

UNIT 2

Image Enhancement

(12 Hours) — Image Processing (CACS404)

⚠ Note: This may contain errors — please verify

These notes were generated with the help of AI and **may contain errors** (wrong formulas, numerical slips, missing details or syllabus mismatches). Please cross-check every important definition, equation and worked example with your prescribed textbook, class notes and lecturer before relying on them for exams or practicals.

2.1 Introduction to Image Enhancement

Image enhancement processes an image so that the result is more suitable than the original for a *specific application*. It is subjective and problem-oriented. Two broad categories exist:

- **Spatial-domain methods** — operate directly on pixels: $g(x, y) = T[f(x, y)]$, where T is an operator defined over a neighbourhood of (x, y) .
- **Frequency-domain methods** — modify the Fourier (or other) transform of the image and then take the inverse transform.

2.2 Image Transforms

A transform converts an image from the spatial domain into another domain (frequency, sequency ...) where processing is easier or more meaningful. A general 2-D forward and inverse transform pair is:

$$T(u, v) = \sum_x \sum_y f(x, y) \cdot r(x, y, u, v) \quad f(x, y) = \sum_u \sum_v T(u, v) \cdot s(x, y, u, v)$$

where r and s are the forward and inverse **kernels** (basis functions). A kernel is **separable** if $r(x, y, u, v) = r_1(x, u) \cdot r_2(y, v)$, allowing a 2-D transform to be computed as two 1-D transforms (rows, then columns).

2.2.1 Fourier Transform (FT)

Any signal can be written as a sum of sines and cosines of different frequencies. The 1-D continuous Fourier transform is $F(u) = \int f(x) e^{-j2\pi ux} dx$, with inverse $f(x) = \int F(u) e^{j2\pi ux} du$.

2.2.2 Discrete Fourier Transform (DFT)

For an $M \times N$ image $f(x, y)$:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cdot e^{-j2\pi(ux/M + vy/N)}, u = 0 \dots M-1, v = 0 \dots N-1$$

$$f(x, y) = (1/MN) \sum_u \sum_v F(u, v) \cdot e^{+j2\pi(ux/M + vy/N)}$$

$F(u, v)$ is complex: $F = R + jI$. **Magnitude spectrum** $|F| = \sqrt{R^2 + I^2}$, **phase angle** $\varphi = \tan^{-1}(I/R)$, **power spectrum** $P = |F|^2$. The magnitude carries the 'how much' of each frequency; the phase carries the structural (location) information.

Properties of the 2-D DFT

Property	Statement
Separability	2-D DFT = 1-D DFT of rows followed by 1-D DFT of columns
Linearity	$a \cdot f_1 + b \cdot f_2 \Leftrightarrow a \cdot F_1 + b \cdot F_2$
Translation	$f(x-x_0, y-y_0) \Leftrightarrow F(u, v) \cdot e^{-j2\pi(ux_0/M + vy_0/N)}$; magnitude is unchanged
Periodicity	$F(u, v) = F(u+M, v) = F(u, v+N)$
Conjugate symmetry	$F(u, v) = F^*(-u, -v)$ for real f ; $ F $ is symmetric
Rotation	Rotating f by θ rotates $F(u, v)$ by the same θ
Scaling	$f(ax, by) \Leftrightarrow (1/ ab) F(u/a, v/b)$: compressing in space expands in frequency
Average value	$F(0, 0) = MN \times (\text{mean of } f)$; the 'DC' term
Centering	$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ — moves the origin to the centre of the spectrum

Example 2.1 — 1-D DFT

Compute the 4-point DFT of $f(x) = \{1, 2, 3, 4\}$. $F(u) = \sum f(x) e^{-j2\pi ux/4} = \sum f(x)(-j)^{ux}$.

