African Institute for Mathematical Sciences — Cameroon
Structured Master's, Data Science Stream · Computer Vision

Edges, Corners and Descriptors

Week 1 · Day 4: Classical Feature Extraction

Yusuf Brima, Ph.D.

Introduction to Computer Vision · AIMS Cameroon, 2026–2027

Today

1. **What makes a good feature**
2. **Image gradients**: where brightness changes, and in which direction
3. **The Canny edge detector**, stage by stage
4. **Corners**: the Harris detector and the structure tensor
5. **Scale**: blobs, scale space and SIFT
6. **HOG**: describing shape with histograms of gradients

By the end of today you should be able to

- compute gradients with Sobel by hand;
- explain every stage of Canny and tune it;
- derive the Harris response and read its eigenvalues;
- explain how SIFT achieves scale and rotation invariance;
- compute the size of a HOG descriptor.

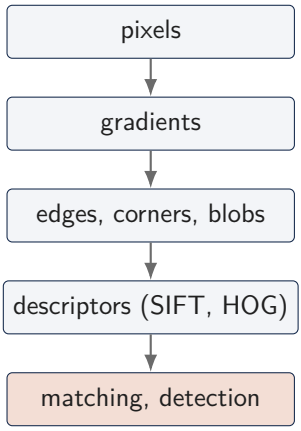
Where we are

So far

- Day 2: an image is a grid of noisy numbers.
- Day 3: convolution with a kernel; a kernel is a *template*; Sobel lights up at edges.

Today

- From *processing* images to *describing* them.
- A **feature** is a compact, informative summary of part of an image: a step from pixels towards meaning.



What makes a good feature?

1. **Repeatable**: found again in another photo of the same scene, despite viewpoint, light and scale changes.
2. **Distinctive**: different places give different descriptions, so they can be told apart.
3. **Local**: depends on a small region, so occlusion only spoils a few features.
4. **Efficient**: cheap to compute, compact to store.

A jigsaw puzzle

Which pieces do you place first? Not the blue-sky pieces (not distinctive), not straight-edge pieces in the middle of a wall (could go anywhere along it), but pieces with a *corner* of a window or a face. Good features are the pieces you can place with certainty.

Practitioner's Note

Repeatable and distinctive pull against each other: Day 1's invariance vs. selectivity tension, at patch level.

The naive idea: template matching

- To find an object, cut out a **template** and slide it over the image (correlation, Day 3).
- At each position, score how similar the patch is to the template.

Normalised cross-correlation

$$\mathrm{NCC} = \frac{\sum (T - \bar{T})(P - \bar{P})}{\sqrt{\sum (T - \bar{T})^2 \sum (P - \bar{P})^2}} \in [-1, 1]$$

T: template, P: image patch. Subtracting means and dividing by spreads ignores brightness and contrast.

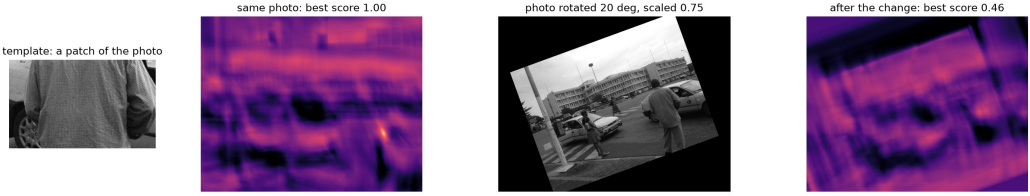
Intuition

NCC is the correlation coefficient from statistics, computed between two patches. 1 = identical up to brightness and contrast; 0 = unrelated.

Practitioner's Note

Template matching works well when nothing changes but position: reading a fixed form, aligning PCB images on a production line.

Why template matching is not enough



Bdx, CC BY-SA 4.0

In the original photo the template is found perfectly (NCC $= 1.00$, the bright peak). Rotate by 20° and shrink to 75%: the best score falls to 0.46 and the peak is lost among false matches.

Intuition

Raw pixels are not invariant. We need descriptions that survive rotation, scale and lighting, built from *gradients* rather than raw values. That is today's programme.

Part 1

Image gradients

What causes an edge in an image?



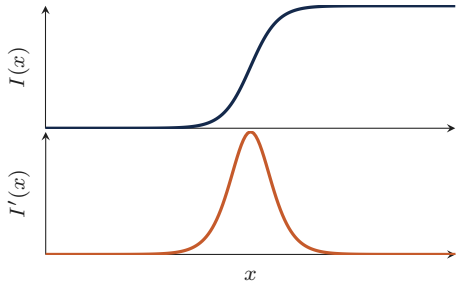
Bdx, CC BY-SA 4.0

- **Depth discontinuity**: the outline of the man against the road.
- **Surface orientation**: the fold between a taxi's roof and its side.
- **Reflectance change**: the red dress on the yellow taxi, zebra stripes on the road.
- **Illumination**: the edge of a shadow.

Intuition

To the camera, all four look the same: a sudden change in brightness. Only the first three tell you about *objects*; shadow edges are traps. Telling them apart is one reason why edges alone cannot solve vision.

