

COMPUTER VISION

Filtering and edges

Session 03

Spatial and frequency filtering,
gradients and edge detection

Mohammad Sabokrou
New Uzbekistan University



What we already know

Session 1

An image is an array.

Raw pixel comparisons can fail when recording conditions change.

Session 2

A histogram counts values.

It loses the positions that describe shape and boundaries.

Today: relationships between neighbouring pixels

Which nearby values support a boundary at this location?

The missing information is spatial

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

8 dark
8 light
in each

A: one straight boundary

0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

B: repeated transitions

The counts match. The neighbours differ.

Point operations and neighbourhood operations

Point operation

$$J[r,c] = 1.4 I[r,c] - 20$$

One input value controls one output value.

Neighbourhood operation

10	10	10
10	90	10
10	10	10

The centre output uses several nearby input values.

Here **r** means row and **c** means column.

Today's questions and route

Minutes	Question
0-7	Why do neighbours matter?
7-31	How does convolution work in 1D and on an image?
31-61	Which spatial and frequency filters solve which problems?
61-70	Why use horizontal, vertical or separable filtering?
70-75	Break
75-112	How do directional derivatives become edge detections?
112-120	Discussion and self-check

Predict the effect, calculate a small example, then inspect an image.

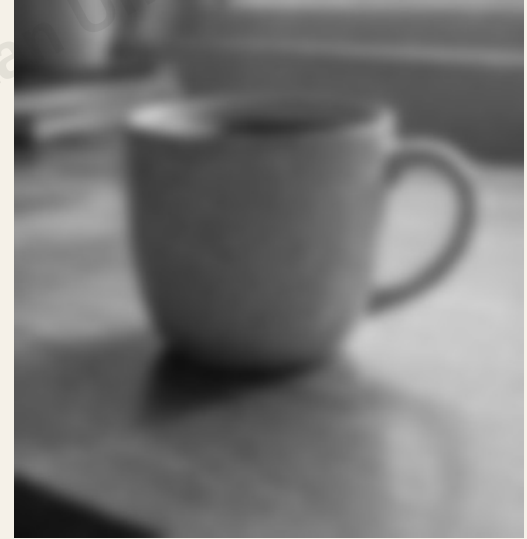
Can we clean the image and keep the rim?



Noisy input



Moderate smoothing



Strong smoothing

What improved? What useful information disappeared?

Convolution in one dimension

$$y[n] = (x * h)[n] = \sum h[k] x[n - k]$$

For $k = -1, 0, 1$:

$$y[n] = h[-1]x[n+1] + h[0]x[n] + h[1]x[n-1]$$

x: input signal h: kernel weights y: output signal

n: output position k: kernel offset

One output is a weighted sum. The same rule moves along the signal.

1D convolution: first complete window

$x[n]$

2	4	6	8	10	12
---	---	---	---	----	----

$h = [1, 2, 0]$ · flipped overlay = $[0, 2, 1]$

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

$$y[1] = 0 \times 2 + 2 \times 4 + 1 \times 6 = 14$$

The centre of the first valid window is $n = 1$.

1D convolution: move one position

$x[n]$

2	4	6	8	10	12
---	---	---	---	----	----

The kernel moves. Its weights stay fixed.

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

$$y[2] = 0 \times 4 + 2 \times 6 + 1 \times 8 = 20$$

Always read from the original input, not from an output already computed.

1D convolution: finish the valid output

Centre n	Window	Convolution result
1	2, 4, 6	$0 \times 2 + 2 \times 4 + 1 \times 6 = 14$
2	4, 6, 8	$0 \times 4 + 2 \times 6 + 1 \times 8 = 20$
3	6, 8, 10	$0 \times 6 + 2 \times 8 + 1 \times 10 = 26$
4	8, 10, 12	$0 \times 8 + 2 \times 10 + 1 \times 12 = 32$

Valid output

14	20	26	32
----	----	----	----

With a symmetric kernel, convolution and correlation give the same values.

A neighbourhood surrounds one pixel

12	14	16	18	20
14	18	20	22	24
16	20	90	24	26
18	22	24	26	28
20	24	26	28	30

Target: the centre value 90

The highlighted 3×3 region contains the pixels we will combine.

A 5×5 window includes a wider neighbourhood.

The window chooses which inputs can affect one output.

Convolution on an image uses two sums

Convolution: flip the kernel in both axes

$$J[r,c] = \sum \sum K[a,b] I[r-a,c-b]$$

Correlation: apply the weights as written

$$J[r,c] = \sum \sum K[a,b] I[r+a,c+b]$$

For a 3×3 kernel, a and b each take the values -1, 0, 1.

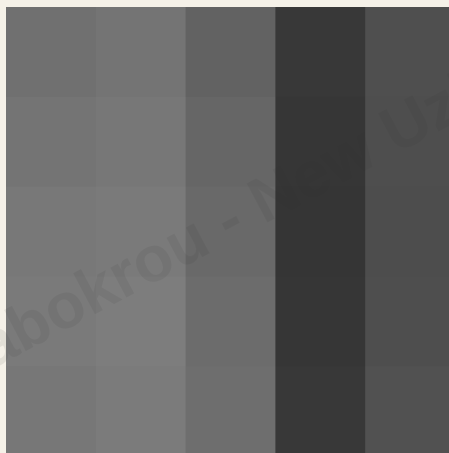
There are nine products at each image location.

A real image patch becomes a small matrix

Crop: rows 110–114, columns 75–79 (zero-based)



Image input



5 × 5 pixel crop

112	116	98	56	79
116	119	102	54	78
120	122	105	53	77
122	125	108	54	78
119	123	110	56	81

Exact gray values

A 3×3 mean kernel replaces each centre by the average of its neighbourhood.

Image convolution: the first output

Input: current 3 × 3 window

112	116	98	56	79
116	119	102	54	78
120	122	105	53	77
122	125	108	54	78
119	123	110	56	81

Kernel

1	1	1
1	1	1
1	1	1

× 1/9

Output built so far

112.22	.	.
.	.	.
.	.	.

$$112 + 116 + 98$$

$$116 + 119 + 102$$

$$120 + 122 + 105$$

Sum = 1010 Mean = $1010 / 9 = 112.22$ Store at J[0,0].

