Digital Image Processing

Imperial College London

Liyang DOU

Contents

Introduction

Digitalisation = Sampling + Quantisation

- **Sampling** → Discretise spatial coordinates (x, y) → affects spatial resolution
- **Quantisation** → Discretise signal amplitude (intensity) ⇒ e.g., 256 grey levels: 0 (black) → 255 (white)

Chapter Flow and Purpose

1. **Image Transforms:** Spatial domain → Other domain (e.g., frequency) ⇒ Insight extraction, compact representation
2. **Image Enhancement:** Improve visual quality ⇒ Highlight features (e.g., edges, contrast)
3. **Image Restoration:** Corrupted → Clean image ⇒ Reverse degradation (e.g., blur, noise, defocus) *Similar goal to enhancement, but different approach*
4. **Image Compression:** Reduce data volume ⇒ Efficient storage/transmission ⇒ Compress → Channel → Decompress

Part 1: Image Transforms

1.1 Basics

1.1.1 Orthogonal / Unitary Transform

Comparison

Orthogonal Matrix (Q)	**Unitary Matrix (U)**
Real-valued	Complex-valued
$Q^T = Q^{-1}$	$U^H = U^{-1}$
Examples: DCT, KLT	Example: DFT

All orthogonal matrices ⇒ unitary matrices, but not vice versa

Key Properties

- Norm-preserving: $\|T \cdot \underline{x}\| = \|\underline{x}\|$
- Eigenvalues have magnitude 1: $|\lambda| = 1$
- Rows and columns of T: form orthonormal/unitary basis in $\mathbb{R}^N$ or $\mathbb{C}^N$
- Correspond to energy-preserving, reversible linear transforms

A signal originally expressed as a combination of delta bases (in time or space domain): $f = \sum_{x=0}^{N-1} f(x) \cdot \delta_x \longrightarrow g = \sum_{u=0}^{N-1} g(u) \cdot \phi_u$ is now re-expressed using orthogonal (e.g., Fourier) basis functions.

1.1.2 Generic Form of Transforms

1D and 2D Calculation

1D Forward Transform	**1D Inverse Transform**
$g(u) = \sum_{x=0}^{N-1} T(u,x) f(x), \quad 0 \le u \le N-1$	$f(x) = \sum_{u=0}^{N-1} I(u,x) g(u), \quad 0 \le x \le N-1$
$\underline{g} = T \cdot \underline{f}$	$\underline{f} = I \cdot \underline{g}$

- $T(u,x)$: forward transformation kernel; $I(u,x)$: inverse transformation kernel
- Each row of T and I: one basis function of the target domain
- Output $g(u)$ / $f(x)$: depends on all values of $f(x)$ / $g(u)$
- $\underline{f} = I \cdot \underline{g} = I \cdot T \cdot \underline{f} \Rightarrow I = T^{-1}$; for orthogonal/unitary transforms: $T^{-1} = T^T$ or T^H, simplifies inverse computation

2D Forward Transform	2D Inverse Transform
$g(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} T(u,x,v,y) f(x,y)$	$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} I(u,x,v,y) g(u,v)$

- **Separable**: $T(u,x,v,y) = T_1(u,x) \cdot T_2(v,y)$, inverse defined similarly
- **Symmetric**: $T_1(u,x) = T_2(v,y) = T(x,y)$, inverse defined similarly

If both hold, then it can be written as: $g(u,v) = \sum_{x=0}^{M-1} T(u,x) \sum_{y=0}^{N-1} T(v,y) f(x,y)$. Computation:

- Fix x, vary $y \rightarrow$ row-wise 1D transform $\rightarrow$ intermediate sequence of each row $F(x,v)$
- Then fix v, vary $x \rightarrow$ column-wise 1D transform $\rightarrow$ final $g(u,v)$
- Row-column order is interchangeable; inverse follows the same logic

Two crucial properties

1. **Energy preservation (Parseval's theorem):**

$$\|\underline{g}\|^2 = \|\underline{f}\|^2 \quad \Rightarrow \quad \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |f(x,y)|^2 = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} |g(u,v)|^2$$