The derivative measures change



- Along a row of pixels, an edge is a *ramp* from dark to bright.
- The derivative $I'(x)$ is near 0 on flat parts and **peaks** in the middle of the ramp.
- So: **edge = peak of the first derivative**.

Finite difference

$$I'(x) \approx \frac{I(x+1) - I(x-1)}{2}$$

As a kernel: $\frac{1}{2}[-1,\ 0,\ 1]$.

In 2D: the gradient

Gradient, magnitude and direction

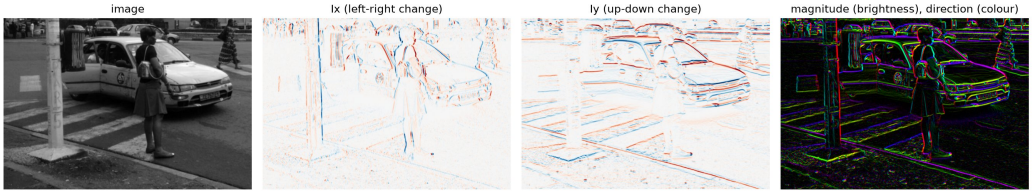
$$\nabla I = \begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} \partial I/\partial x \\ \partial I/\partial y \end{pmatrix}, \qquad \|\nabla I\| = \sqrt{I_x^2 + I_y^2}, \qquad \theta = \operatorname{atan2}(I_y,\, I_x)$$

- I_x: how fast brightness changes left to right; I_y: top to bottom.
- The gradient is an **arrow** at every pixel. It points *uphill*, towards brighter.
- Its length is the edge **strength**; the edge itself runs *perpendicular* to it.

One pixel

$I_x = 30$, $I_y = 40$.
$\|\nabla I\| = \sqrt{900 + 1600} = 50$
$\theta = \operatorname{atan2}(40, 30) \approx 53^\circ$
The edge runs at $53^\circ + 90^\circ = 143^\circ$.

Gradients of a real photograph

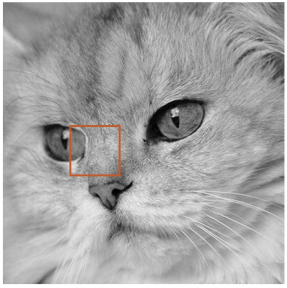


Bdx, CC BY-SA 4.0

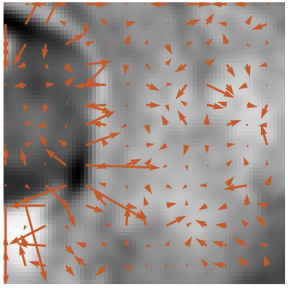
I_x responds to *vertical* edges (the pole, the woman's legs); I_y to *horizontal* ones (zebra stripes, the car's body lines). Red: positive, blue: negative.

Right: brightness shows the edge strength, colour shows the direction. Parallel edges with the same polarity share a colour.

Gradient arrows, up close



Gourdo 1 at en.wikipedia, Public domain



- Around the cat's eye, every arrow points from the dark pupil towards the bright fur.
- Long arrows sit on the strong outline; short arrows on smooth fur.
- The *pattern* of arrow directions describes local shape. SIFT and HOG are built exactly on this.

The Sobel operator

Sobel = smooth across $\times$ differentiate along

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

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

The $[1, 2, 1]$ part averages over three rows, which reduces noise; the $[-1, 0, 1]$ part takes the difference.

A vertical edge

Patch $\begin{bmatrix}20 & 20 & 200\\20 & 20 & 200\\20 & 20 & 200\end{bmatrix}$

G_x: $(200 - 20) + 2(200 - 20) + (200 - 20) = 720$

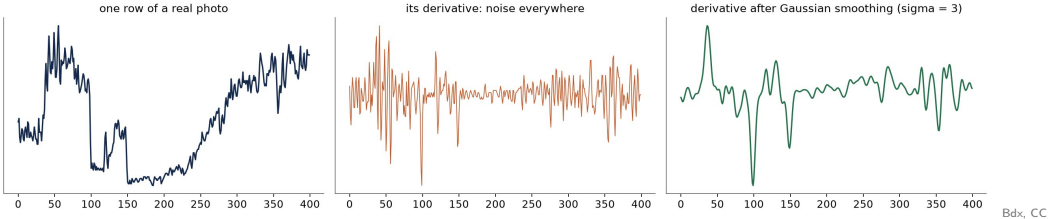
G_y: 0 (rows are identical)

$\|\nabla I\| = 720$, $\theta = 0°$: a vertical edge, bright on the right.

The same patch rotated $90°$

Bright row at the bottom: $G_x = 0$, $G_y = 720$, $\|\nabla I\| = 720$, $\theta = 90°$ (pointing down, towards brighter).

Derivatives amplify noise



The raw derivative of one row of a real photo is dominated by pixel-to-pixel noise. After Gaussian smoothing, the derivative shows clean peaks at the true edges.

Intuition

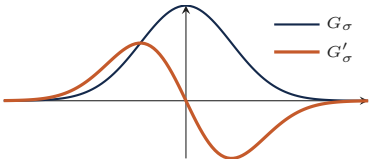
Noise is fast wiggling: high frequency (Day 3). A derivative *is* a high-pass filter, so it boosts noise most. Rule: **always smooth before differentiating**.