Image convolution: one column to the right

Input: current 3 × 3 window

112	116	98	56	79
116	119	102	54	78
120	122	105	53	77
122	125	108	54	78
119	123	110	56	81

Kernel

1	1	1
1	1	1
1	1	1

× 1/9

Output built so far

112.22	91.67	.
.	.	.
.	.	.

$$\begin{aligned}
 &116 + 98 + 56 \\
 &119 + 102 + 54 \\
 &122 + 105 + 53
 \end{aligned}$$

Sum = 825 Mean = $825 / 9 = 91.67$ Store at J[0,1].

Image convolution: start the next output row

Input: current 3 × 3 window

112	116	98	56	79
116	119	102	54	78
120	122	105	53	77
122	125	108	54	78
119	123	110	56	81

Kernel

1	1	1
1	1	1
1	1	1

× 1/9

Output built so far

112.22	91.67	78.00
115.44	.	.
.	.	.

$$116 + 119 + 102$$

$$120 + 122 + 105$$

$$122 + 125 + 108$$

Sum = 1039 Mean = $1039 / 9 = 115.44$ Store at J[1,0].

Image convolution: the completed result

All 3×3 valid outputs

112.22	91.67	78.00
115.44	93.56	78.78
117.11	95.11	80.22



Full image: 3×3 mean

A small worked patch and a full image use exactly the same local calculation.

The image border needs a rule

At the first pixel, the left neighbour lies outside the image.

Rule	Extended neighbourhood	Mean
Zero	0, 10, 20	10.00
Replicate edge	10, 10, 20	13.33
Reflect, no edge repeat	20, 10, 20	16.67

Padding changes border outputs. Record the exact rule.

Kernel size changes the valid output size

$$H_{out} = H + 2p - k + 1$$

$$W_{out} = W + 2p - k + 1$$

H, W: input size k: kernel size p: padding on each side

Input	Kernel	Padding	Output
5×5	3×3	$p = 0$	3×3
5×5	3×3	$p = 1$	5×5

Here the stride is 1 and the kernel has no gaps.

Different weights produce different effects



Identity: keep values



Average: smooth detail



Sharpen: boost differences

The weights control smoothing, preservation or amplification of local changes.

The weight sum predicts a constant-region response

If every input is b , then $J = b \sum K$

Kernel weight sum	Constant input $b = 50$	Interpretation
1	$J = 50$	Preserves the constant
0	$J = 0$	Responds to change
2	$J = 100$	Doubles the constant

A sum of 1 is necessary for constant preservation, not proof of smoothing.

Gaussian smoothing favours nearby pixels

1	2	1
2	4	2
1	2	1

× 1/16

The centre receives more weight.
Farther neighbours receive less.

$$K[a,b] \propto \exp(-(a^2+b^2)/(2\sigma^2))$$

σ sets the spatial spread in pixels.
Normalize the discrete weights
so their sum equals 1.

The 3×3 kernel shown is a binomial approximation.

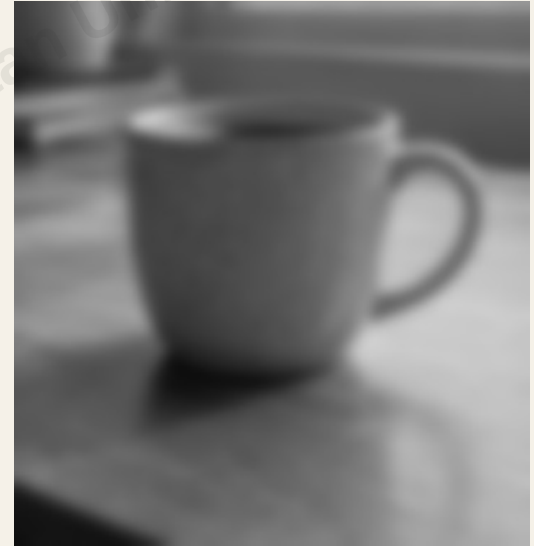
Sigma controls the scale of smoothing



$\sigma = 0.6$ px



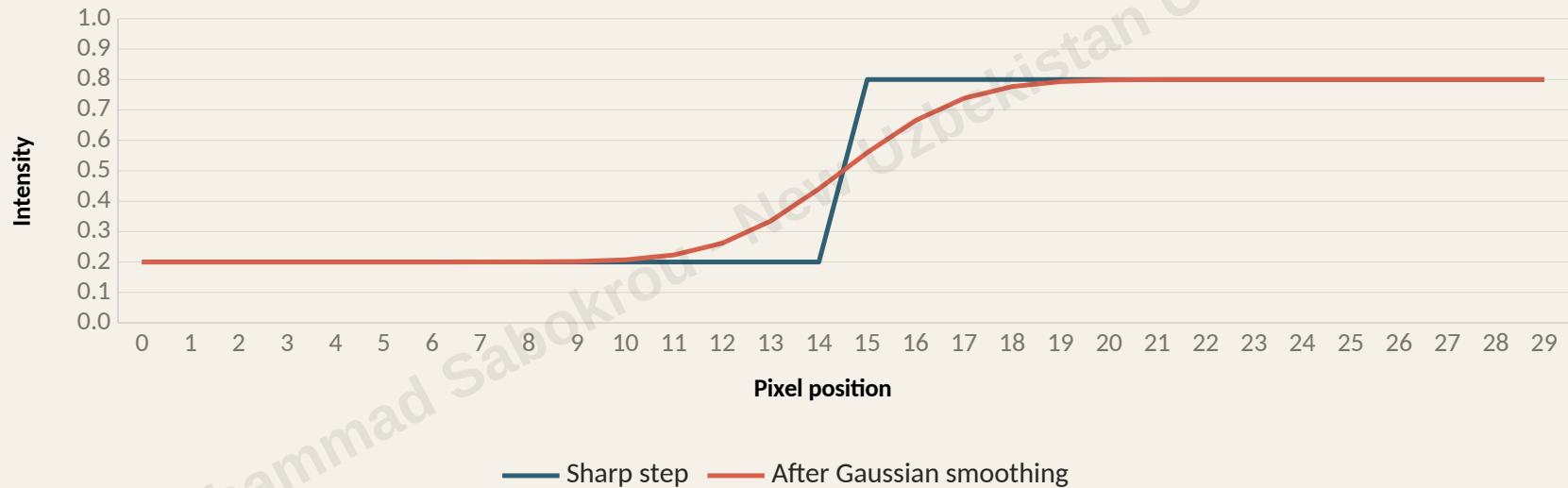
$\sigma = 2.5$ px



$\sigma = 5.0$ px

Larger σ mixes values over a wider spatial neighbourhood.

