theta$
   - Increment $A[\rho][\theta]$
4. Find **peaks** in the accumulator $\Rightarrow$ detected lines.

Complexity

$O(n \cdot q_\theta)$ where n = number of edge pixels, q_θ = quantization levels.

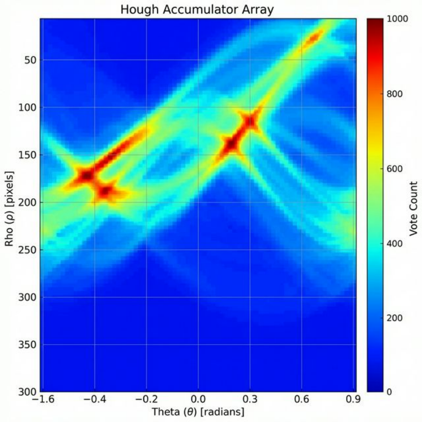


Figure: Accumulator array heatmap with bright peaks at detected line parameters

Hough Transform – Example

Example: Three collinear points $(1,1)$, $(2,2)$, $(3,3)$.

For each point, the Hough curve is:

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

- Point $(1,1)$: $\rho = \cos\theta + \sin\theta$
- Point $(2,2)$: $\rho = 2\cos\theta + 2\sin\theta$
- Point $(3,3)$: $\rho = 3\cos\theta + 3\sin\theta$

All three curves intersect at $\theta = 45$, confirming the line $y = x$.
At $\theta = 45$: $\rho = \sqrt{2}, 2\sqrt{2}, 3\sqrt{2}$ – Wait, they intersect where all produce same (ρ, θ), which is at $\theta = 135$, $\rho = 0$ (line through origin).

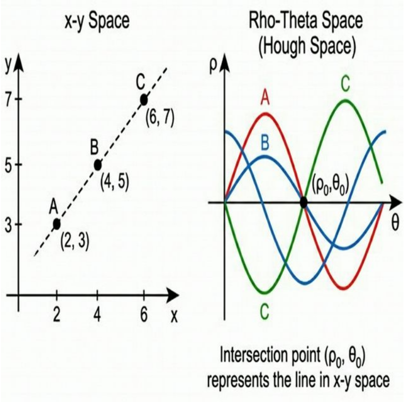


Figure: Sinusoidal Hough curves for three collinear points intersecting at a single (ρ, θ) peak

Harris Corner Detector – Motivation

- **Corners** are points where intensity changes significantly in *all* directions.
- Consider a small window shifted by (u, v). The **Sum of Squared Differences (SSD)**:

$$E(u, v) = \sum_{(x,y) \in W} w(x, y) \left[I(x + u, y + v) - I(x, y) \right]^2$$

- Using Taylor expansion $I(x + u, y + v) \approx I(x, y) + I_x u + I_y v$:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

Structure Tensor (Second Moment Matrix)

$$\mathbf{M} = \sum_{(x,y) \in W} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Harris Corner Detector – Response Function

- Let λ_1, λ_2 be eigenvalues of $\mathbf{M}$:
  - **Flat**: $\lambda_1 \approx \lambda_2 \approx 0$
  - **Edge**: One $\lambda \gg 0$, other ≈ 0
  - **Corner**: Both λ_1, λ_2 large
- **Harris response** (no eigenvalues needed):

$$R = \det(\mathbf{M}) - k\,(\mathrm{trace}(\mathbf{M}))^2$$

where $\det(\mathbf{M}) = \lambda_1\lambda_2$, $\mathrm{trace}(\mathbf{M}) = \lambda_1 + \lambda_2$, $k \in [0.04, 0.06]$.

- Decision:
  - $R > T \Rightarrow$ **Corner**
  - $R < 0, |R|$ large $\Rightarrow$ **Edge**
  - $|R|$ small $\Rightarrow$ **Flat**

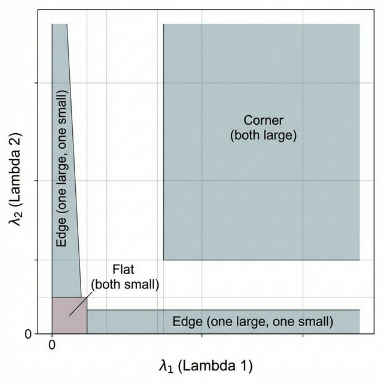


Figure: λ_1 vs λ_2 plot showing corner, edge, and flat regions

Harris Corner Detector – Properties

Algorithm Steps:

1. Compute image gradients I_x, I_y.
2. Compute products I_x^2, I_y^2, I_xI_y.
3. Apply Gaussian window to each product.
4. Compute Harris response R at each pixel.
5. Apply non-maximum suppression.
6. Threshold $R > T$ to select corners.