A trick worth remembering: differentiate the kernel

Derivative of Gaussian

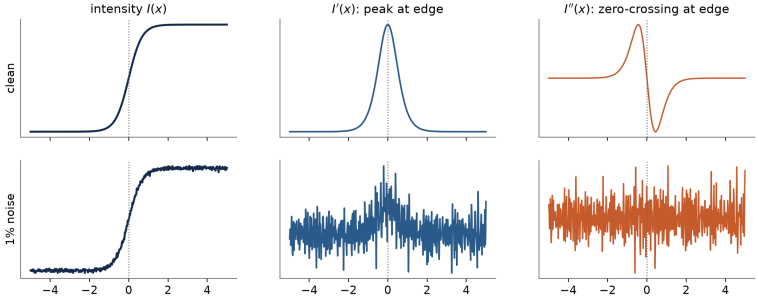
$$\frac{\partial}{\partial x}(G_\sigma * I) = \left(\frac{\partial G_\sigma}{\partial x}\right) * I$$

- Differentiation and convolution are both linear and shift-invariant, so they can be swapped.
- "Smooth, then differentiate" becomes **one** filter: a derivative-of-Gaussian kernel.
- σ is a **scale** knob: small σ finds fine edges, large σ coarse ones.



The derivative of a Gaussian is a smoothed $[-1, 0, 1]$; Sobel is a cheap approximation to it.

Second derivatives: the Laplacian



Laplacian

$$\nabla^2 I = I_{xx} + I_{yy} \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I$$

- Edge = **peak** of I' = **zero-crossing** of I''.
- Even 1% noise destroys I'' (bottom row).
- So smooth first: the **Laplacian of Gaussian**, LoG $= \nabla^2 G_\sigma * I$.

Part 2

The Canny edge detector

What should an edge detector do?

John Canny's three criteria (1986)

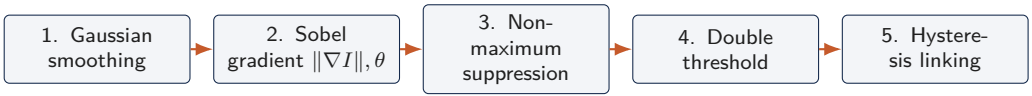
1. **Good detection**: find real edges, ignore noise.
2. **Good localisation**: put the edge where it really is.
3. **Single response**: one thin line per edge, not a thick band.

He showed that a derivative of Gaussian is close to the best filter for these criteria, then added two clever clean-up steps.

Intuition

Thresholding the gradient magnitude alone fails on all three: noise gives speckle, a blurred edge gives a band several pixels wide, and faint parts of a real edge break off. Canny fixes each problem with a dedicated stage.

Canny in five stages

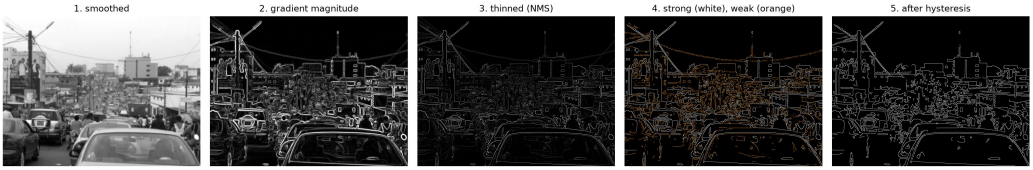


1. Suppress noise before differentiating.
2. Edge strength and direction at every pixel.
3. Keep a pixel only if it is the *maximum along its gradient direction*: thins ridges to 1 pixel.
4. Strong: $> T_{\text{high}}$. Weak: between T_{low} and T_{high}. The rest is dropped.
5. Keep weak pixels only if they connect to a strong one.

Why two thresholds?

Be *strict to start* an edge and *lenient to continue* it. A faint segment survives if it is attached to strong evidence; an isolated faint response (probably noise) does not.

Canny, stage by stage, on a Douala street



Guaka, CC BY-SA 4.0

Our own implementation. Stage 2 gives thick, fuzzy ridges; stage 3 thins them to single-pixel lines; stage 4 splits them into strong (white) and weak (orange); stage 5 keeps only weak pixels attached to strong chains.

Practitioner's Note

Thresholds here: T_{high} = 90th and T_{low} = 75th percentile of the gradient magnitude. Percentiles adapt to each image's contrast better than fixed numbers.

Non-maximum suppression, explained

40	90	60
~~70~~	150	~~95~~
30	80	50

gradient points right ($\theta = 0°$): compare 150 with 70 and 95

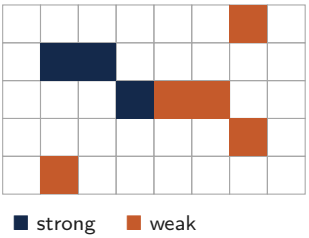
1. Round each pixel's gradient direction to one of four: $0°$, $45°$, $90°$, $135°$.
2. Look at the two neighbours *along* that direction (across the edge).
3. If the pixel is not larger than both, set it to 0.

Centre pixel

Magnitude 150, gradient pointing right: neighbours across the edge are 70 and 95. Since $150 > 95$ and $150 > 70$, keep it. The neighbour with 95 would be suppressed when it is tested against 150.