Smoothing reduces noise and broadens transitions



A smoother signal can make the exact edge location less precise.

The median handles isolated extreme values differently



Impulse noise



3×3 mean



3×3 median

Median filtering is a neighbourhood operation, but not convolution.

Low-pass and high-pass kernels in space

Low-pass: local mean

1	1	1
1	1	1
1	1	1

Divide by 9. Weight sum = 1.

High-pass: input - local mean

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

Divide by 9. Weight sum = 0.

Low-pass smooths rapid variation. High-pass extracts a signed detail residual.

Sharpening adds detail back to the input

$$L = \text{blur}(I) \quad H = I - L \quad J = I + \alpha H$$



Low-pass L



Detail H: gray = 0



Sharpened J ($\alpha = 1.5$)

The high-pass residual isolates detail. Adding it back increases local contrast.

Bilateral filtering also compares pixel values

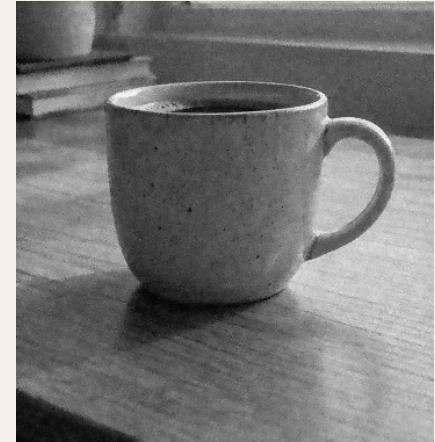
Weight = spatial closeness $\times$ intensity similarity



Noisy input



Gaussian



Bilateral

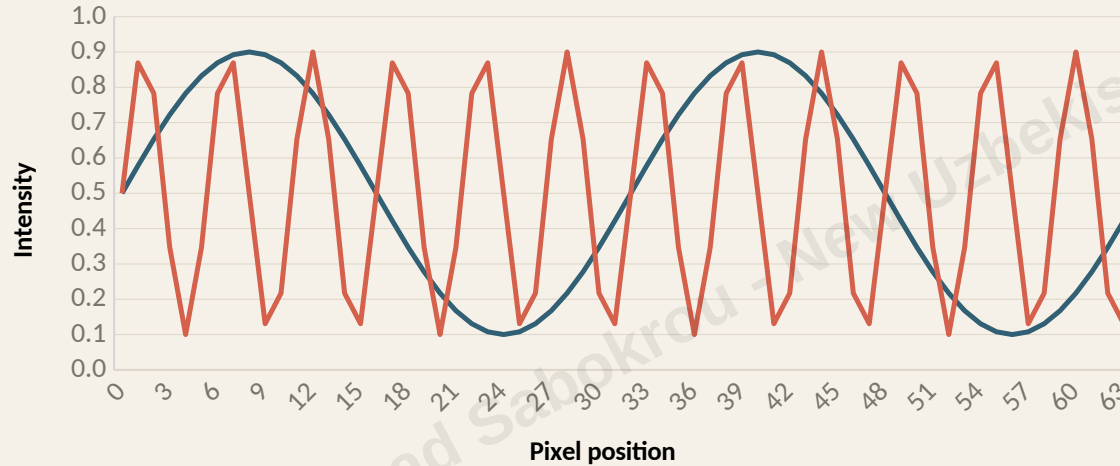
Nearby pixels across a large intensity jump can receive smaller bilateral weights.

Classical filter choice depends on the problem

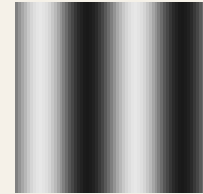
Goal	Classical choice	Main trade-off
Reduce small fluctuations	Mean or Gaussian	Fine detail also weakens
Remove sparse extreme pixels	Median	Small structures can disappear
Limit mixing across edges	Bilateral	More parameters and work
Measure local changes	Prewitt, Sobel, Scharr	Direction and noise sensitivity
Boost detail	Unsharp mask / Laplacian	Noise, halos and clipping

The filter name is a starting point. The image and the task determine its usefulness.

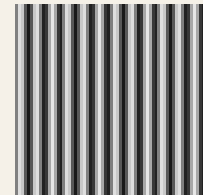
Spatial frequency means how fast values change



— 2 cycles in 64 pixels — 12 cycles in 64 pixels



Slow variation



Rapid variation

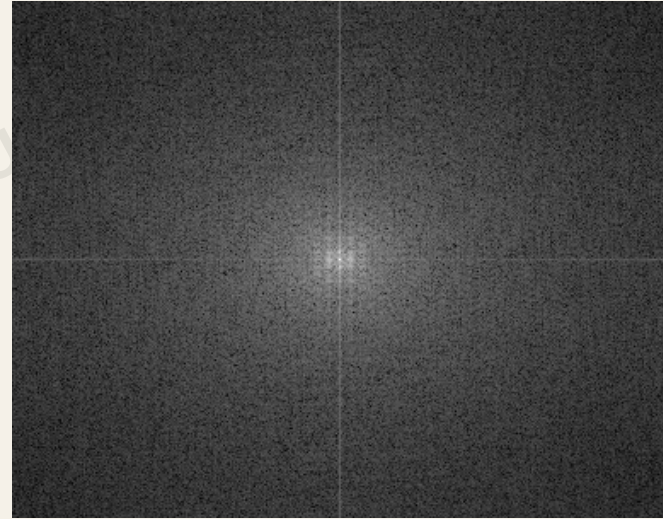
Broad smooth regions vary slowly. Fine texture, sharp edges and noise vary rapidly.

The Fourier spectrum describes image frequencies

DFT = discrete Fourier transform



Image: where values occur



Spectrum: rates of variation

Centre = zero / low frequency. Farther out = faster spatial variation.

Spatial convolution becomes frequency multiplication

$$J = I * K$$

$$F = \text{DFT}(I) \quad H = \text{DFT}(K)$$

$$J = \text{IDFT}(F \odot H)$$

$\odot$ multiplies matching frequency coefficients.