Properties:

- **Rotation invariant** (eigenvalues unchanged).
- **NOT scale invariant**.
- Partially invariant to affine intensity changes.

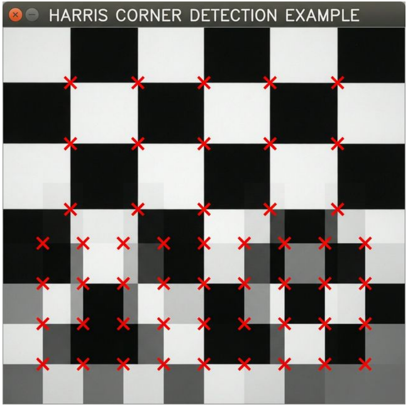


Figure: Image with detected Harris corners marked as red dots overlaid on a checkerboard pattern

Hessian Affine Detector

- The **Hessian matrix** at point (x, y) at scale σ:

$$\mathbf{H}(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}$$

  where $L_{xx} = \sigma^2 \frac{\partial^2}{\partial x^2}(G_\sigma * I)$, etc.
- **Blob/corner detection**: $\det(\mathbf{H}) = L_{xx}L_{yy} - L_{xy}^2 > T$
- **Hessian-Affine** extends this with:
  1. Multi-scale detection using Hessian determinant.
  2. **Affine adaptation**: Iteratively estimate local affine shape using the second moment matrix.
  3. Normalize the region to a canonical circular shape.

Key Advantage

Provides **affine-invariant** interest regions, enabling matching under viewpoint changes.

Harris vs Hessian Affine – Comparison

Property	**Harris**	**Hessian Affine**
Detects	Corners	Blobs + Corners
Matrix	Structure Tensor **M**	Hessian **H**
Scale Inv.	No	Yes (multi-scale)
Affine Inv.	No	Yes (iterative)
Comp.	Fast	Moderate–High
Repeat.	Good	Very Good

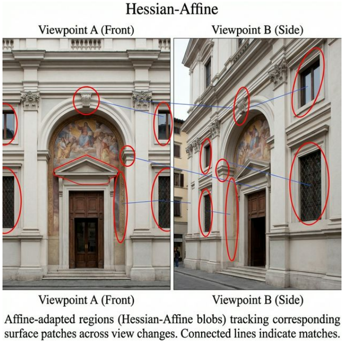


Figure: Hessian-Affine regions matched across views

Orientation Histogram

- An **Orientation Histogram** captures distribution of gradient directions.
- Steps:
  1. Compute gradient magnitude m and orientation θ.
  2. Quantize θ into n bins (e.g., $n = 36$ bins).
  3. Vote weighted by m.
- **Dominant orientation** = bin with max count.
- Used in SIFT $\Rightarrow$ rotation invariance.

Example

If gradients mostly point at 45, the histogram peak is in the $[40, 50]$ bin.

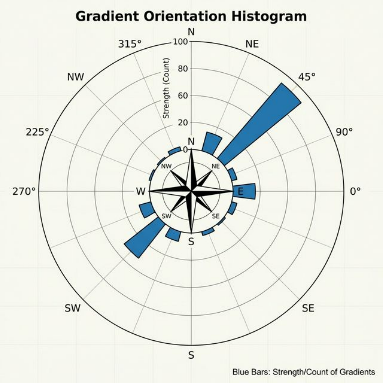


Figure: Polar histogram of gradient orientations with a dominant peak at 45

SIFT – Overview (Lowe, 2004)

Goal

Detect and describe features invariant to **scale, rotation**, robust to affine/illumination.

Four Major Steps:

1. **Scale-space extrema**: DOG pyramid.
2. **Keypoint localization**: Sub-pixel.
3. **Orientation assignment**: Histograms.
4. **Keypoint descriptor**: 128-D vector.