Intuition: a blurry edge is a ridge several pixels wide; NMS keeps only its crest.

Hysteresis, explained

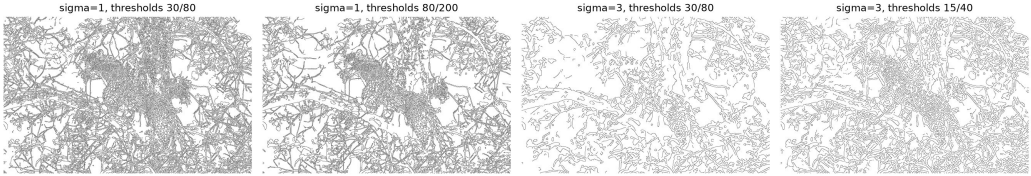


- The three **weak** pixels continuing the strong chain are *kept*: they extend a real edge that got fainter.
- The two **isolated weak** pixels (bottom left, top right) are *dropped*: nothing strong vouches for them.
- Implementation: connected components (Day 2) on strong $\cup$ weak, keep components containing at least one strong pixel.

Two extreme settings

$T_{\text{low}} = 0$: every thinned pixel attached to a strong chain survives: lots of texture and noise.
$T_{\text{low}} = T_{\text{high}}$: there are no weak pixels, so hysteresis does nothing; Canny becomes a single threshold, and faint parts of edges break off.

Tuning Canny on a real photograph



Davidvraju, CC BY-SA 4.0

- Larger σ: fewer, smoother, coarser edges.
- Higher thresholds: only the strongest edges survive.
- Heuristic: $T_{\text{high}} \approx 2\text{–}3 \times T_{\text{low}}$.

Intuition

Look how much of the leopard's camouflage is edges of leaves and spots. Edges are about *local change*, not objects. No setting outlines "the leopard".

Where edge detection is still used

- **Document scanning**: find the four edges of a page, then straighten it.
- **Lane detection**: edges + line fitting (Hough transform) in driver assistance.
- **Industrial measurement**: the precise outline of a machined part, to sub-pixel accuracy.
- **Medical**: outlines of bones, vessels and organs as a first step.

Practitioner's Note

`cv2.Canny(gray, 50, 150)` in one line. Smooth first, or pass a larger aperture. In controlled settings (factory, scanner) classical edges are fast, exact and need no training data.

Intuition

Edges are also what the first layer of every CNN learns (Day 5).

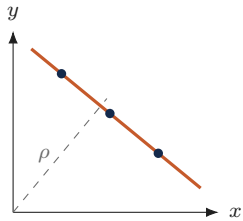
From edge pixels to lines: the Hough transform

- Canny gives *pixels*, not *lines*. Many applications need the line itself: a road edge, a document border, a lane.
- **Hough (1962)**: every edge pixel **votes** for all lines that could pass through it.
- Lines with many votes are real. Gaps and clutter barely matter.

A line in polar form

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

Each pixel (x, y) votes for a curve of (ρ, θ) pairs; votes pile up at the true line.



three pixels, one line:
all three vote for the same (ρ, θ)

The Hough transform on a real street

Canny edge pixels



16 straight segments found by Hough voting



Bdx, CC BY-SA 4.0

Canny edges, then probabilistic Hough voting (`cv2.HoughLinesP`): 16 long straight segments, from the kerb and the building's floors to the pole.

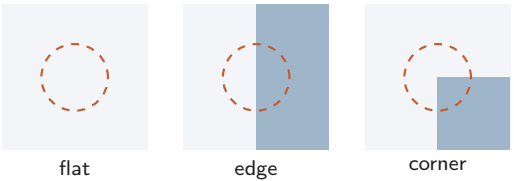
Intuition

Voting is a powerful classical idea: each piece of weak evidence votes, and agreement reveals structure. RANSAC (later today) is another voting scheme.

Part 3

Corners

Why corners? The aperture problem



Look through a small window and slide it:

- **flat**: nothing changes in any direction;
- **edge**: changes only *across* the edge; sliding along it is invisible;
- **corner**: changes in *every* direction, so its position is pinned down.

Intuition

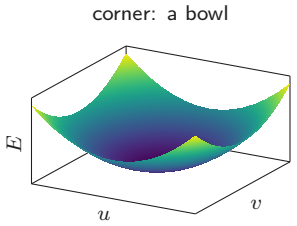
Matching, tracking, panoramas and 3D reconstruction need points that can be found *again* in another image. Corners are repeatable and precisely located; points on a straight edge are not. (The jigsaw piece again.)

Measuring "how much does the window change?"

Sum of squared differences

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

Shift the window by (u, v) and add up the squared changes. w is a Gaussian weight.



- Flat: $E \approx 0$ for every shift.
- Edge: E is large for shifts across, ≈ 0 along.
- Corner: E is large for every shift.

Intuition: the *shape* of E near zero tells what is in the window: a plain (flat), a valley (edge) or a bowl (corner).