The inverse transform returns the result to image space.

DFT filtering needs a boundary rule too. Padding prevents unwanted wraparound.

Low-pass masks retain slow variation

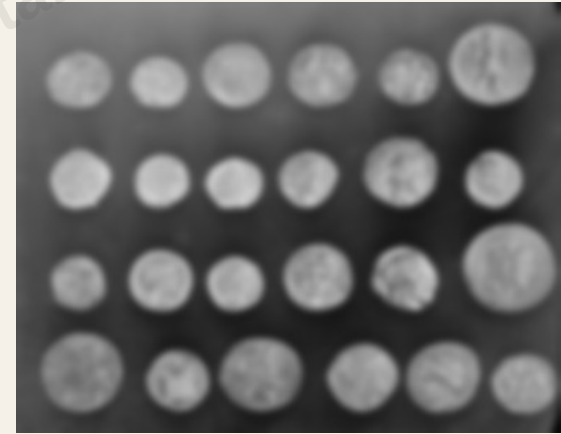
$$H^L(D) = \exp(-D^2 / (2D_0^2))$$



Input



Mask: white keeps, black blocks



Reconstructed low-pass

A wider central mask retains more high-frequency detail and produces less blur.

High-pass masks suppress slow variation

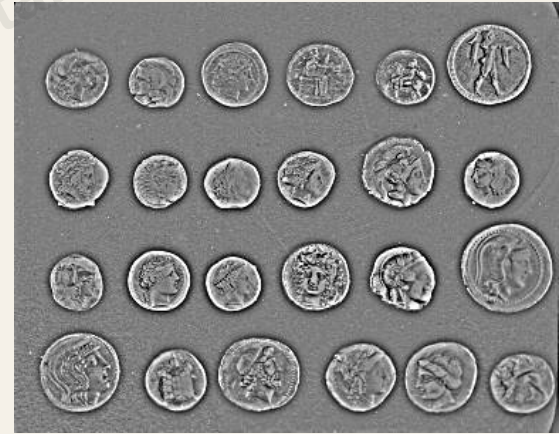
$$H^H = 1 - H^L \quad I = \text{low-pass}(I) + \text{high-pass}(I)$$



Input



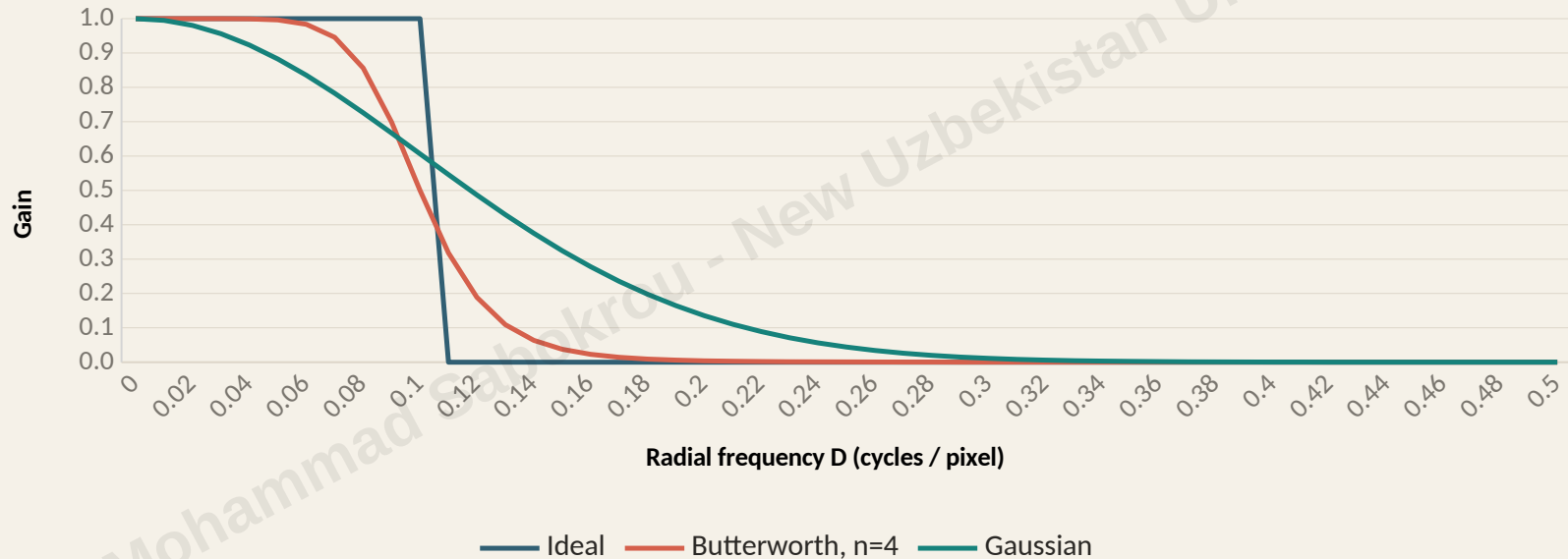
Mask: white keeps, black blocks



Signed detail: gray = 0

High-pass extracts detail. Edge detection still needs selection and thinning.

Ideal, Butterworth and Gaussian responses



A sharp cutoff is selective, but can introduce ripples near strong image transitions.

The mask formula controls the transition

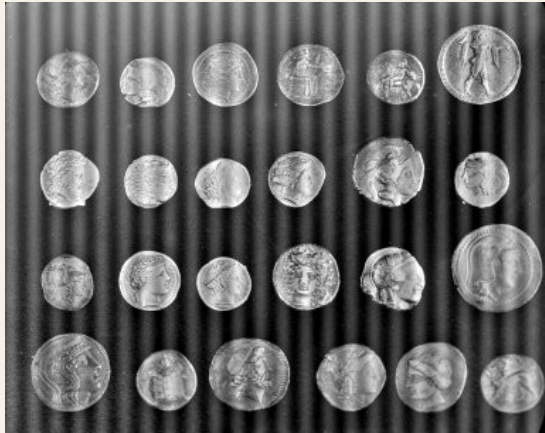
Low-pass family	Gain $H^L(D)$	Behaviour
Ideal	1 if $D \leq D_0$, otherwise 0	Abrupt cutoff
Butterworth	$1 / (1 + (D/D_0)^{2n})$	Adjustable transition
Gaussian	$\exp(-D^2 / (2D_0^2))$	Smooth transition

D : radial frequency D_0 : frequency scale n : filter order

A complementary high-pass mask is $H^H = 1 - H^L$.