Solution

$$F(0) = 1 + 2 + 3 + 4 = \mathbf{10}$$

$$F(1) = 1 + 2(-j) + 3(-1) + 4(j) = (1 - 3) + j(-2 + 4) = \mathbf{-2 + j2}$$

$$F(2) = 1 + 2(-1) + 3(1) + 4(-1) = \mathbf{-2}$$

$$F(3) = 1 + 2(j) + 3(-1) + 4(-j) = (1 - 3) + j(2 - 4) = \mathbf{-2 - j2}$$

Check: $F(0)/N = 2.5$ is the mean of the data; $F(3) = F^*(1)$ as expected for real input.

2.2.3 Fast Fourier Transform (FFT)

Direct computation of an N -point DFT needs N^2 complex multiplications. The **FFT** (Cooley–Tukey, 1965) exploits the symmetry and periodicity of $W_N = e^{(-j2\pi/N)}$ to split an N -point DFT into two $N/2$ -point DFTs (even- and odd-indexed samples), recursively, requiring only $(N/2) \cdot \log_2 N$ multiplications — complexity **$O(N \log_2 N)$** instead of $O(N^2)$. N must be a power of 2 (zero-padding is used otherwise).

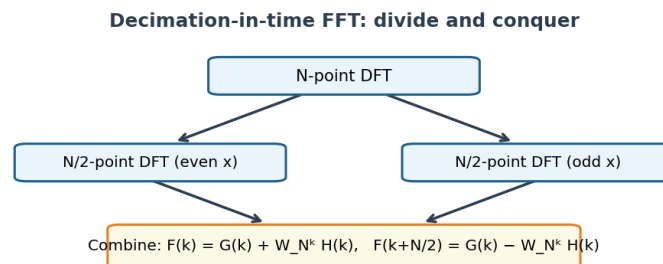


Figure 2.1 — Decimation-in-time FFT splits the DFT into even- and odd-indexed halves

With $F(k) = G(k) + W_N^k H(k)$ and $F(k + N/2) = G(k) - W_N^k H(k)$, where G and H are the DFTs of even and odd samples. This pair of equations is called a **butterfly**. For $N = 1024$, direct DFT needs $\approx 1,048,576$ multiplications whereas the FFT needs only $\approx 5,120$ — about 200× fewer.

2.2.4 Discrete Cosine Transform (DCT)

$C(u, v) = \alpha(u) \alpha(v) \sum_x \sum_y f(x, y) \cos[(2x+1)u\pi / 2M] \cos[(2y+1)v\pi / 2N]$, where $\alpha(0) = \sqrt{1/N}$ and $\alpha(u \neq 0) = \sqrt{2/N}$.

- The DCT is **real-valued** (no complex arithmetic) and has excellent **energy compaction**: most of the image energy is packed into a few low-frequency coefficients at the top-left corner.
- It implicitly assumes an even-symmetric extension of the image, so it avoids the boundary discontinuities (and resulting artefacts) of the DFT.
- It is the core of **JPEG** and **MPEG** compression (applied on 8×8 blocks — see Unit 4).

2.3 Frequency-Domain Image Enhancement

Idea: low frequencies correspond to slowly varying intensity (smooth regions, background); high frequencies correspond to rapid changes (edges, noise, fine detail). Filtering selects which frequencies pass.

Steps of Frequency-Domain Filtering: $G(u,v) = H(u,v) \cdot F(u,v)$

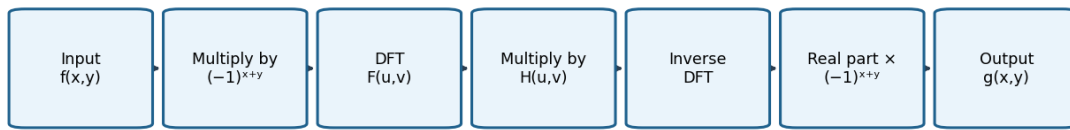


Figure 2.2 — Frequency-domain filtering procedure

$G(u, v) = H(u, v) \cdot F(u, v)$, where H is the filter transfer function. $D(u, v) = \sqrt{[(u - M/2)^2 + (v - N/2)^2]}$ is the distance from the centre of the (centred) spectrum, and D_0 is the cut-off frequency.