Harris (1988): the derivation in three lines

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

$$E(u,v) \approx \sum_W w\,(I_x u + I_y v)^2 = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}, \qquad M = \sum_W w \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}.$$

The structure tensor M

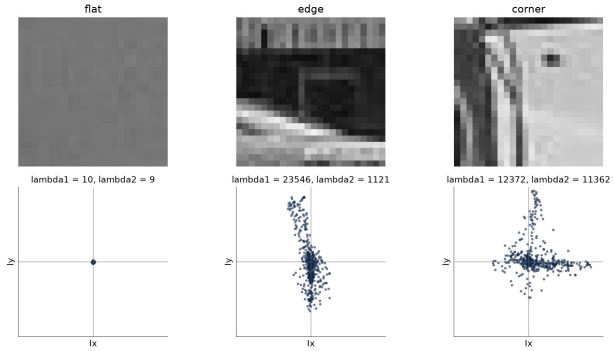
A 2×2 matrix built from gradients in the window. Its eigenvalues $\lambda_1 \geq \lambda_2$ measure how fast E grows along its steepest and its flattest direction.

- both small: **flat**;
- λ_1 large, λ_2 small: **edge**;
- both large: **corner**.

Intuition

M summarises the cloud of gradient vectors in the window. Its eigenvalues say how spread out the cloud is in two perpendicular directions.

The structure tensor on real patches



Bdx, CC BY-SA 4.0

Each dot is one pixel's gradient (I_x, I_y) in a 24×24 patch from the Yaoundé photo.

- **Flat**: all gradients near zero. $\lambda_1 = 10$, $\lambda_2 = 9$.
- **Edge**: gradients along one line. $\lambda_1 = 23\,546$, $\lambda_2 = 1121$.
- **Corner**: gradients in two directions. $\lambda_1 = 12\,372$, $\lambda_2 = 11\,362$.

The Harris response: avoiding eigenvalues

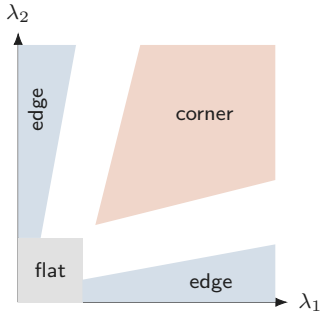
Harris response

$$R = \det M - k\,(\operatorname{tr} M)^2 = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2$$

$k \approx 0.04$–0.06. $\det$ and tr need no eigendecomposition.

Three cases, $k = 0.04$

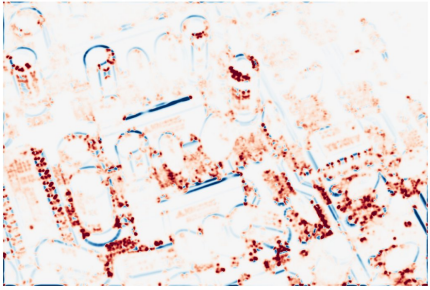
Flat $(1,1)$: $R = 1 - 0.16 = 0.84$ (small)
Edge $(100,1)$: $R = 100 - 408 = -308$ (negative)
Corner $(100,100)$: $R = 10\,000 - 1600 = 8400$ (large)



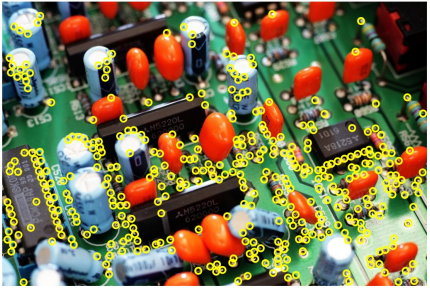
$R > 0$ large: corner; $R < 0$: edge

Harris on a circuit board

Harris response R (red: corner, blue: edge)



369 corners (local maxima of R)



Ashley Pomeroy, CC BY 4.0

Pipeline: Sobel $I_x, I_y \rightarrow$ products $I_x^2, I_y^2, I_x I_y$ $\rightarrow$ Gaussian window $\rightarrow R \rightarrow$ keep local maxima above a threshold. 369 corners: chip pins, solder pads, component ends.

Intuition

Red (corners) clusters on the pins and pads; blue (edges) traces the long straight sides of capacitors. Exactly the flat/edge/corner picture, on a real board.

Is Harris invariant?

Rotation: yes

Rotating the image rotates every gradient: $M \mapsto QMQ^\top$ with Q a rotation. Eigenvalues are unchanged, so R is unchanged.

Adding brightness $I + c$: yes

Gradients ignore constant offsets; M does not change.

Contrast scaling aI: partly

Gradients scale by a, so $M \mapsto a^2 M$ and $R \mapsto a^4 R$. The *locations* of maxima stay, but a fixed threshold now keeps more or fewer corners.

Image scale: no

Zoom in on a rounded corner and, inside a fixed window, it looks like an edge. The window size must match the structure: the scale problem.

A popular variant: Shi–Tomasi

Shi–Tomasi (1994)

$$R = \min(\lambda_1, \lambda_2) = \lambda_2$$

A corner is a window whose *weakest* direction still changes a lot.

- Simpler to interpret than Harris: no constant k.
- Slightly more stable for tracking from frame to frame.

Practitioner's Note

`cv2.goodFeaturesToTrack(gray, maxCorners=500, qualityLevel=0.01, minDistance=8)` plus the Lucas–Kanade tracker is still how many drones and phones follow points in video, because it is fast and needs no training.