Band-pass and notch filters select specific frequencies

Band-pass: retain a frequency band. Notch: reject a narrow frequency pair.



Periodic stripes added



Notch pair (frequency zoom)



After notch rejection

Notch filtering removes periodic interference and some detail at the same frequencies.

Which filter would you try first?

A A flat wall has small random brightness fluctuations.

B A few pixels suddenly become black or white.

C A one-pixel crack must remain visible.

Explain both the expected benefit and the possible loss.

The noise model and the task guide the choice

Case	Reasonable first test	What to inspect
A: small fluctuations	Modest Gaussian smoothing	Noise reduction and edge spread
B: sparse extremes	Small median window	Removed outliers and lost detail
C: thin crack	Very weak or no smoothing	Whether the crack survives

Judge the output using the structure needed for the task.

A separable kernel is an outer product

1
2
1

Divide by 4

×

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

Divide by 4

=

1	2	1
2	4	2
1	2	1

Divide by 16

One vertical vector and one horizontal vector define the full kernel.

Two one-dimensional passes give the same answer

10	10	10
10	90	10
10	10	10

Horizontal
[1,2,1] / 4

10
50
10

Vertical
[1,2,1] / 4

30

Direct 2D: $(4 \times 90 + 12 \times 10) / 16 = 30$

Separability reduces the work per pixel

Kernel width k	Direct 2D: k^2	Two 1D passes: $2k$
3	9	6
7	49	14
15	225	30

$$225 / 30 = 7.5$$

times fewer weighted terms
for a 15×15 separable kernel

The count predicts arithmetic savings, not an exact runtime speed-up.

Can these kernels separate?

A

1	2	1
2	4	2
1	2	1

B

0	1	0
1	4	1
0	1	0

Are all nonzero rows scaled copies of one row?

Answer: A separates. B cannot be one outer product.

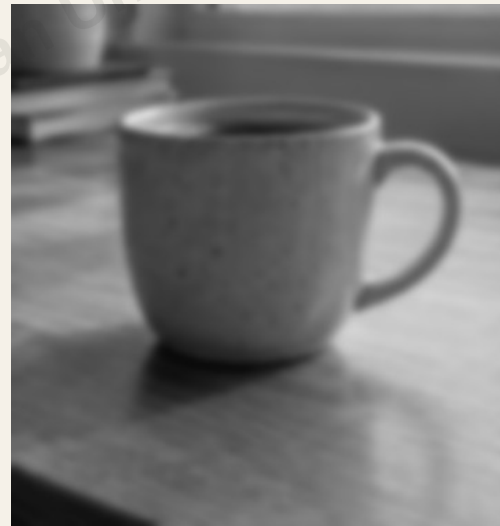
Horizontal, vertical and joint smoothing



Across columns only



Down rows only



Both axes, in sequence

Two 1D smoothing passes give a 2D result with fewer weighted terms.

Five-minute break

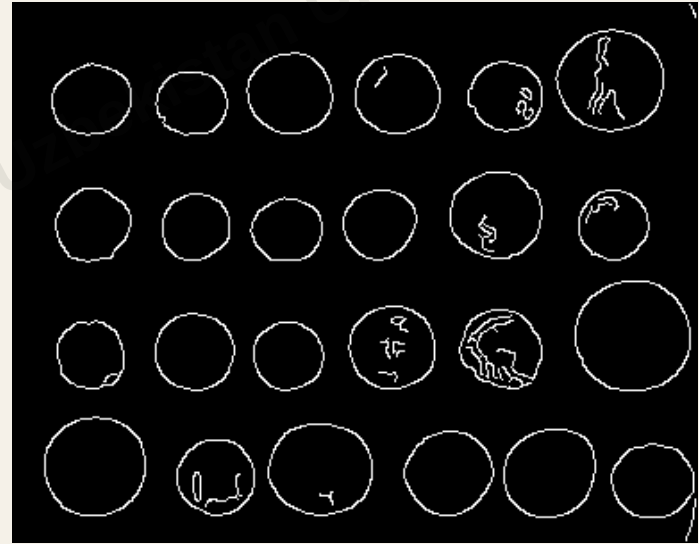
So far: we selected local detail
in space and frequency.

Next: we compare neighbours
to locate changes.

An edge is a local change in image intensity



Image measurements



Detected intensity changes

An edge map gives locations of change, not object identities.

Different causes can create edges



Object boundary

One surface ends or occludes another.

Shadow or illumination change

Light changes across the same surface.

Texture or material change

The surface itself changes appearance.

A strong edge does not prove that the object ends there.

A derivative measures how quickly intensity changes

Intensity row $I[c]$

20	20	20	20	180	180	180	180
----	----	----	----	-----	-----	-----	-----

Forward difference: $I[c+1] - I[c]$

0	0	0	160	0	0	0
---	---	---	-----	---	---	---

Large local differences suggest an edge. Flat regions give zero.

The sign records the direction of change

Patch	Right minus left	Meaning
20, 20, 180	+160	Dark to light
180, 20, 20	-160	Light to dark
20, 20, 20	0	No local change

A negative derivative is meaningful. Keep signed values.

Two image axes need two derivatives

Gx: change across columns

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

Large response at a vertical boundary.

Gy: change down rows

-1
0
1

Large response at a horizontal boundary.

We use x = column and y = row, increasing downward.

Roberts and Prewitt are classical derivative filters

Roberts: diagonal differences

1	0
0	-1

0	1
-1	0

2×2 support

Small footprint, little smoothing

Prewitt: derivative across columns

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

The second Prewitt direction uses the transposed kernel.

Sobel and Scharr change the smoothing weights

Sobel x

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

For a ramp: divide by 8

Scharr x

-3	0	3
-10	0	10
-3	0	3

For a ramp: divide by 32

Scharr improves directional consistency for small 3×3 derivative kernels.

A vertical boundary produces a horizontal gradient

10	10	90
10	10	90
10	10	90

$$G_x = 80 + 160 + 80 = 320$$

$$G_y = 0$$

Rows are identical.
Columns change sharply.