2.3.1 Low-Pass Filtering (Smoothing)

Attenuates high frequencies and keeps low ones → blurs the image, reduces noise.

Filter	Transfer function $H(u, v)$	Remarks
Ideal LPF (ILPF)	1 if $D \leq D_0$; 0 if $D > D_0$	Sharp cut-off; causes ringing artefacts (sinc in spatial domain)
Butterworth LPF (BLPF)	$1 / [1 + (D/D_0)^{2n}]$	Smooth transition, order n ; ringing small for $n = 1-2$
Gaussian LPF (GLPF)	$e^{-(D^2/2D_0^2)}$	No ringing; Gaussian in both domains

2.3.2 High-Pass Filtering (Sharpening)

Attenuates low frequencies and keeps high ones → sharpens edges (the DC term is removed, so the result looks dark; add back to the original). $H_{hp}(u, v) = 1 - H_{lp}(u, v)$, giving Ideal, Butterworth $[1 / (1 + (D_0/D)^{2n})]$ and Gaussian $[1 - e^{-(D^2/2D_0^2)}]$ high-pass filters.

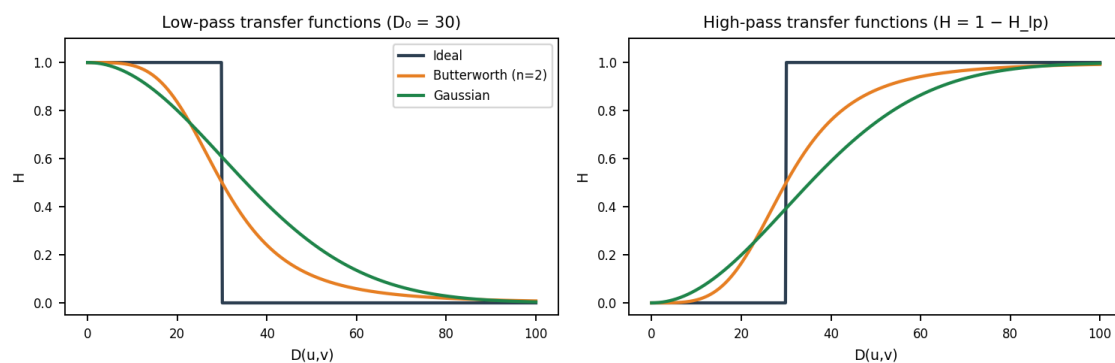


Figure 2.3 — Low-pass and high-pass transfer functions (cross-section)

High-frequency emphasis / high-boost: $H_{hfe} = a + b \cdot H_{hp}$, with $a \geq 0$ and $b > a$; keeps low-frequency content while boosting the edges.

Band-pass / band-reject: $H_{bp} = 1 - H_{br}$; a band-reject filter removes frequencies in a ring around D_0 (e.g., periodic noise).

2.3.3 Gaussian Filter

$H(u, v) = e^{(-D^2(u,v)/2\sigma^2)}$ with $\sigma = D_0$. Its spatial-domain counterpart $h(x, y) = 2\pi\sigma^2 \cdot e^{(-2\pi^2\sigma^2(x^2 + y^2))}$ is also Gaussian, so there is **no ringing**. A narrower Gaussian in frequency (small σ) gives a wider kernel in space (more blur).

2.3.4 Homomorphic Filtering

Used to simultaneously **compress the dynamic range** (reduce illumination variation) and **enhance contrast** (reflectance). From $f = i \cdot r$ we cannot separate the components in the frequency domain directly, since the Fourier transform of a product is not the product of transforms. Take the logarithm:

$$z(x, y) = \ln f(x, y) = \ln i(x, y) + \ln r(x, y) \Rightarrow Z(u, v) = F_i(u, v) + F_r(u, v)$$