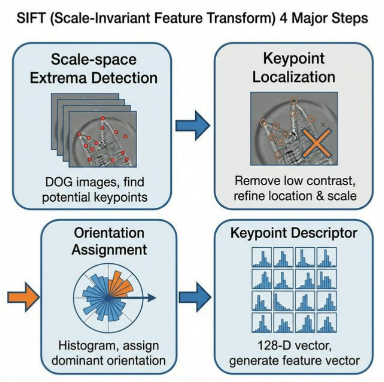


Figure: SIFT pipeline diagram

SIFT – Scale-Space Extrema Detection

- Build **Gaussian scale space**.
- Compute **DOG** images.
- Group into **octaves**.
- Detect **local extrema**: 26 neighbours.

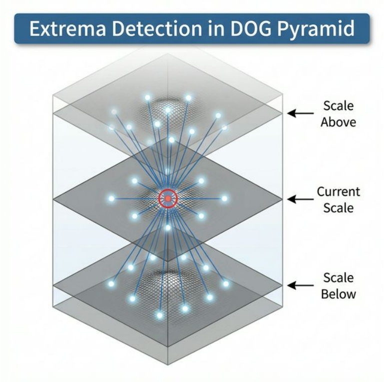


Figure: DOG pyramid extrema

SIFT – Keypoint Localization & Orientation

Keypoint Localization:

- Fit 3D quadratic (Taylor expansion) to refine position to sub-pixel accuracy:

$$D(\mathbf{x}) \approx D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

- Reject low-contrast points: $|D(\hat{\mathbf{x}})| < 0.03$.
- Reject edges using ratio of principal curvatures (Hessian eigenvalue ratio).

Orientation Assignment:

- Build a 36-bin orientation histogram around each keypoint (weighted by magnitude and Gaussian window).
- Dominant peak $\Rightarrow$ keypoint orientation.
- Any peak $\geq 80\%$ of max creates an *additional* keypoint.

SIFT – Descriptor

- Take 16×16 window, rotate.
- Divide into 4×4 sub-regions.
- Compute 8-bin histograms.
- **128**-dim descriptor.
- Normalize and clamp.

Matching

Use **Euclidean distance**. Ratio test < 0.8.

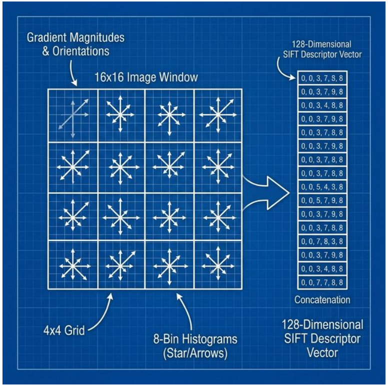


Figure: 128-D SIFT Descriptor

SURF – Overview (Bay et al., 2006)

- **SURF** is faster (approximations).
- Key ideas:
  1. **Integral images**.
  2. **Hessian-based** detection.
  3. **Haar wavelet** responses.

Integral Image

Sum computed in **constant time**.

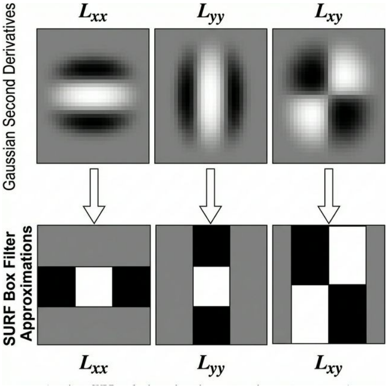


Figure: Box filter approximations

SURF – Detection and Description

Interest Point Detection:

- Approximate Hessian determinant using box filters at multiple scales.
- Non-maximum suppression in $3 \times 3 \times 3$ neighbourhood.
- Sub-pixel interpolation for localization.

Orientation Assignment:

- Compute Haar wavelet responses in x and y within radius $6s$.
- Sliding window of $\pi/3$ finds dominant orientation.

Descriptor:

- 4×4 sub-regions around keypoint.
- Each sub-region: $(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \Rightarrow 4$ values.
- Total: $4 \times 4 \times 4 =$ **64**-dimensional descriptor.