With Sobel weights divided by 8, $G_x = 40$ and $G_y = 0$.

Each derivative highlights a different direction



Smoothed input



$|G_x|$: across columns



$|G_y|$: down rows

Brightness here shows derivative strength, not original pixel intensity.

Magnitude and direction describe the gradient

$$M = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \text{atan2}(G_y, G_x)$$

`atan2` uses both components to return the angle.

Magnitude M tells us how strong the change is.

Direction θ points across the edge, toward brighter pixels.

The edge runs perpendicular to the gradient direction.

A small gradient calculation

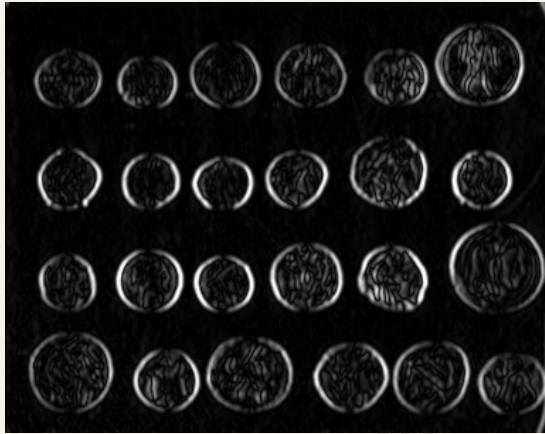
$$G_x = 3 \quad G_y = 4$$

$$M = \sqrt{3^2 + 4^2} = 5$$

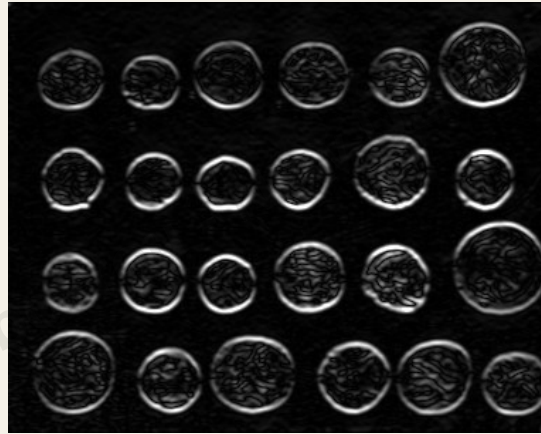
$$\theta = \text{atan2}(4,3) \approx 53.1^\circ$$

If both signs reverse, the strength stays 5 and the direction reverses.

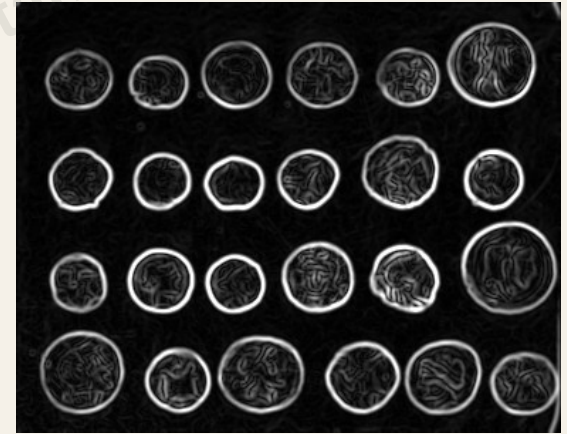
Both derivatives retain more boundary directions



$|G_x|$: left / right sides



$|G_y|$: top / bottom sides



M : directions combined

One component selects a direction. Magnitude retains varied boundary orientations.

A direct signed sum can cancel a strong edge

Gx	Gy	Gx + Gy	$\sqrt{(Gx^2 + Gy^2)}$
3	-3	0	4.24
3	4	7	5.00

Keep the components to estimate direction.
Combine their squares to estimate overall strength.

Applying two derivatives in sequence does not compute gradient magnitude.

The task determines which directions to keep

Task	Useful operation	Benefit
Inspect vertical cracks	Gx: change across columns	Emphasize the expected orientation
Locate a horizontal boundary	Gy: change down rows	Reduce responses to other orientations
Trace coins or arbitrary shapes	$\sqrt{(Gx^2 + Gy^2)}$	Retain changes in varied directions
Denoise without a preferred axis	Horizontal then vertical Gaussian	Efficient smoothing in both axes

Measurement direction and strongest edge orientation are perpendicular.

The Laplacian adds second derivatives

$$\nabla^2 I = \partial^2 I / \partial x^2 + \partial^2 I / \partial y^2$$

0	1	0
1	-4	1
0	1	0

Combines curvature along both axes.

Gives one signed response.
No separate edge direction.

With this kernel sign, $I - \alpha \nabla^2 I$ boosts local detail.

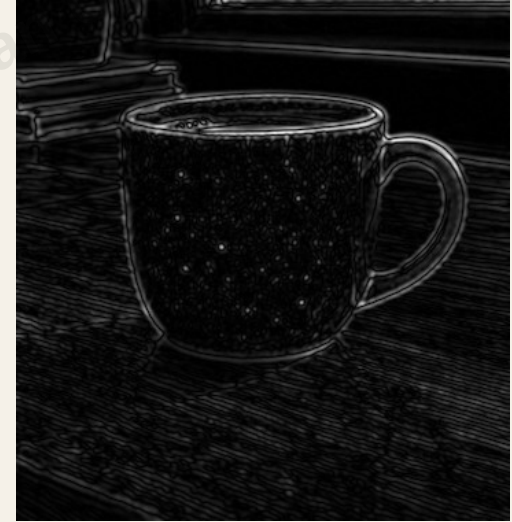
Gaussian smoothing before the Laplacian



Input



|Laplacian|



|Laplacian after blur|

LoG first sets a spatial scale, then measures second-order change at that scale.