2. **Energy compaction:** Transformed energy is concentrated in a few values near the origin $\rightarrow$ reflects that natural images are smooth $\rightarrow$ dominated by low-frequency components

1.2 Discrete Fourier Transform (DFT)

Definition

	Forward Transform	Inverse Transform
1D	$F(u) = \sum_{x=0}^{N-1} f(x)\, e^{-j\frac{2\pi ux}{N}}$	$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} e^{j\frac{2\pi ux}{N}} F(u)$
2D	$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$	$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$

- 1D forward kernel: $T(u,x) = e^{-j\frac{2\pi ux}{N}}$ and inverse: $I(u,x) = \frac{1}{N} e^{j\frac{2\pi ux}{N}}$
- $T^{-1} = \frac{1}{N} T^*$, since rows/columns of T are orthogonal but not orthonormal
- 2D Kernel: $e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} = e^{-j2\pi\frac{ux}{M}} \cdot e^{-j2\pi\frac{vy}{N}} \rightarrow$ 2D DFT is symmetric and separable $\rightarrow$ Can be split into two independent 1D DFTs: first row-wise, then column-wise (or vice versa)

Spectrum Definitions Suppose $R(u,v)$, $I(u,v)$ are real and imaginary parts of $F(u,v)$, then:

- **Amplitude / Magnitude spectrum:** $|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)}$; typically shown as $D(u,v) = c\log(1+|F(u,v)|)$ for better visual contrast; reflects the **strength** of each frequency
- **Phase spectrum:** $\phi(u,v) = \tan^{-1}\left(\frac{I(u,v)}{R(u,v)}\right)$; reflects the **spatial positioning** of each frequency
- **Power spectrum:** $P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$

Properties

- **Periodicity:** $F(u,v) = F(u+M,v) = F(u,v+N) = F(u+M,v+N)$
- **Conjugate symmetry:** $F(u,v) = F^*(-u+pM,-v+qN), \forall\, p,q \in \mathbb{Z}$; implies $|F(u,v)| = |F(-u,-v)|$
  - $f(x,y)$ is real and even (symmetric) $\rightarrow$ $F(u,v)$ is real and even
  - $f(x,y)$ is real and odd (antisymmetric) $\rightarrow$ $F(u,v)$ is imaginary and odd
- **Linearity:** $\mathcal{F}\{af(x,y) + bg(x,y)\} = a\mathcal{F}\{f(x,y)\} + b\mathcal{F}\{g(x,y)\}$
- **Translation (can be used in visualisation):**
  - Spatial shift $\rightarrow$ frequency modulation: $f(x-x_0, y-y_0) \leftrightarrow F(u,v)e^{-j2\pi\left(\frac{ux_0}{M}+\frac{vy_0}{N}\right)}$
  - Frequency shift $\rightarrow$ spatial modulation: $f(x,y)e^{j2\pi\left(\frac{u_0x}{M}+\frac{v_0y}{N}\right)} \leftrightarrow F(u-u_0, v-v_0)$
- **Average value (DC component):** $\bar{f}(x,y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \Rightarrow F(0,0) = MN\bar{f}(x,y)$
- **Rotation:** Rotating $f(x,y)$ by $\theta \rightarrow$ rotates $F(u,v)$ by θ

1.3 Discrete Cosine Transform (DCT)

Definition DCT is a family of transforms with various definitions. We focus on only one here.

1D	$C(u) = a(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$	$f(x) = \sum_{u=0}^{N-1} a(u)C(u)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$
2D	$C(u,v) = a(u)a(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2M}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$	
	$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} a(u)a(v)C(u,v)\cos\left[\frac{(2x+1)u\pi}{2M}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$	

where $a(u) = \sqrt{\frac{1}{N}}$ when $u = 0$, and $a(u) = \sqrt{\frac{2}{N}}$ when $u = 1, \ldots, N-1$.