SIFT vs SURF – Comparison

Property	SIFT	SURF
Detector	DOG	Hessian (approx)
Dim.	128	64
Speed	Slow	$\sim 3\times$ Faster
Scale Inv.	Yes	Yes
Rot. Inv.	Yes	Yes
Robust	Excellent	Good

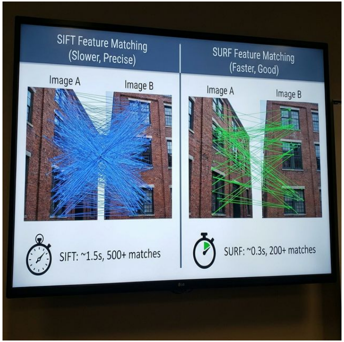


Figure: SIFT (left) vs SURF (right)

HOG – Overview (Dalal & Triggs, 2005)

- **HOG** describes local shape via distribution of gradient directions.
- Originally designed for **pedestrian detection**.
- Key idea: Object appearance can be characterized by the distribution of local intensity gradients, even without precise knowledge of positions.

HOG Pipeline:

1. Compute gradients ($[-1, 0, 1]$ filter).
2. Divide image into **cells** (e.g., 8×8 pixels).
3. Compute 9-bin orientation histogram per cell (unsigned: 0–180).
4. Group cells into overlapping **blocks** (e.g., 2×2 cells).
5. Normalize each block (L2-norm) $\Rightarrow$ illumination invariance.
6. Concatenate all block descriptors.

HOG – Descriptor Construction

Example Calculation:

- Image window: 64×128 pixels.
- Cell size: $8 \times 8 \Rightarrow 8 \times 16$ cells.
- Block size: 2×2 cells with stride 1 cell.
- Number of blocks: $7 \times 15 = 105$ blocks.
- Each block: $2 \times 2 \times 9 = 36$ values.
- Total descriptor: $105 \times 36 = \mathbf{3780}$ dimensions.

Key Properties

- Captures edge/gradient structure.
- Block normalization handles illumination.
- Dense, overlapping computation.

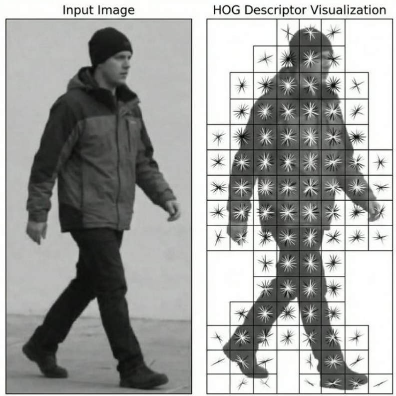


Figure: HOG visualization – image of a person with gradient orientation "hedgehog" glyphs overlaid on cells

GLOH – Gradient Location and Orientation Histogram

- **GLOH** (Mikolajczyk & Schmid, 2005): SIFT extension.
- Uses **log-polar grid**:
  - 3 radial bins (6, 11, 15 px).
  - 8 angular bins (outer), 1 central.
  - Total: 17 spatial bins.
- 16-bin orientation histogram ⇒ 272 dims.
- **PCA** reduction to **128 dims**.

Advantage over SIFT

Log-polar grid better captures radial structure, higher distinctiveness.

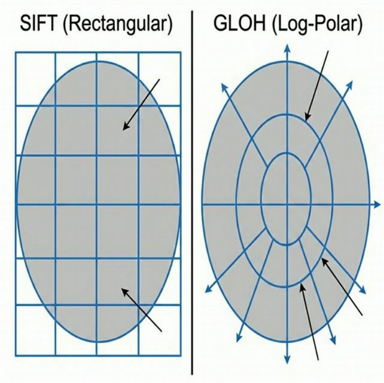


Figure: SIFT (Left) vs GLOH (Right)

Scale-Space Analysis – Motivation

- Real-world objects exist at **multiple scales**. A feature detector must handle this.
- **Scale space**: Family of smoothed images $L(x, y, \sigma) = G_\sigma * I(x, y)$ parametrized by scale σ.
- The Gaussian is the **unique** kernel that satisfies the scale-space axioms (linearity, shift invariance, semi-group, non-enhancement of local extrema).
- Scale-space representation allows detecting structures at their **characteristic scale**.