Differentiation also responds to noise



Noisy input



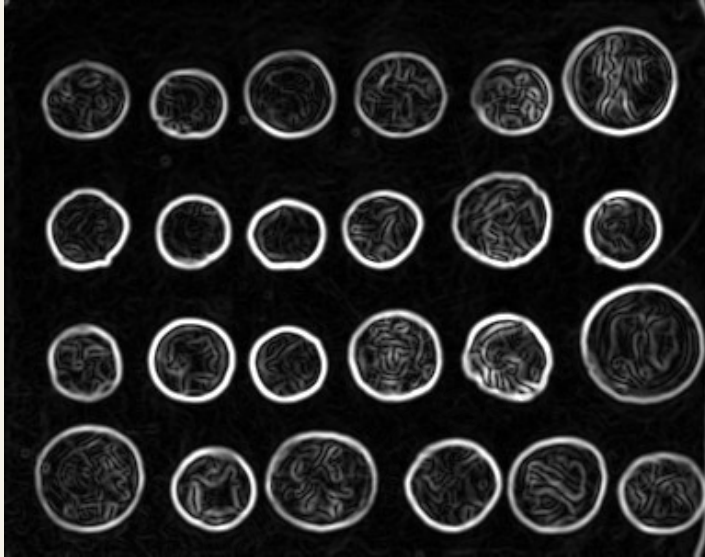
Direct gradient



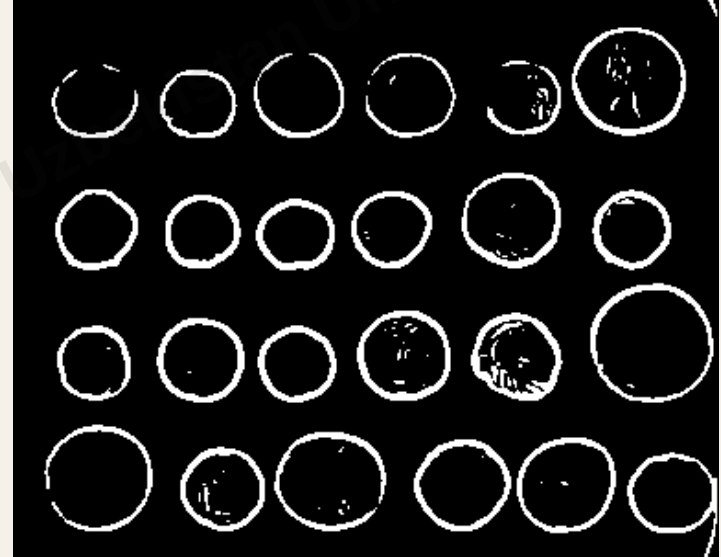
Smooth, then differentiate

Smoothing helps suppress false responses, but also weakens fine edges.

A threshold turns strength into a decision



Continuous gradient magnitude



Magnitude ≥ 0.065

Thresholding alone can produce thick, broken or cluttered edges.

Canny refines gradient responses

1	Smooth	Reduce rapid noise fluctuations
2	Differentiate	Measure magnitude and direction
3	Thin	Keep local maxima across the edge
4	Link	Keep weak candidates connected to strong ones

Each stage addresses a specific weakness of raw gradient thresholding.

Non-maximum suppression thins a ridge

Magnitudes sampled across an edge

0	2	7	4	1
---	---	---	---	---

Only the local maximum survives

0	0	7	0	0
---	---	---	---	---

Compare along the gradient direction, which crosses the edge.

Hysteresis uses strength and connectivity

10	0	0	0	6
0	6	0	0	0
0	0	6	0	0
0	0	0	0	0

High threshold = 8

Low threshold = 4

10 starts a strong edge.

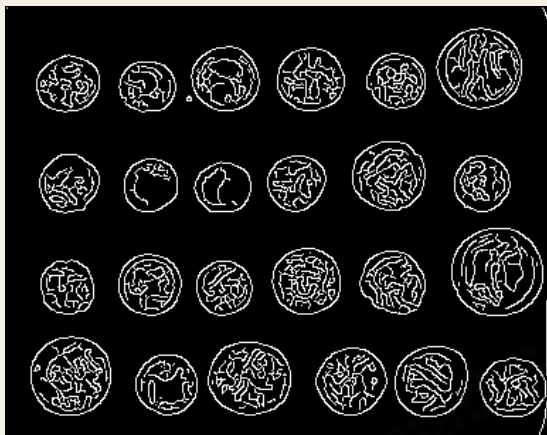
Connected 6s survive.
The isolated 6 disappears.

Connections include diagonal neighbours (8-connectivity).

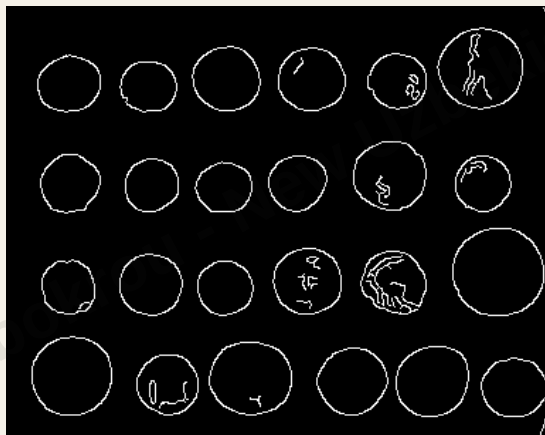
A chain of weak candidates can connect to a strong response.

Thresholds change what survives

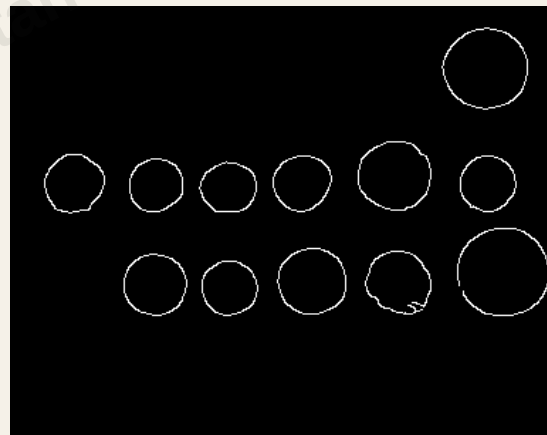
Pairs show low / high thresholds. Gaussian σ stays at 1.2 px.