Intuition

Lucas–Kanade optical flow also solves a system with the *same* matrix M: points where M is well conditioned (corners) are exactly the points whose motion can be measured.

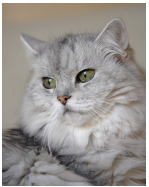
Part 4

Scale: blobs, scale space and SIFT

The problem of scale



Hisashi from Japan, CC BY-SA 2.0



Gourdo 1 at en.wikipedia, Public domain

- A cat's eye covers 3 pixels in one photo and 80 in another.
- A detector with a fixed window size sees completely different structures.
- We need features that **choose their own size**: report not just *where* but also *how big*.

Intuition

Idea: look at the image at *every* scale at once by blurring it more and more (Day 3: σ sets the scale of detail kept). A structure "belongs" to the scale where it responds most strongly.

Image pyramids

Gaussian pyramid: blur, then halve, repeatedly (1024x683 -> 512x342 -> 256x171 -> 128x86 -> 64x43)



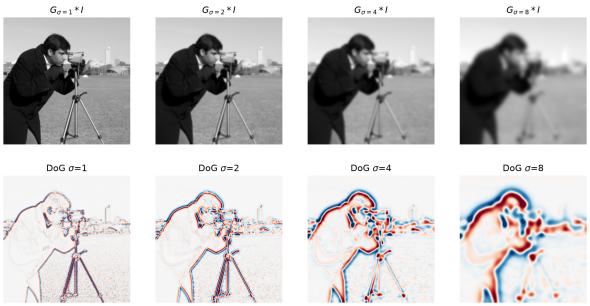
Minette Lontsie, CC BY-SA 4.0

Blur, then halve the size; repeat. Each level shows the scene at a coarser scale with a quarter of the pixels, so the whole pyramid costs only a third more than the original.

Intuition

Blurring before halving is Day 2's anti-aliasing rule. A fixed-size detector run on every level finds both near (large) and far (small) objects. CNNs build the same pyramid internally (Week 2).

Scale space



Scale space and difference of Gaussians

$$L(x, y, \sigma) = G_\sigma * I$$

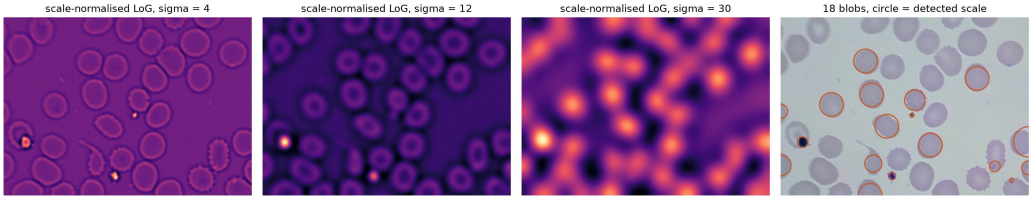
$$D = L(\cdot, k\sigma) - L(\cdot, \sigma) \approx (k-1)\,\sigma^2 \nabla^2 G_\sigma * I$$

The DoG is a cheap approximation of the scale-normalised Laplacian. And it is the retina's centre–surround filter from Day 1.

Intuition

A blob "lights up" in the DoG at the σ matching its size (radius $\approx \sqrt{2}\,\sigma$). Find maxima over x, y *and* σ, and you get a position and a characteristic scale.

Blob detection on a real blood smear



OpenFlexure Project, CC BY-SA 4.0

At $\sigma = 4$ the small dark parasites respond; at $\sigma = 12$ the red cells light up; at $\sigma = 30$ only clusters of cells. Scale-space maxima give 18 blobs, each with a circle of its own size.

Practitioner's Note

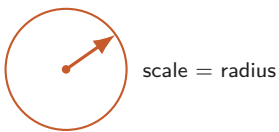
`skimage.feature.blob_log` and `blob_dog`. Used for cell and nucleus detection, star detection in astronomy, and counting fruit or seedlings in drone images.

SIFT (Lowe, 2004): the detector

1. Build a DoG pyramid: blur, subtract, downsample by 2, repeat ("octaves").
2. Keep points that are larger or smaller than all **26 neighbours**: 8 in the same scale, 9 above, 9 below.
3. Refine to sub-pixel position; drop low-contrast points and edge-like points (a Harris-style test).
4. Assign a **dominant orientation** from a histogram of nearby gradient directions.

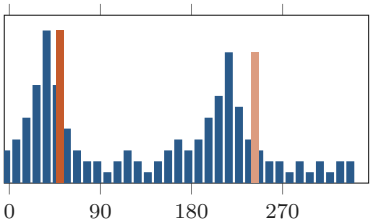
Intuition

After step 4 every keypoint carries its own *position*, *scale* and *orientation*: a little coordinate frame. Describe the patch *in that frame* and the description no longer depends on where, how big or how rotated the object is.



Worked example: assigning an orientation

1. Around the keypoint, compute every pixel's gradient direction and magnitude.
2. Build a histogram with 36 bins of 10°, each vote weighted by magnitude (and a Gaussian of distance).
3. The highest bin is the keypoint's orientation.
4. Any other bin above 80% of the peak creates a *second* keypoint with that orientation.