Scale-Space Axiom

$$L(x, y, \sigma) = G(x, y; \sigma) * I(x, y), \quad \frac{\partial L}{\partial \sigma} = \sigma \nabla^2 L$$

This is the **diffusion equation** – the scale parameter acts like time in heat diffusion.

Image Pyramids

- **Image pyramid**: multi-resolution.
- **Gaussian Pyramid**: Smooth/subsample ($\downarrow 2$).
  - $G_0 = I$
  - $G_{l+1} = \text{subsample}(G_\sigma * G_l)$
- **Laplacian Pyramid**: Band-pass details.

$$Lap_l = G_l - \text{upsample}(G_{l+1})$$

- **Overcomplete**, perfect recon.

Applications

Compression, blending, multi-scale.

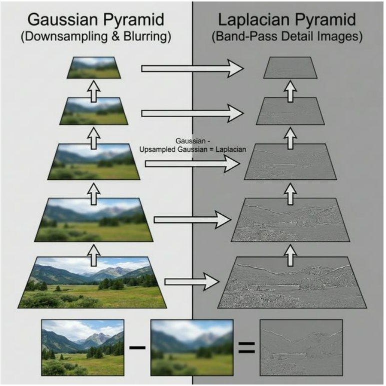


Figure: Gaussian (left) Laplacian (right)

Gaussian Derivative Filters

- **Gaussian derivative filters** combine smoothing and differentiation in one step:
$$\frac{\partial}{\partial x}(G_\sigma * I) = \left(\frac{\partial G_\sigma}{\partial x}\right) * I$$
- First-order derivatives of Gaussian:
$$G_x = \frac{\partial G}{\partial x} = -\frac{x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$
- Second-order: G_{xx}, G_{yy}, G_{xy} – used in Hessian-based detection.
- These form a **steerable filter** basis: any rotated derivative can be expressed as a linear combination of basis filters.

Scale-Space Feature Detection Summary

- **Scale selection**: Detect features at the scale where a normalized derivative response is maximal.
- **Normalized derivatives**: $\sigma^n \nabla^n G_\sigma$ ensures comparable response across scales.
- **Characteristic scale**: The scale σ^* at which the normalized response achieves its maximum.

$$\sigma^* = \arg\max_{\sigma} \left|\sigma^2 \nabla^2 L(x, y, \sigma)\right|$$

Multi-Scale Detection Pipeline

1. Build scale-space (Gaussian pyramid or dense scales).
2. Compute normalized feature response at each scale.
3. Detect extrema across both space *and* scale.
4. Extract features at their characteristic scale.

Gabor Filters – Definition

- A **Gabor filter** is a Gaussian modulated by a sinusoidal wave:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$

  where $\quad x' = x\cos\theta + y\sin\theta, \quad y' = -x\sin\theta + y\cos\theta.$
- Parameters:
  - λ – wavelength of sinusoidal component.
  - θ – orientation of the filter.
  - ψ – phase offset.
  - σ – Gaussian envelope width.
  - γ – spatial aspect ratio.

Gabor Filters – Properties and Applications

Properties:

- Achieve optimal **joint resolution** in space and frequency (uncertainty principle).
- Mimic simple cells in the **human visual cortex**.
- Selective to **orientation** and **spatial frequency**.

Gabor Filter Bank:

- Apply filters at multiple orientations (θ) and scales (λ).
- Typically: 4–8 orientations $\times$ 3–5 scales.
- Feature vector = concatenated filter responses.

Applications:

- Texture analysis and segmentation.
- Face recognition, fingerprint enhancement.
- Character recognition (OCR).

DWT – Discrete Wavelet Transform

- **DWT** decomposes an image into frequency sub-bands at multiple scales.
- Uses a pair of filters: **low-pass** (scaling function ϕ) and **high-pass** (wavelet ψ).
- For 2D images, apply row-wise then column-wise:

LL (Approximation)	**LH** (Horizontal detail)
HL (Vertical detail)	**HH** (Diagonal detail)

- **LL** = low-pass in both directions (coarse approximation).
- **LH, HL, HH** = detail coefficients capturing edges at different orientations.
- Multi-level DWT: recursively decompose the LL sub-band.