Low pair: 0.010 / 0.025



Middle pair: 0.035 / 0.080

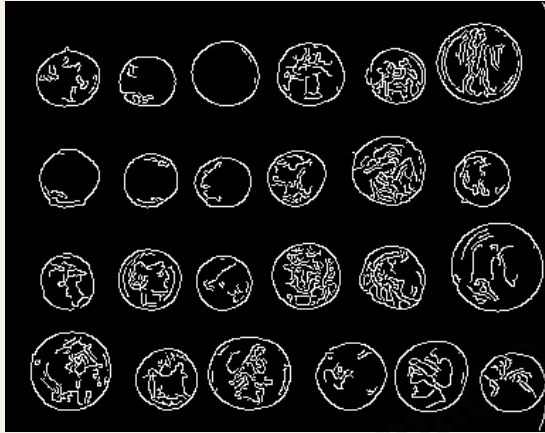


High pair: 0.080 / 0.170

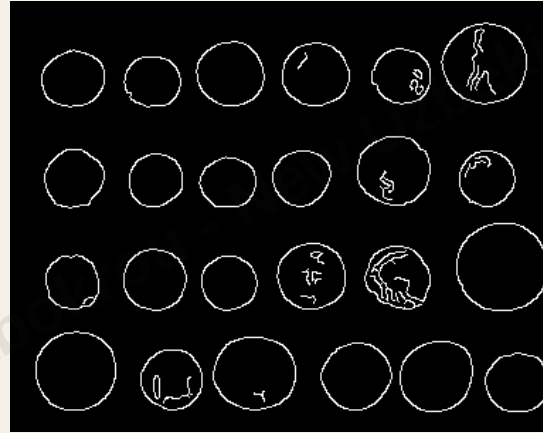
Lower thresholds keep more detail and clutter. Higher ones can break outlines.

The smoothing scale changes the edges we see

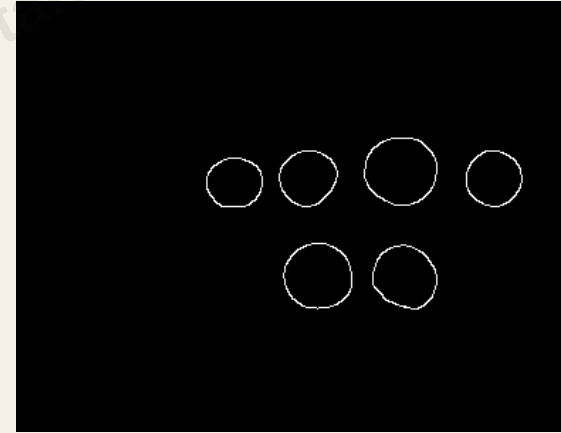
Low / high thresholds stay at 0.035 / 0.080. Only σ changes.



$\sigma = 0.6$ px



$\sigma = 1.2$ px



$\sigma = 3.0$ px

Scale determines which structures remain strong enough to detect.

Gradients respond differently to offset and contrast

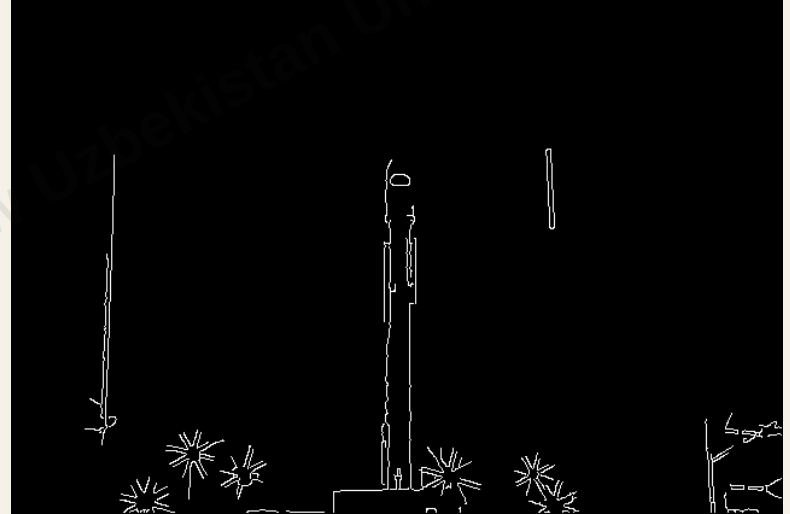
Image change	Gradient response	Condition
$J = I + b$	$\nabla J = \nabla I$	Constant b , no clipping
$J = aI$	$\nabla J = a\nabla I$	Constant scale a
Clipping or a shadow	May create or erase responses	Simple rule no longer holds

Constant brightness offsets cancel. Contrast changes can cross a threshold.

Detected edges depend on the task



Input scene



Detected intensity changes

Bright lamps add edges. Parts of the rocket outline disappear.

How would you test an edge detector?

Task: locate the outer boundary of each coin.

Decision	Your explanation
Useful structure	Which edges should count as correct?
Controlled comparison	What will you keep fixed?
Parameter change	Which single setting will you vary?
Evidence	How will you detect both missed and false edges?

More detected pixels do not automatically mean a better detector.

Filtering and edges in one view

Concept	What it contributes
Kernel	A weighted rule for neighbouring pixels
Frequency mask	Select which spatial variations survive
Separability	The same 2D result using two 1D passes
Gradient	Local strength and direction of change
Edge detector	Selected, thinned and linked responses

The useful representation depends on the task and the recording conditions.

Exit questions

1. For $h = [1, 2, 0]$, which weights overlay the input in convolution?
2. Which frequencies does a low-pass mask keep? What may be lost?
3. A 9×9 separable kernel uses how many weighted terms per pixel?
4. If $G_x = -3$ and $G_y = 4$, what is the magnitude?
5. Can an isolated weak candidate survive Canny hysteresis?

Exit answers

1. $[0, 2, 1]$: reverse the kernel before the sliding weighted sum.
2. Slow spatial variations. Fine texture and sharp transitions may weaken.
3. 18 terms: 9 horizontally and 9 vertically, instead of 81.
4. 5, because $\sqrt{(-3)^2 + 4^2} = 5$.
5. No. It needs a path of retained candidates to a strong response.

Reading and image sources

Richard Szeliski, Computer Vision: Algorithms and Applications

Second edition, 2022. Image processing and feature detection.
szeliski.org/Book

OpenCV tutorials and reference

Smoothing, image gradients, Fourier transforms and Canny
docs.opencv.org

Course examples

Mug and lens: illustrative artwork. Coins and rocket: sample photographs.
Inputs continue Sessions 1–2. Filter and edge results are computed examples.