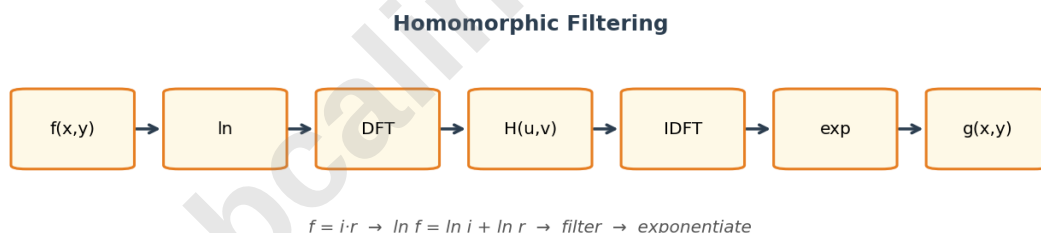


Figure 2.4 — Homomorphic filtering pipeline

Illumination varies slowly (low frequencies); reflectance varies abruptly at object edges (high frequencies). A filter with $\gamma_L < 1$ and $\gamma_H > 1$ attenuates the low frequencies (illumination) and amplifies the high frequencies (reflectance):

$$H(u, v) = (\gamma_H - \gamma_L)[1 - e^{(-c \cdot D^2/D_0^2)}] + \gamma_L, \quad c \text{ controls the steepness of the slope.}$$

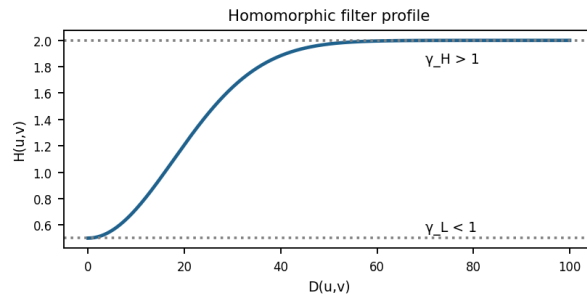


Figure 2.5 — Homomorphic filter transfer function ($\gamma_L = 0.5$, $\gamma_H = 2$)

Finally, $g(x, y) = \exp[s(x, y)]$ where s is the inverse DFT of $H \cdot Z$. Applications: images with uneven lighting, shadow-affected scenes, X-ray enhancement.

2.4 Spatial-Domain Enhancement: Point Processing

Point processing uses a 1×1 neighbourhood: $s = T(r)$, where r is the input gray level and s the output. T is a *gray-level transformation function*.

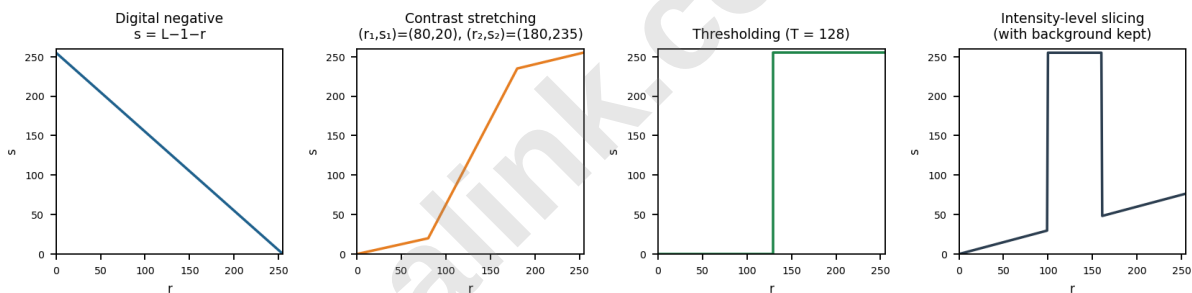


Figure 2.6 — Common point-processing transformation functions

(a) Digital Negative

$s = (L - 1) - r$. Reverses intensity: dark $\leftrightarrow$ bright. Useful for enhancing white or gray detail embedded in dark regions (e.g., mammograms).