DWT – Haar Wavelet Example

- **Haar wavelet** is the simplest wavelet:

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 0.5 \\ -1 & 0.5 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

- Low-pass filter: $h = \frac{1}{\sqrt{2}}[1, 1]$, High-pass filter: $g = \frac{1}{\sqrt{2}}[1, -1]$.

1D Haar DWT Example

Signal: $[4, 6, 10, 12]$

Averages (LL): $\frac{4+6}{2} = 5$, $\frac{10+12}{2} = 11 \Rightarrow [5, 11]$

Differences (HL): $\frac{4-6}{2} = -1$, $\frac{10-12}{2} = -1 \Rightarrow [-1, -1]$

DWT coefficients: $[5, 11 \mid -1, -1]$

Applications in CV: Texture classification, image compression (JPEG 2000), feature extraction for recognition.

Gabor vs DWT – Comparison

Property	Gabor Filters	DWT
Basis	Non-orthogonal	Orthogonal
Redundancy	High (overcomplete)	Minimal (critical sampling)
Orientation Selectivity	Continuous θ	Fixed (H, V, D)
Frequency Resolution	Tunable λ	Dyadic scales
Reconstruction	Approximate	Perfect
Computation	Moderate	Fast (pyramidal)
Bio-inspired	Yes (V1 cortex)	No

When to Use

Gabor: When fine orientation and frequency tuning is needed (face, texture).
DWT: When compact, non-redundant multi-scale representation is needed (compression, fast features).

Derivation – Harris Corner Response Function

Goal: Derive the Harris corner response R from the auto-correlation function.

Step 1: Define the weighted SSD for a shift (u, v):

$$E(u, v) = \sum_{(x,y)} w(x, y) \left[I(x + u, y + v) - I(x, y) \right]^2$$

Step 2: Apply first-order Taylor expansion:

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

$$\Rightarrow E(u, v) \approx \sum_{(x,y)} w(x, y) \left[I_x\, u + I_y\, v \right]^2$$

Step 3: Expand the square:

$$E(u, v) \approx \sum w(x, y) \left[I_x^2\, u^2 + 2 I_x I_y\, uv + I_y^2\, v^2 \right]$$

Derivation – Harris Corner Response (cont.)

Step 4: Write in matrix form:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\sum_{(x,y)} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} u \\ v \end{bmatrix}$$

Step 5: Analyze eigenvalues λ_1, λ_2 of $\mathbf{M}$:

- $E(u, v)$ is an ellipse with axes λ_1, λ_2.
- Both large $\Rightarrow$ corner; one large $\Rightarrow$ edge; both small $\Rightarrow$ flat.

Step 6: Define Harris response (avoids eigenvalue computation):

$$\boxed{R = \det(\mathbf{M}) - k\,[\mathrm{trace}(\mathbf{M})]^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2}$$

where $k \in [0.04, 0.06]$. **Corner**: $R > 0$ (large); **Edge**: $R < 0$; **Flat**: $|R| \approx 0$. □

Numerical 1: Gradient at a Pixel

Problem

Given a 3×3 image patch: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix}$.

Find I_x and I_y at center pixel using: $I_x = I(\text{right}) - I(\text{left})$, $I_y = I(\text{bottom}) - I(\text{top})$.

Solution:

- Center pixel value $= 1$
- $I_x = I(\text{right}) - I(\text{left}) = 3 - 1 = \mathbf{2}$
- $I_y = I(\text{bottom}) - I(\text{top}) = 1 - 1 = \mathbf{0}$
- Magnitude: $|\nabla I| = \sqrt{2^2 + 0^2} = \sqrt{4} = \mathbf{2}$
- Direction: $\theta = \arctan(0/2) = \arctan(0) = \mathbf{0}$

Meaning

Gradient is purely horizontal ($\theta = 0$), so there is a **vertical edge** to the right of the center pixel.

Numerical 2: Harris Corner Response

Problem

Given: $\mathbf{M} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$, $k = 0.04$. Find R and classify.