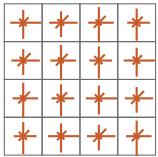
Peak at 50° (orange); the bin at 230° is $12/14 = 86\% > 80\%$ of it, so a second keypoint is created.

Intuition

Every descriptor is then measured *relative* to this orientation. Rotate the image by 30° and the peak moves to 80°, but the description relative to it is unchanged.

SIFT: the 128-number descriptor

1. Take a patch around the keypoint, rotated and scaled to its own frame.
2. Split it into a 4×4 grid of cells.
3. In each cell, a histogram of gradient directions with **8 bins**, weighted by magnitude.
4. $16 \times 8 = \mathbf{128\ numbers}$.
5. Normalise to unit length; clip large values; normalise again (robust to lighting).

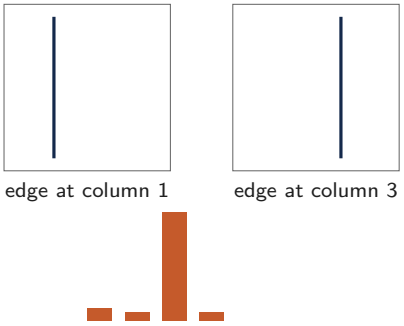


16 cells $\times$ 8 directions $= 128$

Intuition

Histograms forget *exactly* where each gradient was inside its cell: small shifts and deformations barely change the descriptor. This is pooling, the same idea as complex cells in V1 (Day 1).

Why histograms? Tolerance to small shifts



- Raw pixels: shifting the edge by two columns changes many pixel values, a big change.
- Direction histogram: both cells contain "one vertical edge". **Identical description.**
- But the histogram still knows the edge is *in this cell* rather than the next: coarse position survives.

Intuition

Pooling into cells trades a little precision for a lot of robustness. CNNs do the same with pooling layers and strides (Week 2).

Matching descriptors: Lowe's ratio test

- For each descriptor in image A, find the **nearest** (b_1) and **second nearest** (b_2) descriptors in image B (Euclidean distance).
- A good match is much closer than any alternative. An ambiguous one (a window among many identical windows) is not.

Ratio test

accept if $$\frac{\|d_a - d_{b_1}\|}{\|d_a - d_{b_2}\|} < 0.75$$

Two candidate matches

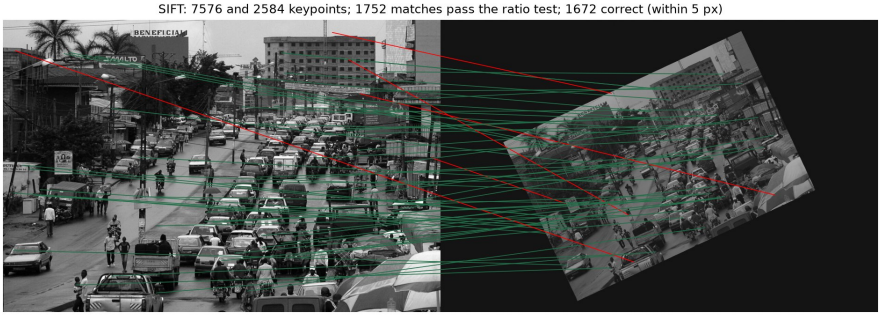
Distances 0.20 and 0.60: ratio 0.33. **Accept**: clearly the best.

Distances 0.30 and 0.32: ratio 0.94. **Reject**: could be either.

Intuition

Repeated patterns (tiles, windows, fences) are the classic trap. The ratio test refuses to guess between near-identical candidates.

SIFT matching on a real photograph



Happiraphael, CC BY-SA 4.0

Right: the same Douala street rotated by $25°$, shrunk to 60% and darkened. Of 1752 matches that pass the ratio test, **1672 (95%)** land within 5 pixels of the true position (green); red lines are errors.

Intuition

Rotation, scale and lighting changed at once, and SIFT still matches. That is what "scale- and rotation-invariant" buys you.

From matches to a panorama



Happiraphael, CC BY-SA 4.0

Two overlapping views (the second shifted and tilted). ORB features are matched; **RANSAC** repeatedly fits a geometric transform (a homography) to random subsets and keeps the one most matches agree with: 632 of 713.

Intuition

RANSAC turns a noisy list of matches into geometry by *voting*. Wrong matches disagree with each other; correct ones agree. Your phone's panorama mode works this way.

ORB and modern uses of local features

ORB (2011): SIFT for phones

- FAST corners: compare a ring of 16 pixels with the centre.
- BRIEF descriptor: 256 yes/no answers to "is pixel p brighter than pixel q?"
- Matched by Hamming distance (count differing bits): very fast.
- About 10$\times$ faster than SIFT; free; good enough for real time.

Where local features live today

- Panoramas and photo stitching
- Visual SLAM: robots and AR headsets locating themselves
- Structure from motion: 3D models from photo collections and drone surveys
- Matching drone images to maps

Practitioner's Note

SIFT has been patent-free since 2020: `cv2.SIFT_create()`, `cv2.ORB_create()`. Learned local features (SuperPoint, LoFTR) now often beat them, but follow the same detect–describe–match recipe.