(b) Contrast Stretching

Expands the range of gray levels in a low-contrast image to the full display range. Piece-wise linear with control points (r_1, s_1) and (r_2, s_2) : if $r_1 = s_1$ and $r_2 = s_2$, T is the identity; if $r_1 = r_2$, $s_1 = 0$, $s_2 = L - 1$, it is thresholding. **Min-max stretching:** $s = (r - r_{\min}) / (r_{\max} - r_{\min}) \times (L - 1)$.

Example 2.2 — Contrast stretching

A low-contrast 8-bit image has $r_{\min} = 80$ and $r_{\max} = 180$. Find the output for $r = 130$ using min-max stretching.

Solution

$s = (130 - 80)/(180 - 80) \times 255 = 0.5 \times 255 = 127.5 \approx \mathbf{128}$. Also $r = 80 \rightarrow 0$ and $r = 180 \rightarrow 255$.

(c) Clipping and Thresholding

- **Clipping:** a special contrast stretch with zero slope outside $[a, b]$: $s = 0$ for $r < a$, $s = \beta(r - a)$ for $a \leq r \leq b$, $s = \beta(b - a)$ for $r > b$. Reduces noise outside the band of interest.
- **Thresholding:** $s = 0$ if $r \leq T$, $s = L - 1$ if $r > T$. Produces a binary image (see segmentation, Unit 5).

(d) Intensity-Level Slicing

Highlights a specific range $[A, B]$ of gray levels (e.g., masses of water in satellite images, flaws in X-rays). **Without background:** $s = L - 1$ for $A \leq r \leq B$, else 0. **With background:** $s = L - 1$ inside the range, $s = r$ elsewhere (rest of the image is preserved).

(e) Log, Power-Law (Gamma) and Bit-Plane Slicing

- **Log:** $s = c \cdot \log(1 + r)$ — expands dark values, compresses bright ones (displaying Fourier spectra).
- **Power-law:** $s = c \cdot r^\gamma$. $\gamma < 1$ brightens (like log), $\gamma > 1$ darkens. Corrects display devices (gamma correction).
- **Bit-plane slicing:** separates the image into 8 binary planes; high-order bits hold most visual information, low-order bits are mostly noise.

2.5 Histogram Processing

The **histogram** of a digital image with gray levels in $[0, L - 1]$ is the discrete function $h(r_k) = n_k$, where n_k is the number of pixels with gray level r_k . The **normalised histogram** (an estimate of the probability density) is $p(r_k) = n_k / (M \cdot N)$.

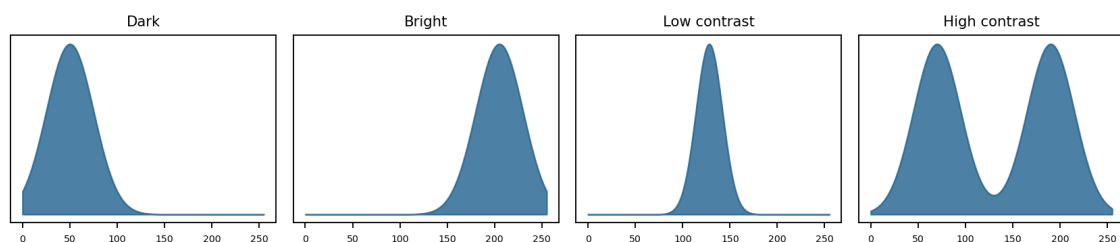


Figure 2.7 — Typical histograms: dark, bright, low-contrast and high-contrast images

- Dark image → histogram concentrated at the low end; bright image → at the high end.
- Low-contrast image → narrow histogram; high-contrast image → wide, spread-out histogram (good).

2.5.1 Histogram Equalization

Aim: transform an image so its histogram is approximately uniform, maximising contrast. The transformation is the cumulative distribution function (CDF):

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p(r_j) = (L - 1) \sum_{j=0}^k n_j / (MN), \text{ rounded to the nearest integer.}$$

T is monotonically increasing and maps $[0, L - 1]$ to $[0, L - 1]$. Equalization is automatic (no parameters), but being an approximation of a continuous process the result is only *approximately* flat for discrete images.