Solution:

- $\det(\mathbf{M}) = 5 \times 5 - 0 \times 0 = \mathbf{25}$
- $\text{trace}(\mathbf{M}) = 5 + 5 = \mathbf{10}$
- $R = \det - k \cdot (\text{trace})^2 = 25 - 0.04 \times 100 = 25 - 4 = \boxed{21}$
- $R > 0 \Rightarrow$ **Corner** ✓

If instead $\mathbf{M} = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}$:

- $\det = 0$, $\text{trace} = 10$, $R = 0 - 0.04 \times 100 = -4 \Rightarrow$ **Edge**

If instead $\mathbf{M} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$:

- $R = 0 \Rightarrow$ **Flat region**

Numerical 3: Hough Transform – Compute ρ

Problem

An edge point is at $(x, y) = (1, 0)$. Compute ρ for $\theta = 0$ and $\theta = 90$.

Solution:

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

- At $\theta = 0$:

$$\rho = 1 \cdot \cos 0 + 0 \cdot \sin 0 = 1 \times 1 + 0 = \boxed{1}$$

- At $\theta = 90$:

$$\rho = 1 \cdot \cos 90 + 0 \cdot \sin 90 = 1 \times 0 + 0 = \boxed{0}$$

Meaning

At $\theta = 0$, the line is vertical at $x = 1$. At $\theta = 90$, the line is horizontal passing through the origin. Each edge point traces a curve in (ρ, θ) space.

Numerical 4: SIFT & SURF Descriptor Size

Problem

Calculate the SIFT and SURF descriptor sizes.

SIFT:

- Sub-regions: $4 \times 4 = 16$
- Bins per sub-region: 8
- Total $= 16 \times 8 = \boxed{128}$ dimensions

SURF:

- Sub-regions: $4 \times 4 = 16$
- Values per sub-region: 4 $\quad (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$
- Total $= 16 \times 4 = \boxed{64}$ dimensions

Key Takeaway

SURF descriptor is **half** the size of SIFT $\Rightarrow$ faster to compute and match, but slightly less distinctive.

Numerical 5: Image Pyramid Sizes

Problem

An image is 128×128. Build a 4-level Gaussian pyramid (subsample by 2). What is the size at each level?

Solution:

Level	**Size**	**Pixels**
0	128×128	16,384
1	64×64	4,096
2	32×32	1,024
3	16×16	256

Total pixels $= 16384 + 4096 + 1024 + 256 = \mathbf{21{,}760}$

Overhead $= \frac{21760}{16384} \approx 1.33\times$ original $\Rightarrow$ only 33% extra memory!

Numerical 6: Haar DWT (4 values)

Problem

Apply 1 level of Haar DWT to $f = [6, 2, 8, 4]$.

Solution:

Averages (approximation):

- $a_1 = \frac{6+2}{2} = \mathbf{4}, \quad a_2 = \frac{8+4}{2} = \mathbf{6}$

Differences (detail):

- $d_1 = \frac{6-2}{2} = \mathbf{2}, \quad d_2 = \frac{8-4}{2} = \mathbf{2}$

Output: $[4, 6 \mid 2, 2]$

Reconstruct:

$a_1 + d_1 = 4 + 2 = 6 \checkmark, \quad a_1 - d_1 = 4 - 2 = 2 \checkmark$
$a_2 + d_2 = 6 + 2 = 8 \checkmark, \quad a_2 - d_2 = 6 - 2 = 4 \checkmark$

Long Numerical 1: Canny Edge Detection (Step-by-Step)

Problem

Given a 5×5 image. After Gaussian smoothing, the center 3×3 gradient magnitudes and directions are:

Gradient Magnitude:

$$\begin{bmatrix} 2 & 8 & 3 \\ 1 & 10 & 4 \\ 2 & 7 & 3 \end{bmatrix}$$

Gradient Direction:

$$\begin{bmatrix} 45 & 90 & 45 \\ 0 & 90 & 0 \\ 135 & 90 & 135 \end{bmatrix}$$

Apply: (a) Non-Maximum Suppression (NMS), (b) Double thresholding with $T_H = 7$, $T_L = 3$.

Step 1 – NMS at center pixel (magnitude $= 10$, direction $= 90$):
Compare along gradient direction (vertical): neighbours are 8 (top) and 7 (bottom).
$10 > 8$ and $10 > 7 \Rightarrow$ **keep** center pixel (it is a local max).

Long Numerical 1: Canny (cont.)