Part 5

HOG: describing shape

From sparse keypoints to a dense description

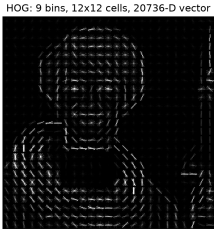
- SIFT describes a few hundred *interesting points*: good for matching the *same* object.
- Recognising a *category* ("a person", any person) needs a description of the *whole window*.
- **HOG** (Dalal & Triggs, 2005): SIFT-like histograms everywhere, on a regular grid.

Intuition

A person's silhouette has typical gradient patterns: vertical edges for the legs and body, a rounded head, horizontal shoulders. HOG records *which edge directions appear where*, and ignores clothing colour.

HOG step by step





1. Compute gradients at every pixel.
2. Divide the window into **cells** (e.g. 8×8 pixels). In each cell, a 9-bin histogram of gradient direction (0–180°, ignoring sign), weighted by magnitude.
3. Group 2×2 cells into overlapping **blocks**; normalise each block, for invariance to contrast and lighting.
4. Concatenate all blocks into one long vector.

The classic pedestrian window

64×128 pixels with 8×8 cells: 8×16 cells, $7 \times 15 = 105$ blocks, each $4 \times 9 = 36$ numbers: $105 \times 36 =$ **3780** numbers.

HOG on a real pedestrian



Bdx, CC BY-SA 4.0

A man crossing the road in Yaoundé, resized to 128×256 with 16×16 cells: again 8×16 cells, 105 blocks, **3780** numbers.

The HOG picture draws, in each cell, line segments along the dominant edge directions. You can still read the outline of shoulders, back and legs.

Practitioner's Note

HOG + a linear SVM (Day 5) was the best pedestrian detector until deep learning, and is still in `cv2.HOGDescriptor` with a pre-trained people detector.

Comparing the classical toolbox

Method	Finds / describes	Invariant to	Typical use
Sobel, Canny	edges	brightness offset	documents, lanes, measurement
Harris	corners	rotation, brightness	tracking, calibration
SIFT	blobs + 128 numbers	scale, rotation, lighting	matching, stitching, 3D
ORB	corners + 256 bits	rotation (partly scale)	real-time SLAM on devices
HOG	dense gradient histograms	small shifts, lighting	sliding-window detection

Intuition

Every row is built from the same few ingredients: **Gaussian smoothing**, **gradients**, **local histograms** (pooling) and **normalisation**. Tomorrow we see that a CNN learns these ingredients instead of us designing them.

Choosing a feature for a problem

Stitching drone photos of a farm

Same scene, different positions and heights: need scale and rotation invariance and precise points. **SIFT or ORB** + RANSAC.

Detecting people at a junction

A *category* with varied clothes, upright pose: **HOG** + a classifier, or a CNN detector.

Counting cells in a smear

Round objects of a known size range: **LoG / DoG blobs**, or threshold + components (Day 3).

Measuring a machined part

Controlled light, known shape, sub-pixel precision: **Canny** + line/circle fitting.

Practitioner's Note

Ask two questions: *same object or same category?* and *which changes must I be invariant to?*

Part 6

Wrapping up

Key takeaways

1. Good features are repeatable, distinctive, local and efficient.
2. The **gradient** is an arrow towards brighter; edges are its peaks. Always smooth before differentiating.
3. **Canny**: smooth, gradient, thin (NMS), two thresholds, link (hysteresis).
4. **Harris**: eigenvalues of the structure tensor tell flat, edge and corner apart.
5. **Scale space**: blur at many σ; structures pick their own scale.
6. **SIFT**: DoG keypoints with scale and orientation + 128-number gradient histograms; match with the ratio test.
7. **HOG**: dense cell histograms + block normalisation for category detection.

Before the next lecture

Practise (tutorial notebook `W1D4_Edges_Corners_Descriptors`)

- Verify the Sobel worked example (720) in code.
- Write NMS and hysteresis; compare with `cv2.Canny`.
- Implement Harris; test rotation invariance.
- Match SIFT and ORB features between two of your own photos; stitch them.

Read

- Course notes, Chapter 4, §4.1
- Szeliski (2022), §7.1–7.2

Next lecture: from engineered to learned features

- The classical pipeline: features + classifier
- Viola–Jones, HOG + SVM, bag of visual words
- An experiment: raw pixels vs. HOG
- Learning a filter by gradient descent
- From a neuron to a convolutional layer

Credits

Photographs from Wikimedia Commons under the licences shown. Other images: scikit-image sample data (`camera`, `astronaut`).

`yaounde_taxis`: Bdx, CC BY-SA 4.0
`cat_persian`: Gourdo 1 at en.wikipedia, Public domain
`cat_small`: Hisashi from Japan, CC BY-SA 2.0
`douala_2014`: Guaka, CC BY-SA 4.0
`leopard_tree`: Davidvraju, CC BY-SA 4.0

`pcb`: Ashley Pomeroy, CC BY 4.0
`malaria_smear`: OpenFlexure Project, CC BY-SA 4.0
`douala_deido`: Happiraphael, CC BY-SA 4.0
`douala_ndokoti`: Minette Lontsie, CC BY-SA 4.0