Example 2.3 — Histogram equalization

A 64×64 image (4096 pixels) has 3-bit gray levels ($L = 8$) with counts:

$r: 0, 1, 2, 3, 4, 5, 6, 7 \rightarrow n_k: 790, 1023, 850, 656, 329, 245, 122, 81$. Equalize the histogram.

Solution

Cumulative sums: 790, 1813, 2663, 3319, 3648, 3893, 4015, 4096.

$s_k = 7 \times \text{cum} / 4096$: 1.35, 3.10, 4.55, 5.67, 6.23, 6.65, 6.86, 7.00. Rounded: **1, 3, 5, 6, 6, 7, 7, 7**.

Mapping: $0 \rightarrow 1, 1 \rightarrow 3, 2 \rightarrow 5, 3 \rightarrow 6, 4 \rightarrow 6, 5 \rightarrow 7, 6 \rightarrow 7, 7 \rightarrow 7$.

New histogram: level 1: 790; 3: 1023; 5: 850; 6: $656 + 329 = 985$; 7: $245 + 122 + 81 = 448$ (total = 4096 ✓).

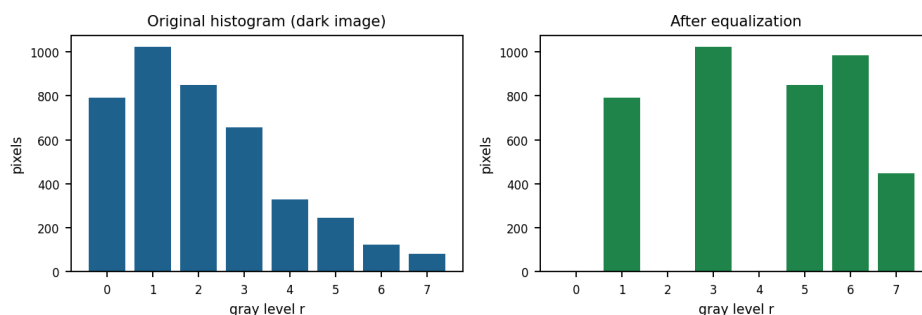


Figure 2.8 — Histogram before and after equalization (Example 2.3)

2.5.2 Histogram Modification / Specification (Matching)

Equalization cannot be controlled. In **histogram specification** a desired output histogram $p_z(z)$ is given:

- Equalize the input image: $s_k = T(r_k)$.
- Compute the transformation $G(z) = (L - 1) \sum p_z(z_i)$ of the desired histogram (equalization of the target).
- Map each s_k to the z value for which $G(z)$ is closest to s_k : $z = G^{-1}(s)$ — the combined mapping is $z = G^{-1}[T(r)]$.

Other modification: **local histogram processing** (equalize within an $m \times n$ sliding neighbourhood) to enhance local detail; **histogram stretching/sliding** (shift and expand the histogram).

2.6 Spatial Filtering (Neighbourhood Processing)

A **spatial filter** (mask, kernel, window) of size $m \times n$ (usually 3×3) is moved over the image; at every position the output pixel is computed from the mask coefficients $w(s, t)$ and the covered pixels:

$$g(x, y) = \sum_s \sum_t w(s, t) \cdot f(x + s, y + t) \text{ (correlation; convolution flips the mask)}$$

- **Linear** filters: weighted sums (averaging, Laplacian). **Non-linear** filters: ordering of pixels (median, min, max).
- Border handling: zero padding, replicating border pixels, reflection, or ignoring the border.

2.6.1 Smoothing (Low-Pass) Spatial Filters

Blur the image and reduce noise. **Mean (averaging) filter** — all coefficients equal, sum to 1:

Box 3×3			Weighted average 3×3		
1/9	1/9	1/9	1/16	2/16	1/16
1/9	1/9	1/9	2/16	4/16	2/16
1/9	1/9	1/9	1/16	2/16	1/16

The weighted (Gaussian-like) mask gives more importance to the centre pixel and blurs less. A larger mask means more blurring, more noise reduction but loss of edges and detail. Trade-off: noise vs. sharpness.

2.6.2 Median Filtering (Order-Statistic)

Replaces a pixel by the **median** of the values in its neighbourhood. Excellent for **salt-and-pepper (impulse) noise**; unlike averaging it preserves edges and does not create new gray values. Max filter finds the brightest points (reduces pepper noise); min filter finds the darkest (reduces salt noise).

Example 2.4 — Median vs. mean

A 3×3 window contains: 10, 15, 20, 200, 25, 30, 35, 40, 45 (the centre value 25, and 200 is an impulse noise pixel). Compute the mean and median.

Solution

Mean = $420 / 9 = 46.7$ — heavily affected by the outlier 200.

Sorted: 10, 15, 20, 25, 30, 35, 40, 45, 200 → median = 5th value = **30** — the outlier is ignored.

2.6.3 Sharpening (High-Pass) Spatial Filters

Sharpening highlights transitions in intensity — equivalent to spatial **differentiation**. The first derivative (gradient) gives thick edges; the second derivative (Laplacian) gives a strong response to fine detail, points and line ends (and a double response at ramp edges).

Laplacian: $\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$.

Laplacian (4-neighbour)			Laplacian (8-neighbour)		
0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

Sharpened image: $g(x, y) = f(x, y) - \nabla^2 f(x, y)$ when the centre coefficient is negative (use + when it is positive). The mask coefficients sum to zero, so flat areas become 0 (black).

Unsharp masking: $g = f - f$ (f = blurred image) gives the detail mask. **High-boost filtering:** $g = A \cdot f - f = (A - 1) \cdot f + (f - f)$, $A \geq 1$. If $A = 1$ it is standard high-pass; larger A retains more of the original image.

Gradient (first-derivative) sharpening: $\nabla f \approx |G_x| + |G_y|$ with Sobel masks (explained in Unit 5).

2.6.4 Spatial Low-, High- and Band-Pass Filters

Type	Mask property	Effect
Low-pass	All coefficients positive; sum = 1	Smooths, removes noise, blurs edges
High-pass	Centre positive, neighbours negative; sum = 0	Emphasises edges and fine details
Band-pass	Difference of two low-pass filters (DoG): $LP(\sigma_1) - LP(\sigma_2)$	Keeps a middle band of frequencies; detects blobs/edges at a given scale

2.7 Spatial vs. Frequency Domain — Comparison

Aspect	Spatial domain	Frequency domain
Operates on	Pixels directly	Transform coefficients
Cost	Cheap for small masks	Cheaper for large masks (FFT)
Intuition	Direct, simple	Clear idea of which frequencies are removed
Link	Convolution $f * h$	Multiplication $F \cdot H$ (convolution theorem)

Key Takeaways

- ✓ Enhancement is application-specific; spatial = direct pixel operations, frequency = modify the transform.
- ✓ DFT: $F(u,v) = \sum \sum f(x,y) e^{-j2\pi(ux/M+vy/N)}$; FFT reduces cost from N^2 to $N \log_2 N$; DCT is real and compacts energy (used in JPEG).
- ✓ Frequency filtering: $G = H \cdot F$; ILPF rings, BLPF is smooth, GLPF has no ringing; HPF = $1 - \text{LPF}$.
- ✓ Homomorphic: $\ln \rightarrow \text{DFT} \rightarrow H$ ($\gamma_L < 1 < \gamma_H$) $\rightarrow \text{IDFT} \rightarrow \exp$ — fixes illumination and enhances reflectance.
- ✓ Point processing: negative, contrast stretch, clipping, threshold, slicing, log/gamma.
- ✓ Histogram equalization: $s_k = (L - 1) \cdot \text{CDF}(r_k)$; specification matches a chosen histogram.
- ✓ Smoothing (mean) vs. median (best for salt-and-pepper); sharpening via Laplacian, unsharp mask, high-boost.

Reminder: these notes may contain errors — verify with your textbook and lecturer.

— End of Unit 2 —