Step 1 (cont.) – NMS for all pixels:

Pixel	Mag	Dir	After NMS
$(0,0)$: mag=2	45	compare diag: 10, –	suppressed ($2 < 10$)
$(0,1)$: mag=8	90	compare vert: –, 10	suppressed ($8 < 10$)
$(1,0)$: mag=1	0	compare horiz: –, 10	suppressed ($1 < 10$)
$(1,1)$: mag=10	90	compare: 8, 7	**kept** ($10 >$ both)
$(2,1)$: mag=7	90	compare: 10, –	suppressed ($7 < 10$)

Step 2 – Thresholding ($T_H = 7,\ T_L = 3$):

- Center pixel: mag $= 10 > T_H = 7 \Rightarrow$ **Strong edge** ✓
- All other pixels were suppressed in NMS $\Rightarrow$ not edges.

Result: Only the center pixel is detected as a **strong edge point**.

Long Numerical 2: Harris Corner Detection (Full Example)

Problem

A 3×3 image window has pixel values: $I = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 5 \\ 1 & 5 & 5 \end{bmatrix}$.

Compute Harris response at center pixel with $k = 0.05$ (use simple differences, no Gaussian weighting for simplicity).

Step 1 – Compute gradients at each pixel (using right-left, bottom-top):

I_x **(right − left):**
Only center has both neighbours:
$I_x = I(1,2) - I(1,0) = 5 - 1 = 4$

I_y **(bottom − top):**
$I_y = I(2,1) - I(0,1) = 5 - 1 = 4$

Step 2 – Form structure tensor M at center (single pixel, no window sum needed):

$$\mathbf{M} = \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} 16 & 16 \\ 16 & 16 \end{bmatrix}$$

Long Numerical 2: Harris Corner (cont.)

Step 3 – Compute $\det(\mathbf{M})$ and $\mathrm{trace}(\mathbf{M})$:

- $\det(\mathbf{M}) = 16 \times 16 - 16 \times 16 = 256 - 256 = \mathbf{0}$
- $\mathrm{trace}(\mathbf{M}) = 16 + 16 = \mathbf{32}$

Step 4 – Compute Harris response:

$$R = \det(\mathbf{M}) - k \cdot [\mathrm{trace}(\mathbf{M})]^2 = 0 - 0.05 \times 32^2 = 0 - 0.05 \times 1024$$

$$\boxed{R = -51.2}$$

Step 5 – Classification:

- $R < 0$ (negative) $\Rightarrow$ **Edge point**, not a corner.

Interpretation

Since $I_x = I_y = 4$ and both are equal, the gradient points in one direction (45). The eigenvalues are $\lambda_1 = 32, \lambda_2 = 0$, meaning intensity changes along only one direction $\Rightarrow$ this is an **edge**, not a corner.

Chapter Summary

1. **Edge Detection**: Canny (optimal multi-step), LOG (zero-crossings), DOG (fast approx.).
2. **Line Detection**: Hough Transform – voting in (ρ, θ) parameter space.
3. **Corner Detection**: Harris (structure tensor $\mathbf{M}$, response R); Hessian Affine (scale + affine invariant).
4. **Orientation Histogram**: Gradient direction distribution $\Rightarrow$ rotation invariance.
5. **SIFT**: 128-D descriptor, scale & rotation invariant (DOG + gradient histograms).
6. **SURF**: 64-D, faster SIFT alternative (integral images + Haar wavelets).
7. **HOG**: Dense gradient histograms with block normalization (pedestrian detection).
8. **GLOH**: Log-polar SIFT variant + PCA for compactness.
9. **Scale-Space**: Gaussian pyramids, derivative filters for multi-scale analysis.
10. **Gabor & DWT**: Frequency-domain features for texture analysis.

References

- J. Canny, “A Computational Approach to Edge Detection,” *IEEE TPAMI*, 1986.
- C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Alvey Vision Conf.*, 1988.
- D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *IJCV*, 2004.
- H. Bay et al., “SURF: Speeded Up Robust Features,” *ECCV*, 2006.
- N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” *CVPR*, 2005.
- K. Mikolajczyk and C. Schmid, “A Performance Evaluation of Local Descriptors,” *IEEE TPAMI*, 2005.
- R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall.
- R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer.