Lecture 4

Point Operations & Filtering

Dr. Hugo Armando GUILLEN RAMIREZ

Department of Visceral Surgery and Medicine
Department of Digital Medicine
University of Bern

Spring 2026

Outline

Learning Objectives

By the end of this lecture you should be able to:

1. Apply brightness, contrast, normalisation, and gamma correction.
2. Implement and explain histogram equalisation.
3. Define 2D convolution, distinguish it from correlation, and handle boundary conditions.
4. Build common linear filters (box, Gaussian, Laplacian, DoG) from scratch.
5. Apply non-linear filters (median, bilateral) and understand their edge-preserving properties.
6. Exploit separable filter structure for computational efficiency.
7. Implement Normalised Cross-Correlation for template matching.

Point Operations

Overview

A **point operation** maps each pixel *independently* of its neighbours:

$$I'(x, y) = T\big(I(x, y)\big)$$

Properties:

- No spatial interaction $\Rightarrow$ $O(N)$ time.
- Trivially parallelisable (GPU/SIMD).
- Implemented as a *look-up table* (LUT) for speed.

Common operations:

- Brightness adjustment
- Contrast scaling
- Normalisation
- Gamma correction
- Histogram equalisation

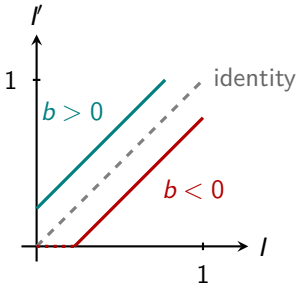
Brightness Adjustment

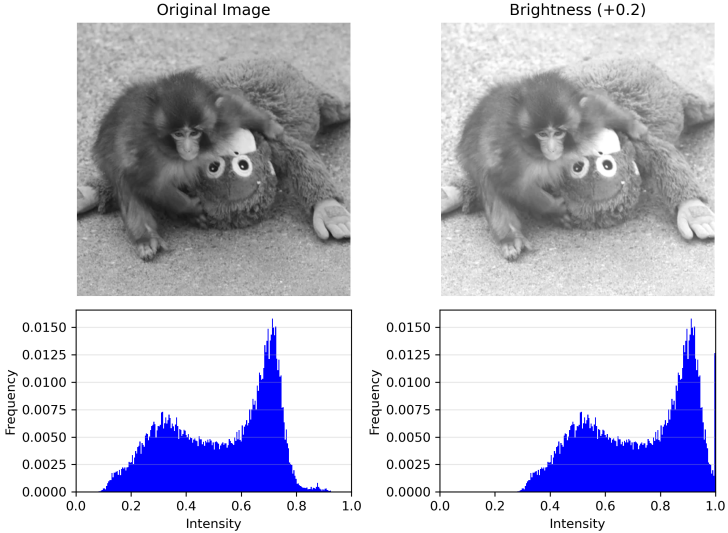
$$I'(x, y) = \mathrm{clip}\big(I(x, y) + b,\ 0,\ 1\big)$$

- $b > 0$: brighter — histogram shifts **right**.
- $b < 0$: darker — histogram shifts **left**.
- Clipping prevents values leaving $[0, 1]$.

Code

```
I_out = np.clip(I + b, 0, 1)
```

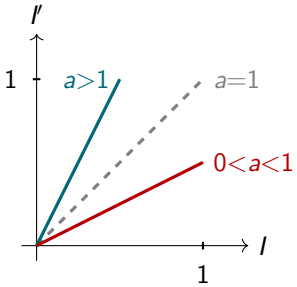


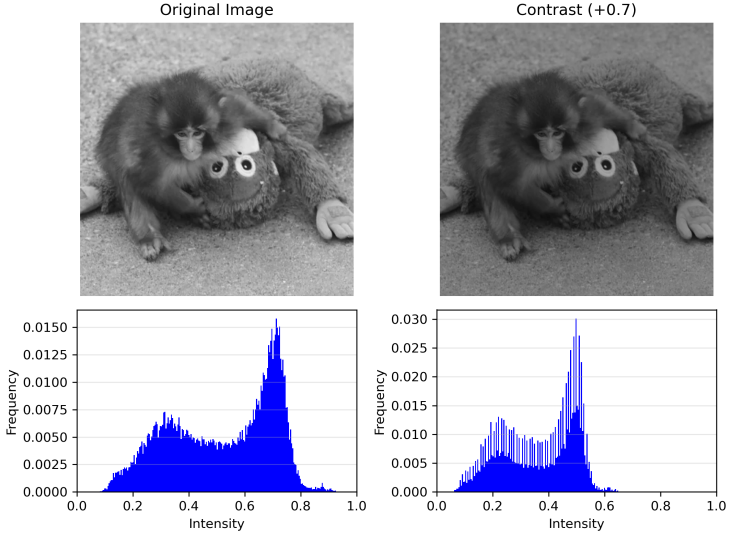

Original Image
Brightness (+0.2)
Frequency
Intensity
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0
0.2
0.4
0.6
0.8
1.0

Contrast Scaling

$$I'(x,y) = \mathrm{clip}\big(a \cdot I(x,y),\ 0,\ 1\big)$$

- $a > 1$: higher contrast — histogram **stretches**.
- $0 < a < 1$: lower contrast — histogram **compresses**.
- $a < 0$: intensity **inversion**.




Original Image
Contrast (+0.7)
Frequency
Intensity
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.0
0.2
0.4
0.6
0.8
1.0

Normalisation

Maps the image to a standard range $[0, 1]$:

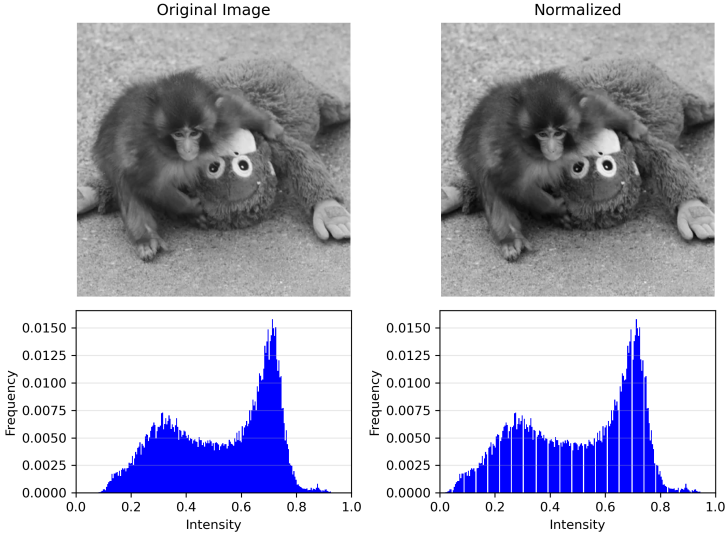
$$I'(x, y) = \frac{I(x, y) - I_{\min}}{I_{\max} - I_{\min}}$$

- Useful when the sensor uses a sub-range (e.g. a 12-bit scanner stores values in $[0, 4095]$).
- Guarantees full dynamic range is used for display.
- Special case: $\min = 0 \Rightarrow$ simple division by maximum.

Code

```
I_norm = (I - I.min())/ (I.max()- I.min()+ 1e-8)
```

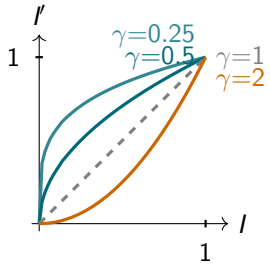
TODO: python norm



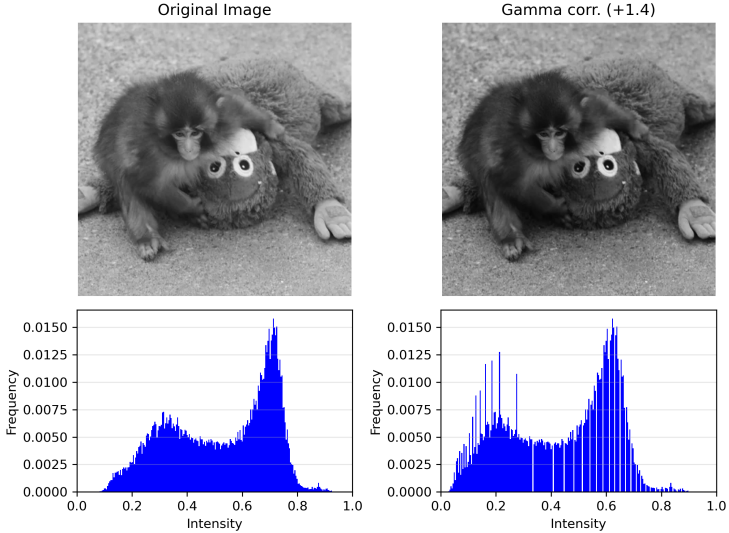
Gamma Correction

$$I'(x, y) = I(x, y)^{\gamma}$$

- $\gamma < 1$: brightens shadows (*encoding gamma*, e.g. γ=1/2.2 for sRGB).
- $\gamma > 1$: darkens highlights (*decoding gamma*).
- Human perception is *approximately logarithmic*: gamma-encoded images appear more perceptually uniform.



γ	**Effect**
0.25	Very bright (lifts shadows)
0.5	Moderately bright
1.0	Identity
2.0	Darkened
4.0	Very dark


Original Image
Gamma corr. (+1.4)
Frequency
Intensity
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0
0.2
0.4
0.6
0.8
1.0

Histogram Equalisation

Goal: redistribute intensities so the CDF becomes approximately linear.

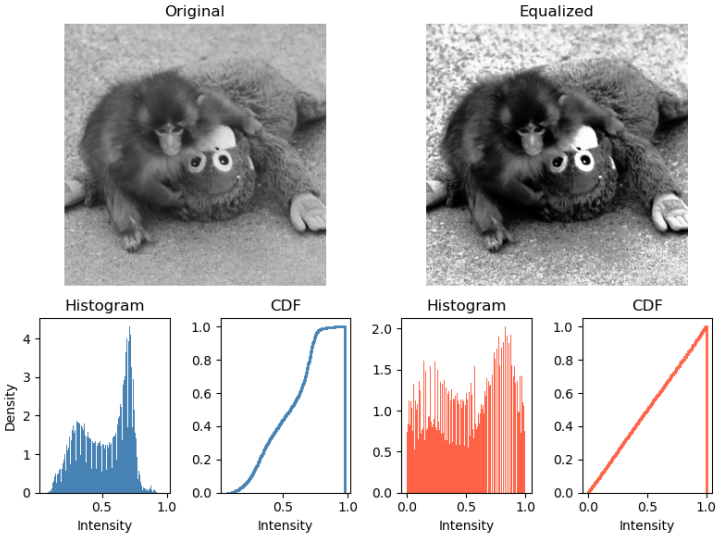
Algorithm:

1. Compute histogram $h[k]$ for $k = 0, \ldots, L-1$.
2. Compute CDF: $\mathrm{CDF}[k] = \sum_{j=0}^{k} h[j]$.
3. Normalise: $\mathrm{LUT}[k] = \left\lfloor \dfrac{\mathrm{CDF}[k] - \mathrm{CDF}_{\min}}{N - \mathrm{CDF}_{\min}} (L-1) \right\rfloor$
4. Apply: $I'(x, y) = \mathrm{LUT}[I(x, y)]$.

CDF: Cumulative Distribution Function.
LUT: Look-Up Table.
Effect: low-contrast images become globally enhanced. Best for images with a *narrow* or uneven histogram.

Limitation

May *over-enhance noise*. CLAHE (Contrast-Limited AHE) adds local clipping to mitigate this.


Original
Equalized
Histogram
CDF
Histogram
CDF
Density
Intensity
Intensity
Intensity
Intensity

Convolution Fundamentals

What is Convolution?

- **The Intuition:** Slide a small window across the signal and compute a weighted average at every position.

1-D Discrete Convolution

For a 1-D discrete signal $f[n]$ and a small discrete kernel $g[k]$, the convolution is:

$$(f * g)[n] = \sum_k f[k]\, g[n-k]$$

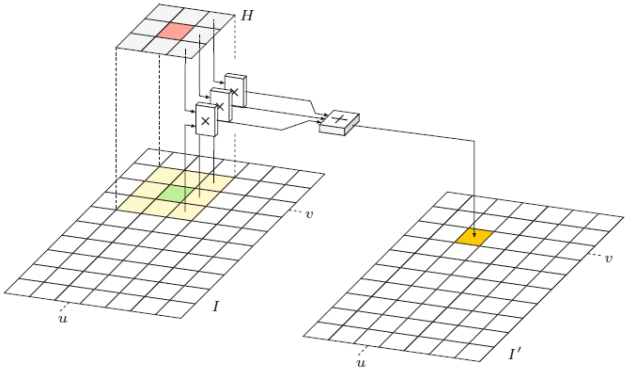
- **The Recipe: Flip** the kernel, **Shift** it, **Multiply** element by element, and **Sum** the products.
- The procedure is identical in two dimensions, just applied to a grid instead of a line.

2D Convolution

The **2D convolution** of image I with kernel K:

$$I'(x,y) = (I * K)(x,y)$$
$$= \sum_{i=-m}^{m} \sum_{j=-n}^{n} K(i,j) \cdot I(x-i,\, y-j)$$

Each output pixel $I'(x,y)$ is a weighted vote of all neighbours; the weight assigned to neighbour $I(x-i, y-j)$ is the kernel $K(i,j)$.



Correlation ($\star$) vs Convolution ($*$)

$$\text{Correlation:} \quad (I \star K)(x, y) = \sum_{i,j} K(i, j) \cdot I(x + i,\ y + j)$$

$$\text{Convolution:} \quad (I * K)(x, y) = \sum_{i,j} K(i, j) \cdot I(x - i,\ y - j)$$

- In convolution, the kernel is **flipped** before sliding:

$$K_{\text{flip}}(i, j) = K(-i, -j)$$

$$I * K \ \equiv \ I \star K_{\text{flip}}$$

- For **symmetric** kernels (Gaussian, Laplacian) the flip has no effect and convolution = correlation.

Property	**Convolution ($*$)**	**Correlation ($\star$)**
Kernel flipped?	Yes ($K_{-i,-j}$)	No
Commutative?	Yes	No
Associative?	Yes	No
Library default	`convolve2d`	`correlate2d`

Warning: Many imaging libraries use *correlation* but call it "convolution".

Identity & Shift Filters

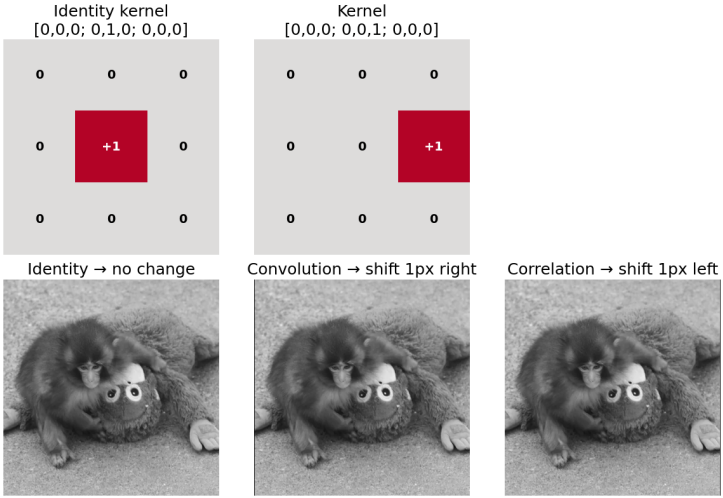
Identity kernel $K = \delta$ (identity) $\Rightarrow$ image unchanged:

$$K_{\text{id}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Centre weight $= 1 \Rightarrow$ each output pixel equals the input pixel.

$$K = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

- **Convolution**: current intensity is the pixel 1 step to the **left**.
- **Correlation**: current intensity is the pixel 1 step to the **right**.



Convolution with a $+1$ weight to the right of center samples the left neighbor, so the output image appears shifted right by 1 px.
Correlation with the same kernel samples the right neighbor, so the output image appears shifted left by 1 px.

Example

Box filter $(1/9)$ on a 3×3 patch of pixel values:

$$\underbrace{\begin{pmatrix}10 & 11 & 10\\ 9 & 10 & 11\\ 10 & 9 & 10\end{pmatrix}}_{I_{\text{patch}}} * \underbrace{\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}}_{K_{\text{box}}} = \frac{1}{9}(10+11+10+9+10+11+10+9+10) = \frac{90}{9} = \mathbf{10}$$

Weighted (ramp) filter $(1/45)$:

$$K_{\text{ramp}} = \frac{1}{45}\begin{pmatrix}9 & 8 & 7\\ 6 & 5 & 4\\ 3 & 2 & 1\end{pmatrix} \quad \Rightarrow \quad \frac{1}{45}(10\cdot9+11\cdot8+10\cdot7+9\cdot6+10\cdot5+11\cdot4+10\cdot3+9\cdot2+10\cdot1) = \frac{454}{45} \approx \mathbf{10}$$

Observation

The patch is locally smooth and results are almost identical. The weighted filter shifts the output slightly towards the top-left neighbours.

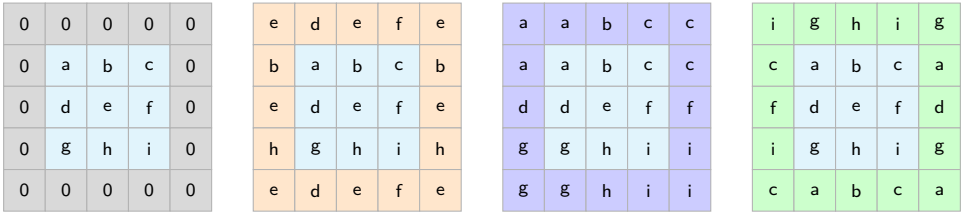
δ			Box			Ramp		
0	0	0	0.11	0.11	0.11	0.20	0.18	0.16
0	1.00	0	0.11	0.11	0.11	0.13	0.11	0.09
0	0	0	0.11	0.11	0.11	0.07	0.04	0.02



Boundary Handling

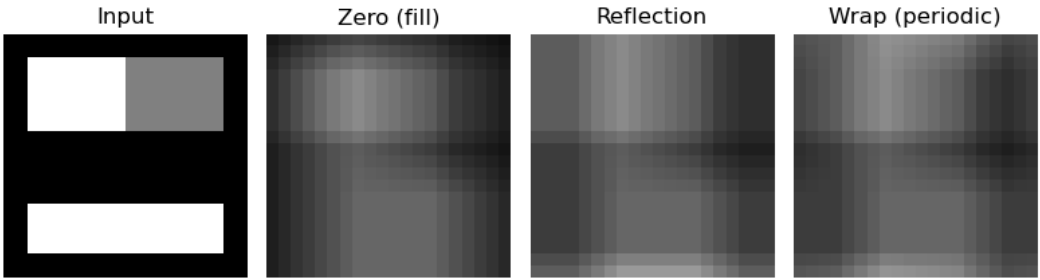
When the kernel extends beyond the border, missing pixels must be filled:

Mode	Description	Best for
Zero-padding	Missing pixels = 0	General purpose, FFT
Reflection	Mirror across border	Smooth images
Replication	Repeat edge pixel	Preserving boundary values
Wrap	Periodic extension	Frequency-domain work



Linear Filters

20x20 image, box K=10



Frequency Pass Filters

- **Low-Pass Filter (LPF):** Allows low frequencies to pass, attenuates high frequencies.
  - **Effect:** Smooths and blurs the image.
  - **Example:** Gaussian filter (purely low-pass, no sidelobes).
- **High-Pass Filter (HPF):** Allows high frequencies to pass, attenuates low (spatial) frequencies.
  - **Effect:** Extracts fine detail and sharp transitions.
  - **Example:** Laplacian filter, which has a strong response at edges and fine detail.

Side Lobes & Ringing:

- Sharp boundaries in one domain (like a rectangular Box filter in space) create oscillating **sidelobes** in the other domain (frequency).
- **Sidelobes** allow unintended frequencies to "leak" through the filter.
- **Consequence:** This leakage causes ringing or Gibbs artefacts near sharp edges in the processed image.

Design Goal

A high-quality smoothing filter (like the Gaussian) is designed to minimize or eliminate side lobes to prevent these visual artefacts.

Gaussian (LPF)

0.00	0.01	0.02	0.01	0.00
0.01	0.06	0.10	0.06	0.01
0.02	0.10	0.16	0.10	0.02
0.01	0.06	0.10	0.06	0.01
0.00	0.01	0.02	0.01	0.00

Laplacian (HPF)

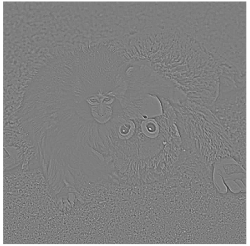
0	-1.00	0
-1.00	4.00	-1.00
0	-1.00	0

Box (Sharp Boundaries)

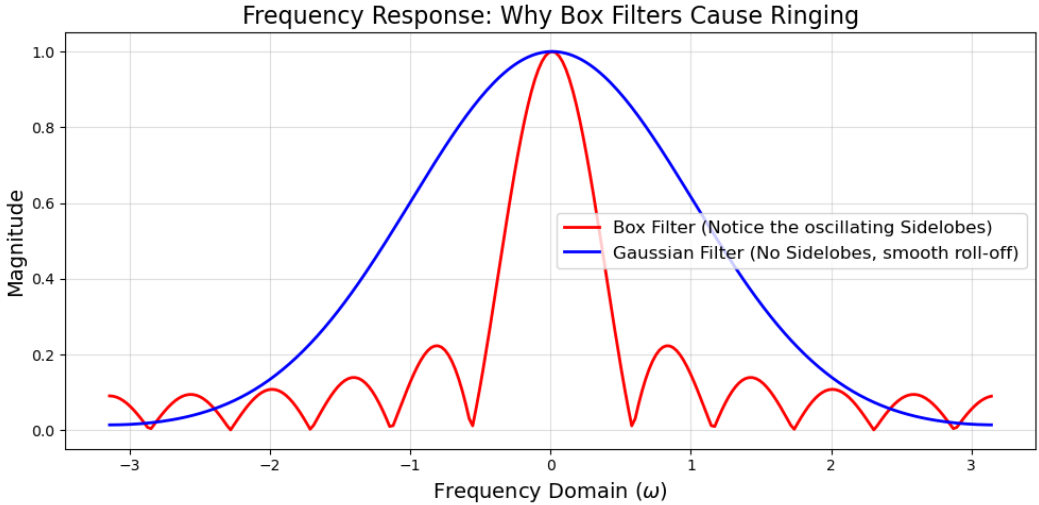
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

Smoothed / Blurred

Edges Extracted



Smoothed (Prone to Ringing)


Frequency Response: Why Box Filters Cause Ringing
Magnitude
Frequency Domain (ω)
Box Filter (Notice the oscillating Sidelobes)
Gaussian Filter (No Sidelobes, smooth roll-off)
1.0
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3

Revisiting the Convolution Theorem

Convolution Theorem

$$\mathcal{F}\{I * K\} = \mathcal{F}\{I\} \cdot \mathcal{F}\{K\}$$

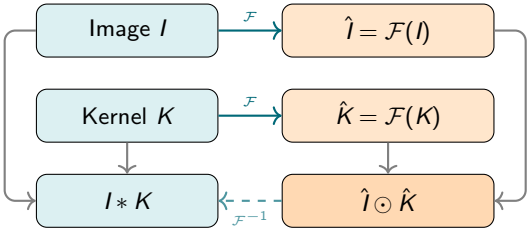
Spatial-domain convolution corresponds to pointwise multiplication in the Fourier domain.

Consequences:

- A narrow kernel $\Leftrightarrow$ broad spectrum (weak filter).
- A wide kernel $\Leftrightarrow$ narrow spectrum (strong low-pass).

Fun facts:

- Large convolutions can be done via FFT in $O(N^2 \log N)$ instead of $O(N^2 M^2)$.
- A Gaussian kernel has a **Gaussian spectrum** — purely low-pass, no sidelobes.



Separable Convolution

A 2-D kernel K is **separable** if:

$$K = \mathbf{h}_1 \cdot \mathbf{h}_2^\top$$

Replace one $M \times M$ 2-D convolution with *two* 1-D convolutions.

Complexity comparison
($N \times N$ image, $M \times M$ kernel):

	Mults/pixel	Total
Full 2-D	M^2	$N^2 M^2$
2×1-D	$2M$	$2N^2 M$
Speed-up	$M/2$	—

Which kernels are separable?

- Gaussian: $G(x, y) = g(x) \cdot g(y)$ ✓
- Box filter: $K = \frac{1}{M} \mathbf{1}_M \mathbf{1}_M^\top$ ✓
- Laplacian: **not** separable ×
- Median: **not** separable ×

Test: SVD (singular value decomposition) of K; if rank $= 1$:

$$K = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top \Rightarrow h_1 = \sqrt{\sigma_1}\, \mathbf{u}_1; h_2 = \sqrt{\sigma_1}\, \mathbf{v}_1$$

Separable Convolutions via SVD

The Theory: SVD and Rank-1 Matrices

- A 2D kernel K is separable if and only if $\text{rank}(K) = 1$.
- Using Singular Value Decomposition (SVD), a rank-1 matrix factors to:

$$K = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top$$

- To get the 1D filters, distribute the scaling factor σ_1:

$$\mathbf{h} = \sqrt{\sigma_1}\mathbf{u}_1 \quad \text{and} \quad \mathbf{v} = \sqrt{\sigma_1}\mathbf{v}_1$$

Example: 3x3 Gaussian Blur

- Consider an unnormalized 2D Gaussian kernel:

$$K = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- The rows are clearly multiples of each other (Rank = 1). Factoring it out yields our two 1D filters:

$$K = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}}_{\text{Vertical } \mathbf{h}} \underbrace{\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}}_{\text{Horizontal } \mathbf{v}^\top}$$

- **Efficiency Gain:** Operations per pixel drop from $3 \times 3 = 9$ to $3 + 3 = 6$.

Exercise break (~10 min)

- Run the cells in the Section “Part 4 — Separable Convolution”
- Examine the implementation of `separable_convolve`.
- What speed-up are you able to achieve?

Linear Filters

A **linear filter** replaces each pixel with a *weighted sum* of its neighbourhood — this is exactly the convolution operation defined above.

Examples covered today:

1. Box filter (uniform blur)
2. Gaussian filter
3. Laplacian (edge detection / sharpening)
4. Difference of Gaussians (DoG)

What to look for:

- Do kernel entries sum to 1? (preserves mean)
- Do they sum to 0? (edge detector)
- Is the kernel separable? (efficiency)
- Is the kernel symmetric? (convolution = correlation)

Box Filter (Uniform Blur)

All M^2 kernel entries are equal:

$$K_{\text{box}} = \frac{1}{M^2} \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}_{M \times M}$$

Properties:

- Equivalent to a **moving average**.
- Separable: $K = \frac{1}{M}\mathbf{1}_M \cdot \frac{1}{M}\mathbf{1}_M^\top$.
- Can be implemented in $O(N^2)$ *regardless of* M using a **sliding sum**.

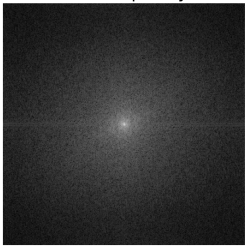
Disadvantage

The rectangular window has large **sidelobes** in the Fourier domain $\Rightarrow$ ringing / Gibbs artefacts near sharp edges. Use Gaussian instead for smooth blurring.

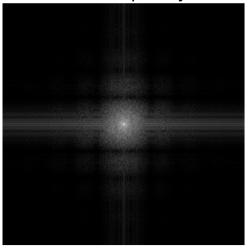
Original

After box filter ($M = 9$)

Raw frequency



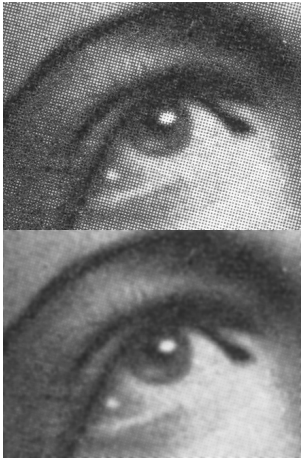
Raw frequency



Gaussian Filter — Physical Motivation

A Gaussian is a good model because:

- Models many **physical processes** via the central limit theorem (sum of many small independent effects $\rightarrow$ Gaussian).
- Defocused optics has a Gaussian **point-spread function** (not a box).



Separability identity

$$G_\sigma(x, y) = G_\sigma(x) \cdot G_\sigma(y)$$

2-D Gaussian = outer product of two 1-D Gaussians.

Gaussian Filter

$$K(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$$

Discretised and normalised so all entries sum to 1.

Key properties:

- σ controls blur amount: larger σ = stronger.
- **No sidelobes** in frequency domain $\rightarrow$ no ringing.
- **Separable**: $K(x, y) = g(x) \cdot g(y)$.
- Rule of thumb: kernel size $\approx 6\sigma$.
- Linear $\Rightarrow$ can be combined with other linear filters.

5×5 kernel ($\sigma = 1$):

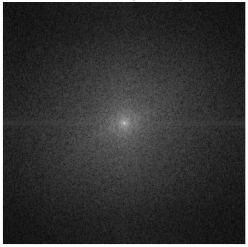
.003	.013	.022	.013	.003
.013	.059	.097	.059	.013
.022	.097	**.159**	.097	.022
.013	.059	.097	.059	.013
.003	.013	.022	.013	.003

Sum of entries = 1.000

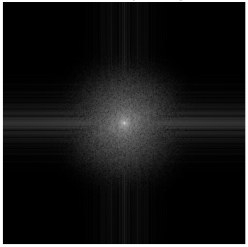
Original

Gaussian kernel ($M = 9,\ \sigma = 1.5$)

Raw frequency



Raw frequency



Analytic Proof of Gaussian Separability

The 2-D isotropic Gaussian:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\, e^{-(x^2+y^2)/(2\sigma^2)}$$

factors as:

$$G_\sigma(x,y) = \underbrace{\frac{1}{\sqrt{2\pi}\sigma}\, e^{-x^2/(2\sigma^2)}}_{g_\sigma(x)} \cdot \underbrace{\frac{1}{\sqrt{2\pi}\sigma}\, e^{-y^2/(2\sigma^2)}}_{g_\sigma(y)}$$

Any 2-D Gaussian convolution can be replaced by **two 1-D passes** (row-wise then column-wise) with identical 1-D Gaussian kernels.

Foundation of the Gaussian pyramid:

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sqrt{\sigma_1^2+\sigma_2^2}}$$

Smoothing twice with $\sigma{=}1$ is identical to smoothing once with $\sigma{=}\sqrt{2}$.

Advanced Filters

Laplacian Filter (Edge Detection)

The **Laplacian** is the sum of second-order partial derivatives:

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

Discrete approximation:

$$K_{\text{Lap}} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

- Strong response at **edges and fine detail**.
- Sum of entries $= 0 \Rightarrow$ zero response on flat regions.
- Highly **noise-sensitive** — always pair with Gaussian.

Laplacian sharpening:

$$I_{\text{sharp}} = I - \lambda \nabla^2 I, \quad \lambda > 0$$

Adds the negated Laplacian back to enhance edges.

$$2I - \tfrac{1}{9}\text{box} \approx I_{\text{sharp}}$$

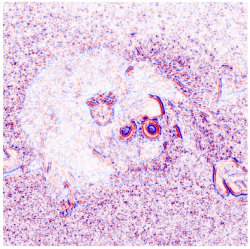
Original

Sharpened (λ=0.5)

Laplacian kernel

0	-1	0
-1	4	-1
0	-1	0

Laplacian response

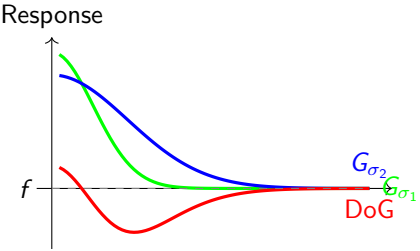


Difference of Gaussians (DoG)

$$\mathrm{DoG}_{\sigma_1,\sigma_2} = G_{\sigma_1} * I - G_{\sigma_2} * I, \qquad \sigma_1 < \sigma_2$$

Properties:

- **Band-pass filter** — passes spatial frequencies between the two scales.
- Approximates the **Laplacian of Gaussian** (LoG) when $\sigma_2/\sigma_1 \approx 1.6$.
- Used in **SIFT** feature detection (Lowe, 2004).

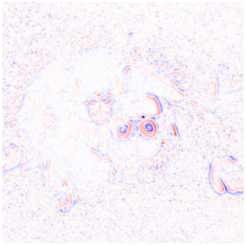


Original

G1 (σ=1.5)

G1 (σ=2.4)

DoG (G1-G2)



Median Filter

Replaces each pixel with the **median** of its neighbourhood (not a weighted average).

Why median?

- **Non-linear**: cannot be expressed as a convolution.
- Extremely effective for **salt-and-pepper** noise (corrupted pixels are outliers; median ignores them).
- **Edge-preserving**: median rarely crosses a strong edge.
- Complexity: $O(N^2 w^2 \log w)$ for $w \times w$ window.

Example — 3×3 window:

$$\begin{pmatrix} 10 & 11 & 9 \\ 255 & 10 & 10 \\ 9 & 11 & 10 \end{pmatrix}$$

Median = **10** (the corrupted 255 is discarded).

vs. Gaussian

Gaussian would give ≈ 38 (badly corrupted by the outlier).

Original

Salt & Pepper Noise

Gaussian Filter

Median Filter

Salt and pepper noise

```
img_sp_noise = IMG.copy()
rand_matrix = np.random.rand(*IMG.shape)
# Salt
img_sp_noise[rand_matrix > 0.95] = 1.0
# Pepper
img_sp_noise[rand_matrix < 0.05] = 0.0
```

Bilateral Filter

$$I'(x,y) = \frac{1}{W_{xy}} \sum_{(i,j)\in\Omega} I(i,j) \underbrace{e^{-\frac{(x-i)^2+(y-j)^2}{2\sigma_s^2}}}_{w_s\text{: spatial}} \underbrace{e^{-\frac{(I(x,y)-I(i,j))^2}{2\sigma_r^2}}}_{w_r\text{: range}}$$

Parameters:

- σ_s: spatial reach (how far away to look).
- σ_r: intensity tolerance (how different is still averaged).

Properties:

- **Edge-preserving**: pixels across a strong edge have very different intensities $\Rightarrow$ small $w_r \Rightarrow$ not averaged together.
- In smooth regions $w_r \approx 1 \Rightarrow$ acts like Gaussian.
- **Non-linear** — weights depend on image content.

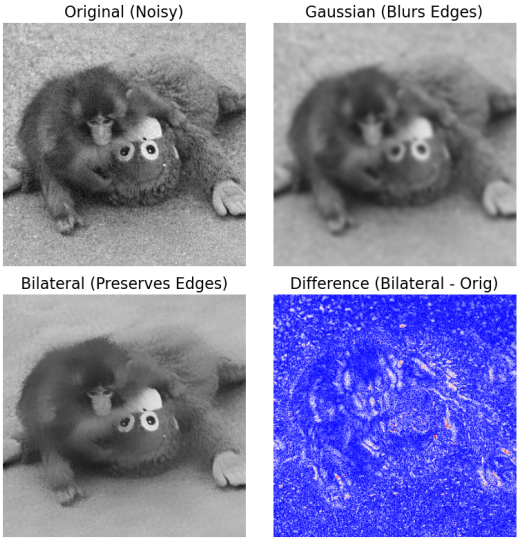
Intuition

Two weights: one for *how close*, one for *how similar*.

Average only neighbours that are *both* nearby in space *and* similar in intensity.

Applications:

- HDR tone mapping
- Skin retouching
- MRI/CT denoising


Original (Noisy)
Gaussian (Blurs Edges)
Bilateral (Preserves Edges)
Difference (Bilateral - Orig)

Template Matching and NCC

Correlation Again: Filters as Templates

Correlation with filter $F =$ **dot product** between the filter vector and the image patch:

$$I'(x, y) = F \cdot \text{vec}(I_{xy}) = |F|\,|I_{xy}| \cos\theta$$

where θ is the angle between the filter and patch vectors. The dot product is **maximised** when the patch looks *exactly like the filter*:

- Edge filter $\Rightarrow$ fires at edges.
- Gaussian blob filter $\Rightarrow$ fires at blobs.
- Template $\Rightarrow$ fires where object appears.

Normalised Correlation

Divide by $|F|\,|I_{xy}|$ to remove magnitude dependence:

$$\hat{I}'(x, y) = \cos\theta$$

Pure *similarity* measure in $[-1, 1]$.

Normalised Cross-Correlation (NCC)

$$\mathrm{NCC}(x,y) = \frac{\sum_{i,j} \left(I_{xy}(i,j) - \bar{I}_{xy}\right)\left(T(i,j) - \bar{T}\right)}{\sqrt{\sum_{i,j} \left(I_{xy}(i,j) - \bar{I}_{xy}\right)^2 \cdot \sum_{i,j} \left(T(i,j) - \bar{T}\right)^2}}$$

- Range $[-1, 1]$: $+1$ = perfect match, -1 = perfect inverse.
- **Invariant** to additive and multiplicative intensity changes.
- Complexity: $O(H \cdot W \cdot t_H \cdot t_W)$.
- FFT acceleration: $O(HW \log HW)$.

Applications: face detection, logo recognition, cell counting, image registration.

Limitations

- Not invariant to rotation, scale, or viewpoint.
- Degrades with partial occlusion.
- Cost scales with template size.

NCC: σ_I/σ_T Formulation

Alternative equivalent expression that makes invariances *explicit*:

$$NCC(u,v) = \frac{1}{N} \sum_{x=-K}^{K} \sum_{y=-K}^{K} \frac{(I(u+x, v+y) - \bar{I}_{u,v})(T(x,y) - \bar{T})}{\sigma_{I_{u,v}} \sigma_T}$$

(u, v) : current pixel in the target image
$T(x, y)$: template pixel at local coordinates (x, y)
N : total number of pixels in the template
$\bar{I}_{u,v}$: mean intensity of the patch centered at (u, v)
$\bar{T}$: mean intensity of the template
$\sigma_{I_{u,v}}$: standard deviation of the patch centered at (u, v)
σ_T : standard deviation of the template

Not linear!

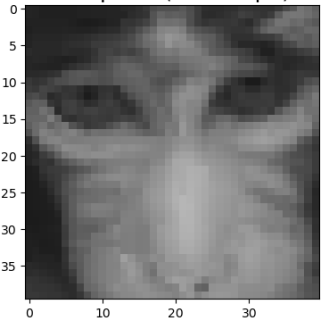
The denominator depends on image content.

NCC *cannot* be expressed as a single convolution(it requires computing local statistics).

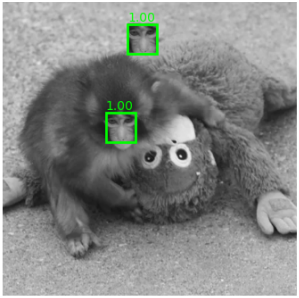
Two explicit invariances:

1. **Additive invariance**
   Subtracting the mean removes DC offset (bias lighting, background level).
2. **Multiplicative invariance**
   Dividing by σ removes gain (exposure level, reflectance scale).

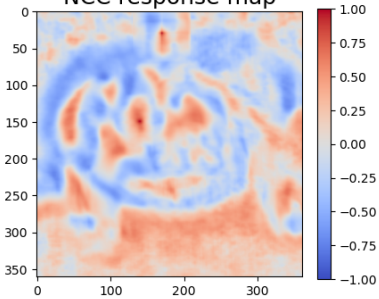
Template (40×40 px)



Original+planted template



NCC response map

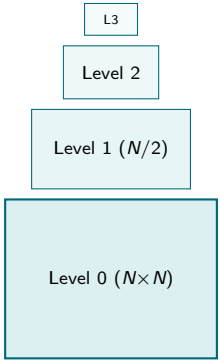


Gaussian pyramids for different scales

When finding hands or faces, we don't know what size they will be in a particular image. Template size is constant, but image size changes.

Multi-scale NCC strategy:

1. Build a **Gaussian pyramid** of the search image (smooth + downsample $2\times$ per level).
2. Apply NCC with a *fixed-size* template at each pyramid level.
3. The level where NCC score is highest indicates the object's scale.
4. Refine around the coarse-scale peak at finer resolution.



Complexity: Geometric series $\Rightarrow$ all extra pyramid levels add only $\frac{1}{3}$ more work:

$$N^2\left(1+\tfrac{1}{4}+\tfrac{1}{16}+\cdots\right)=\tfrac{4}{3}N^2$$

Summary

Connections to Medical Imaging

- **Windowing (brightness/contrast):** CT DICOM viewers map Hounsfield units via window/level point operation.
- **Histogram equalisation:** chest X-ray enhancement — apply carefully to avoid noise amplification.
- **Gaussian pre-filtering:** anti-aliasing before MRI reconstruction; scale-space analysis for lesion detection.
- **Bilateral / NLM denoising:** preserves thin structures (vessels, sulci) while smoothing.
- **Template matching:** microscopy cell counting, cardiac landmark detection, surgical tool tracking.

Lecture Summary

Topic	Key Take-away
Brightness/Contrast	Additive/multiplicative point ops; clip to valid range.
Normalisation	Min–max rescales intensities to $[0, 1]$; preserves ordering, not histogram shape.
Gamma correction	Power-law remapping; $\gamma < 1$ brightens, $\gamma > 1$ darkens (sRGB is often approximated by $\gamma \approx 1/2.2$ for encoding).
Histogram equalisation	CDF-based remapping that makes the histogram more uniform; can over-enhance noise.
Convolution vs Corr.	Convolution flips the kernel; same as correlation for symmetric kernels.
Warm-up filters	Identity leaves the image unchanged; an off-centre weight produces a shift (direction depends on conv. vs corr.).
Boundary handling	Zero / reflect / replicate / wrap — border choice affects results.
Convolution Theorem	Spatial convolution corresponds to pointwise multiplication in Fourier; Gaussian is low-pass.
Separable conv.	Rank-1 kernels reduce cost from M^2 to $2M$ ops/pixel ($\approx M/2\times$ speed-up).
Box filter	Moving average; separable; can be $O(1)$ per pixel with sliding sums/integral images; prone to ringing.
Gaussian filter	Smooth low-pass filter; controlled by σ; separable; no sidelobes/ringing.

Lecture Summary (cont)

Topic	Key Take-away
Gaussian physical basis	Motivated by CLT-style aggregation and a useful blur model; closed under convolution.
σ vs noise grid	Larger σ suppresses more noise but blurs fine structure.
$G(x)\,G(y)$ identity	2-D Gaussian is the outer product of two 1-D Gaussians.
Laplacian / DoG	Zero-sum edge/detail operators; noise-sensitive; DoG approximates LoG and underlies SIFT scale-space.
Median filter	Best for salt & pepper noise; edge-preserving; non-linear.
Bilateral filter	Spatial + range weights; edge-preserving denoising.
Filters as templates	Correlation is a dot product: strong response where image and filter are similar.
NCC template matching	Score in $[-1, 1]$; invariant to additive bias and positive gain changes; FFT-accelerable.
NCC σ formulation	σ_I, σ_T in the denominator make bias/gain invariance explicit.
Scale-space matching	Peak NCC across pyramid levels indicates the best matching object scale.

Learning Goals — Self-Check

Can you now...

1. ✓ **Apply** brightness, contrast, normalisation, and gamma correction.
2. ✓ **Explain and implement** histogram equalisation.
3. ✓ **Define** 2-D convolution, distinguish it from correlation, and explain boundary handling.
4. ✓ **Build** box, Gaussian, Laplacian, and DoG filters from scratch.
5. ✓ **Compare** linear and non-linear filters, including median and bilateral filtering.
6. ✓ **Use** separability to reduce computational cost.
7. ✓ **Implement and interpret** Normalised Cross-Correlation for template matching.

If not...

Review the corresponding slides and work through the examples.

Thank you for attending!

Questions?

hugo.guillenramirez@unibe.ch

Next week: Lecture 5: Edge Detection & Morphological Operations

Assignment 1 is released today.

See you next week!

References

1. Gonzalez, R.C. & Woods, R.E. (2018). Digital Image Processing (4th ed.). Pearson.
2. Burger, W. & Burge, M. (2016). Digital Image Processing: An Algorithmic Introduction Using Java (2nd ed.). Springer.