- 2D DCT is symmetric and separable → Can be split into two independent 1D DCTs: first row-wise, then column-wise (or vice versa)
- An image can be divided into patches (blocks) and apply DCT separately in each block (JPEG)

1.4 Walsh-Hadamard Transform

Signal: $f(x), 0 \le x \le N-1$, where $N = 2^n$ (size of the signal must be 2^n). $f(x) \to W(u)$ or $H(u)$
Representation: the variables x and u in n-digital binary form

$$(x)_{10} = (b_{n-1}(x)\ \ldots\ b_1(x)\, b_0(x))_2 \qquad (u)_{10} = (b_{n-1}(u)\ \ldots\ b_1(u)\, b_0(u))_2$$

Discrete Walsh Transform (DWT)

1D	$W(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_{n-1-i}(u)}$	$f(x) = \sum_{u=0}^{N-1} W(u)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_{n-1-i}(u)}$
	$\underline{W} = T \cdot \underline{f}$	$\underline{f} = N \cdot T \cdot \underline{W}$
2D	$W(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$	
	$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} W(u,v)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$ (idential to the forward)	

For 1D DWT:

- Transform kernel: $T(u,x) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1} b_i(x)b_{n-1-i}(u)}$. Each kernel value is either $\frac{1}{N}$ or $-\frac{1}{N}$, i.e., binary modulation
- Matrix T is real and symmetric with orthogonal columns and rows, $T^{-1} = N \cdot T = N \cdot T^T = I$, thus the inverse kernel $I(u,x) = N \cdot T(u,x) = (-1)^{\sum_{i=0}^{n-1} b_i(x)b_{n-1-i}(u)}$

For 2D DWT:

- Forward / inverse kernel: $\frac{1}{N}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$, symmetric and spearable, thus can be applied separately
- Square-wave ($\frac{1}{N}$ or $-\frac{1}{N}$) basis function, more computationally efficient than exponential basis

Discrete Hadamard Transform (DHT)

2D	$H(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]}$
	$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} H(u,v)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]}$ (identical to the forward)

- DHT differs from DWT only in the order of basis functions, with shared properties; The order of DHT basis does not support fast computation by using FFT
- Hadamard matrix supports recursive construction: $H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$ → enables fast generation of H_N from small base case

The concept of frequency Defined by number of sign changes (**sequency**) in basis vectors.

- Original DHT: sequency unordered; Original DWT: sequency roughly ordered
- Ordered DHT/DWT (Walsh-Hadamard Transform, WHT): rows rearranged by increasing sequency → become identical → unified as WHT → better energy compaction

1.5 Karhunen-Loève Transform (KLT)

Pipeline:

- Given n images as a population, denote $\underline{x}_{(k,l)} = \underline{x} = [x_1(k,l),\, x_2(k,l),\, \ldots,\, x_n(k,l)]^T$ as the vector of grey levels at the same position (k,l) across all images.
- Assume each image i is ergodic, so the spatial average approximates the statistical mean: $m_i = \mathbb{E}\{\underline{x}_{(k,l)}\} \approx \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} x_{i,(k,l)}$.
- Form the mean vector of the population $\underline{m}_{\underline{x}} = [m_1,\, m_2,\, \ldots,\, m_n]^T$.
- Compute the covariance matrix of each pixel position (k,l): $C_{\underline{x}} = \mathbb{E}\{(\underline{x} - \underline{m}_x)(\underline{x} - \underline{m}_x)^T\}$.
- Perform eigen-decomposition on $C_{\underline{x}}$ and form matrix A with rows as eigenvectors sorted by descending eigenvalues.
- Apply **KLT** and get a transformed population: $\underline{y} = A(\underline{x} - \underline{m}_x)$.
- Since $A^{-1} = A^T$, the **inverse KLT** gives: $\underline{x} = A^T\underline{y} + \underline{m}_{\underline{x}}$

Properties:

- Zero mean: $\underline{m}_{\underline{y}} = \mathbb{E}\{\underline{y}\} = 0$
- Diagonal covariance: $C_{\underline{y}} = AC_{\underline{x}}A^T = \mathrm{diag}(\lambda_1,\, \lambda_2,\, \ldots,\, \lambda_n)$
  - $C_{\underline{y}}$ is diagonal $\Rightarrow$ transformed vectors $\underline{y}$ are decorrelated
  - As image index increases, λ_i (principal components) decreases $\Rightarrow$ lower components carry negligible information $\Rightarrow$ enables compression
- Compression by cropping to top-K dimensions:
  - Form $\underline{y}_K$ by keeping top-K elements of $\underline{y}$
  - Use cropped matrix A_K of size $K \times n$ (rows: top-K eigenvectors)
  - Approximate reconstruction: $\hat{\underline{x}} = A_K^T\underline{y}_K + \underline{m}_{\underline{x}}$
  - MSE of reconstruction: $e_{MSE} = \|\underline{x} - \hat{\underline{x}}\|^2 = \sum_{j=K+1}^{n}\lambda_j$ (sum of discarded eigenvalues)
- Drawbacks of KLT:
  - Data-dependent: basis must be recomputed for each image
  - No fast algorithm: unlike FFT or DCT
  - Requires multiple realisations of each image
  - Perfect decorrelation not guaranteed (images are rarely truly ergodic)

Part 2: Image Enhancement

2.1 Basic Techniques Overview

Two categories:

- **Frequency domain methods**
- **Spatial domain methods:** local neighbourhood processing: $g(x,y) = T[f(x,y)]$ (mask/filter/kernel)

Point processing / intensity transformation (1×1 neighbourhood):

- $s = T(r)$, where r, s are grey levels of f, g at (x,y)
- Plot: can be visualised with r on the x-axis and s on the y-axis, range $[0, L-1]$

Examples:

- **Contrast manipulation:** stretch / enhance / threshold
- **Image negatives:** $s = L - 1 - r$ (invert)
- **Log transform:** $s = c\log(1+r)$ (boost low intensities)
- **Power-law:** $s = cr^\gamma$, c, γ: constants; Often called gamma correction
- **Grey-level slicing:** highlight intensity ranges
- **Bit-plane slicing:** analyse binary planes
- **Histogram processing:** redistribute intensities

2.2 Histogram Equalisation

Histogram Definition $h(r_k) = n_k$, where $r_k \in [0, L-1]$ is intensity and n_k is the number of pixels with that intensity. The normalised histogram is given by $p(r_k) = \frac{n_k}{MN}$, where $M \times N$ is the image size.

- Low contrast $\rightarrow$ histogram concentrated in narrow range
- High contrast $\rightarrow$ histogram spread across wide range
- Histogram contains no spatial information, different images may share identical histograms

Intensity transformation $s = T(r)$, where $r \in [0, L-1]$. Here, $r = 0$ denotes black and $r = L-1$ denotes white. The function $T(r)$ must satisfy the following conditions:

1. $T(r)$ is (strictly) monotonically increasing in $r \in [0, L-1] \Rightarrow$ guarantees output intensities preserve input order (avoids reversal). If strictly increasing, the mapping from s back to r is possible.
2. $0 \leq T(r) \leq L-1$ for $r \in [0, L-1] \Rightarrow$ ensures output range matches input range.

Histogram Equalisation Model intensities r and s as random variables. The transformation $s = T(r)$ is defined as the cumulative distribution function (CDF) of r. In histogram equalisation, the goal is to remap intensities such that the resulting histogram is uniformly distributed. May cause irreversible loss of detail, but visually more balanced.

	Continuous (ideal)	**Discrete (practical)**
Variables	$r,\ s$	$r_k,\ s_k$
PDF	$p_r(r),\ p_s(s)$	$p_r(r_k),\ p_s(s_k)$
Transform	$T(r) = (L-1)\int_0^r p_r(w)\,dw$	$T(r_k) = s_k = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$
Result	$p_s(s) = \frac{1}{L-1}$ (Uniform distribution)	$s_{L-1} = T(r_{L-1}) = L-1$ (Not flat but more stretched)

Extensions of Histogram Equalisation

- **Histogram specification:** given an image with intensity variable $r \sim p_r(r)$, and a target histogram $p_z(z)$, find a transformation $z = T(r)$ such that the output intensities follow $p_z(z)$.
  - Derivation:
$$\left.\begin{aligned} r \rightarrow s = T(r) = (L-1)\int_0^r p_r(w)\,dw \\ z \rightarrow s = G(z) = (L-1)\int_0^z p_z(w)\,dw \end{aligned}\right\} \quad \Rightarrow \quad z = G^{-1}(T(r))$$
  - Given $p_r(r)$ and $p_z(z)$, derive z via T, G, and G^{-1}
  - Special case: if $z \sim \text{Uniform}[0, L-1]$, then $s = G(z) = (L-1)\int_0^z \frac{1}{L-1}dw = z \Rightarrow s = z = T(r)$, i.e. reduces to histogram equalisation.
- **Local (spatially adaptive) histogram equalisation:** enhance details over small areas in an image by devising and applying transformation functions based on the intensity distribution in a **neighbourhood around every pixel**; Post-processing is required to remove the blocking artifacts

2.3 Spatial Filters

Image Averaging

1. Noisy samples $g_i(x,y) = f(x,y) + n_i(x,y)$, $i = 1, \ldots, L$ with white noise: $\mathbb{E}[n_i] = 0$, $\text{Var}(n_i) = \sigma_n^2$
2. Average image: $\bar{g}(x,y) = \frac{1}{L}\sum_{i=1}^{L} g_i(x,y) = f(x,y) + \frac{1}{L}\sum_{i=1}^{L} n_i(x,y) \rightarrow$ Mean of averaged noise: $\mathbb{E}[n] = 0$, variance: $\text{Var}(n) = \frac{1}{L}\sigma_n^2$ (averaging cancels out random noise, law of large numbers)

Spatial Filtering Definition Convolve image with spatial kernel.
Separable: 2D kernel = product of two 1D kernels.

Smoothing / Averaging (Low-pass Filtering)

1. **Box filter** (uniform): constant coefficients, normalised; needs padding (e.g. black border artefact).
2. **Gaussian filter** (symmetric, separable): $w(x,y) = Ke^{-\frac{x^2+y^2}{2\sigma^2}} \Rightarrow w(r) = Ke^{-\frac{r^2}{2\sigma^2}},\ r = \sqrt{x^2+y^2}$.
3. **Median filter** (non-linear): removes outliers; good for salt-and-pepper noise.

High-pass and High-boost Filtering

1. **High-pass:** central pos + surround neg or vice versa; sum = 0. Slow variation → 0, fast variation → strong edge.
2. **High-boost:** $f_{\text{boost}} = Af(x,y) - \text{low-pass}(f),\ A > 1$; i.e. original image plus enhanced details; partial recovery of low-frequency; used in publishing.

Edge Detection Edges = local intensity jumps (step, roof, ridge). Can be detected via gradient:

- Edge strength: $\|\nabla f\| = \sqrt{(\partial f/\partial x)^2 + (\partial f/\partial y)^2} \approx |\partial f/\partial x| + |\partial f/\partial y|$
- Edge orientation: $\theta = \tan^{-1}((\partial f/\partial y)/(\partial f/\partial x))$

Typical gradient-based **operators**:

1. **Roberts operator:** $\text{abs}\left\{\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\right\} + \text{abs}\left\{\begin{bmatrix}1 & -1\\ 0 & 0\end{bmatrix}\right\}$
2. **Prewitt operator:** $\text{abs}\left\{\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}\right\} + \text{abs}\left\{\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}\right\}$
3. **Sobel operator:** $\text{abs}\left\{\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\right\} + \text{abs}\left\{\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}\right\}$
4. **Laplacian (2nd order):** $\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$ or $\begin{bmatrix}0 & -1 & 0\\ -1 & 4 & -1\\ 0 & -1 & 0\end{bmatrix}$

Part 3: Image Restoration

Definition & Aim to remove/reduce distortions (e.g. noise, defocus, motion blur)

- Deterministic / stochastic:
  - deterministic: use pixel values directly
  - stochastic: use statistical properties
- Non-blind / semi-blind / blind: degradation process fully / partially / not known
- Direct / iterative: solution computed directly / progressively

3.1 System Model

Degradation model $y(i,j) = H[f(i,j)] + n(i,j)$, where (i,j) are spatial coordinates, $f(i,j)$ is the original image, $H[\cdot]$ is the degradation function (e.g. blur), and $n(i,j)$ is additive noise.

LSI properties

- Linearity: $H[k_1 f_1 + k_2 f_2] = k_1 H[f_1] + k_2 H[f_2]$
- Shift-invariance: $H[f(i-i_0, j-j_0)] = y(i-i_0, j-j_0)$

LSI degradation model (convolution)

$$y(i,j) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f(k,l)h(i-k,j-l) + n(i,j) = f(i,j) * *h(i,j) + n(i,j)$$

Task: Given observation y, estimate original image $\hat{f}$. (The degradation model H, noise statistics and some prior knowledge of f is known)

Several metrics

- **BSNR** (Blurred Signal-to-Noise Ratio): objective degradation metric reflecting the severity of additive noise in relation to the blurred image.
  $\text{BSNR} = 10\log_{10}\left(\frac{\frac{1}{MN}\sum_i\sum_j[z(i,j)-\bar{z}(i,j)]^2}{\sigma_n^2}\right)$ where $z(i,j) = y(i,j) - n(i,j)$ is the blurred noiseless image, $\bar{z}(i,j) = \mathbb{E}[z(i,j)]$, and σ_n^2 is noise variance.

- **ISNR** (Improvement in Signal-to-Noise Ratio): objective restoration metric reflecting the quality of the restored image. If positive, restoration improves the image; if negative, the image worsens.
  $\text{ISNR} = 10\log_{10}\left(\frac{\sum_i\sum_j[f(i,j)-y(i,j)]^2}{\sum_i\sum_j[f(i,j)-\hat{f}(i,j)]^2}\right)$

3.2 Implementation

Periodic Extension DFT assumes periodicity in both space and frequency → Extend all signals via zero-padding → treat as virtually periodic

- **1D case:**
  - $x[n]$: M samples, $h[n]$: N samples
  - Pad both to $M+N-1$ (with dummy zeros)
  - Linear convolution length → $2M+2N-3$, only first $M+N-1$ samples are valid (rest = 0) ⇒ Circular convolution of size $M+N-1$ equals linear convolution of original signals
- **2D case:**
  - Image $f(x,y)$: size $A\times B$; degradation $h(x,y)$: size $C\times D$
  - Extend both to size $M\times N$ with: $M\geq A+C-1$; $N\geq B+D-1$
  - Example: 256×256 image, 3×3 degradation ⇒ pad to 258×258
  - Assumed periodic after padding and get: $y_e(i,j)=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}f_e(k,l)\,h_e(i-k,j-l)+n_e(i,j)$

Lexicographic Ordering

Vectorised model: Stack image rows (lexicographic ordering) ⇒ Images $f(i,j), y(i,j)$ → vectors $\mathbf{f},\mathbf{y}$ (size $MN\times1$) ⇒ Model: $\mathbf{y}=\mathbf{Hf}+\mathbf{n}$

Block-circulant matrices: If degradation is LSI → $\mathbf{H}$ becomes block-circulant matrix (size $MN\times MN$)

$$\mathbf{H}=\begin{bmatrix}\mathbf{H}_0 & \mathbf{H}_{M-1} & \cdots & \mathbf{H}_1\\ \mathbf{H}_1 & \mathbf{H}_0 & \cdots & \mathbf{H}_2\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{H}_{M-1} & \mathbf{H}_{M-2} & \cdots & \mathbf{H}_0\end{bmatrix}$$

where each submatrix $\mathbf{H}_j$ is circulant, i.e.:

$$\mathbf{H}_j=\begin{bmatrix}h_e(j,0) & h_e(j,N-1) & \cdots & h_e(j,1)\\ h_e(j,1) & h_e(j,0) & \cdots & h_e(j,2)\\ \vdots & \vdots & \ddots & \vdots\\ h_e(j,N-1) & h_e(j,N-2) & \cdots & h_e(j,0)\end{bmatrix}$$

Transformation to frequency domain: Due to block-circulant property, the large system $\mathbf{y}=\mathbf{Hf}+\mathbf{n}$ can be transformed into MN scalar equations:

$$Y(u,v)=MN\cdot H(u,v)F(u,v)+N(u,v)\quad\text{for } u=0,\ldots,M-1,\ v=0,\ldots,N-1$$

3.3 Restoration methods

Inverse filtering: Solving the ideal equation without noise for $F(u,v)$ in frequency domain:

$$\hat{F}(u,v)=\frac{H^*(u,v)(Y(u,v)-N(u,v))}{|H(u,v)|^2}\qquad\Rightarrow$$

$$\hat{f}(i,j)=\mathscr{F}^{-1}\left[\frac{H^*(u,v)(Y(u,v)-N(u,v))}{|H(u,v)|^2}\right]=\mathscr{F}^{-1}\left[\frac{(Y(u,v)-N(u,v))}{H(u,v)}\right]$$

Drawback: Inverse filtering fails when $H(u,v)$ is close to zero, causing instability or noise amplification.

Pseudoinverse / generalized inverse filtering: to solve the problem of noise amlification

$$\hat{F}(u,v) = \begin{cases} \frac{H^*(u,v)(Y(u,v)-N(u,v))}{|H(u,v)|^2}, & H(u,v) \neq 0 \\ 0, & H(u,v) = 0 \end{cases}$$

or

$$\hat{F}(u,v) = \begin{cases} \frac{H^*(u,v)(Y(u,v)-N(u,v))}{|H(u,v)|^2} = \frac{Y(u,v)-N(u,v)}{H(u,v)}, & |H(u,v)| \geq \varepsilon \\ 0, & \text{otherwise} \end{cases}$$

Constrained Least Squares (CLS) Restoration
To stabilise inverse filtering, CLS introduces a smoothness constraint by minimising

$$\min_{\mathbf{f}} \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2 \quad \text{subject to} \quad \|\mathbf{C}\mathbf{f}\|^2 < \varepsilon \quad \Rightarrow \quad \min_{\mathbf{f}} \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2 + \alpha\|\mathbf{C}\mathbf{f}\|^2,$$

where $\mathbf{C}$ is a high-pass filter and α is a Lagrange multiplier (regularisation parameter). The solution is

$$(\mathbf{H}^T\mathbf{H} + \alpha\mathbf{C}^T\mathbf{C})\mathbf{f} = \mathbf{H}^T\mathbf{y}.$$

In the frequency domain:

$$\hat{F}(u,v) = \frac{H^*(u,v)\,(Y(u,v) - N(u,v))}{|H(u,v)|^2 + \alpha|C(u,v)|^2}.$$

The denominator remains stable even when $|H(u,v)|$ is close to zero, thanks to the $\alpha|C(u,v)|^2$ term. Regularisation parameter α controls the trade-off between data fidelity $\|\mathbf{y}-\mathbf{H}\mathbf{f}\|^2$ and smoothness $\|\mathbf{C}\mathbf{f}\|^2$.

- **Small** α: emphasises fidelity; when $\alpha = 0$, CLS reduces to inverse filtering. The restored image may preserve sharpness but amplify noise.
- **Large** α: emphasises smoothness; suppresses noise effectively but may result in a blurry image.

Wiener Filter Estimator (Stochastic Regularisation) Minimise $\mathbb{E}\left[\|\mathbf{f} - \hat{\mathbf{f}}\|^2\right]$ given known second-order statistics $(\mathbf{R}_{ff}, \mathbf{R}_{nn})$. Solution is given by:

Spatial domain:

$$\hat{\mathbf{f}} = \mathbf{R}_{ff}\mathbf{H}^T\left(\mathbf{H}\mathbf{R}_{ff}\mathbf{H}^T + \mathbf{R}_{nn}\right)^{-1}\mathbf{y}$$

Frequency domain:

$$W(u,v) = \frac{S_{ff}(u,v)H^*(u,v)}{S_{ff}(u,v)|H(u,v)|^2 + S_{nn}(u,v)}, \quad \hat{F}(u,v) = W(u,v)Y(u,v)$$

- $S_{ff}(u,v) = |F(u,v)|^2$: power spectral density of $f(i,j)$
- $S_{nn}(u,v) = |N(u,v)|^2$: power spectral density of $n(i,j)$

The noise variance has to be known, otherwise it is estimated from the observed image.

Assume no blur: $H(u,v) = 1$

$$W(u,v) = \frac{S_{ff}(u,v)}{S_{ff}(u,v) + S_{nn}(u,v)} = \frac{\text{SNR}}{\text{SNR} + 1}$$

- $\text{SNR} \gg 1 \Rightarrow W(u,v) \approx 1$
- $\text{SNR} \ll 1 \Rightarrow W(u,v) \approx \text{SNR}$

Since SNR is high at low frequencies and low at high frequencies, $W(u,v)$ behaves like a low-pass (smoothing) filter.

Relation with Inverse Filtering

If $S_{nn}(u,v) = 0$, then Wiener filter reduces to inverse filter:

$$W(u,v) = \frac{1}{H(u,v)}$$

If $S_{nn}(u,v) \to 0$, then Wiener filter becomes pseudoinverse filter:

$$W(u,v) = \frac{S_{ff}(u,v)H^*(u,v)}{S_{ff}(u,v)|H(u,v)|^2 + S_{nn}(u,v)} \to \begin{cases} \frac{1}{H(u,v)}, & H(u,v) \neq 0 \\ 0, & H(u,v) = 0 \end{cases}$$

Part 4: Image Compression

4.1 Introduction

Redundancy arises from correlation:

- **Spatial correlation**: between neighbouring pixels
- **Spectral correlation**: between bands or sensors in the frequency domain
- **Temporal correlation**: across time frames (e.g. video)

A generic compression system:

- Encoder (source coder + channel coder) → transmission/storage → Decoder (channel decoder + source decoder)
- **Source coder** performs compression: **lossless / lossy** (perfect / approximate reconstruction)

There are four key dimensions of tradeoffs:

- **Signal Quality** Metrics: bit error probability, SNR, mean opinion score, etc.

$$\text{SNR} = 10 \log_{10} \left(\frac{\text{encoder input signal energy}}{\text{noise energy}} \right)$$

  But note: **high SNR** does not always mean perceptually better quality.
- **Coding Efficiency** Described by **compression ratio**:

$$c_r = \frac{\text{source coder input size}}{\text{source coder output size}}$$

- **Coding Delay** Time delay from encoding + decoding process.
- **Coder Complexity** Concerns memory usage, power consumption, operations per second.

4.2 Information Theory Basics

Discrete Memoryless Source (DMS) Simplest form of an information source defined by source alphabet $S = \{s_1, s_2, \ldots, s_n\}$ and probabilities $P = \{p_1, p_2, \ldots, p_n\}$.

- **Self-Information of symbol** s_i (ideal length): $I(s_i) = -\log_2 p_i$
- **Entropy** (average information per symbol) of DMS: $H(S) = -\sum p_i \log_2 p_i$
- **DMS extension** (group the original symbols into blocks of N symbols): $H(S^N) = N \cdot H(S)$
  The entropy increases with symbol length. The entropy per **original symbol** remains the same.

Noiseless Source Coding Theorem For a source with entropy $H(S)$, by coding N source symbols together, it is possible to construct a code such that the average code length per symbol l_{avg} satisfies

$$H(S) \leq l_{\text{avg}} < H(S) + \delta, \quad \text{for any } \delta > 0$$

As $N \to \infty$, $l_{\text{avg}} \to H(S)$, but this is impractical since the alphabet size grows exponentially with N.

- **Average length**: $l_{\text{avg}} = \sum l_i p_i$; Entropy is the theoretical minimum l_{avg} (coding limit)
- **Efficiency**: $\eta = \frac{H(S)}{l_{\text{avg}}}$, **Redundancy**: $l_{\text{avg}} - H(S)$
- **Prefix codes**: no codeword is a prefix (beginning) of another codeword; ensure unique decodability

Lossy Compression & Rate-Distortion Theory Allows controlled distortion to reduce bit rate. Given the level of image **loss (or distortion)**, there is always a bound on the minimum **bit rate** (bits per pixel) of the compressed bit stream.

- $R(D)$ curve: lower distortion → higher rate
- $R(0) = H$: theoretical minimum for discrete signal
- $R(0) = \infty$: rises without limit for continuous signal

4.3 Typical Compression Methods

4.3.1 Huffman Coding

Optimal prefix code for known p_i

- Frequent symbols → shorter codes
- Can be built via a binary tree by repeatedly merging the two least probable symbols.
- Performance bound: $H(S) \leq l_{\text{avg}} < H(S) + 1$
  - If $p_{\max} < 0.5$, then $l_{\text{avg}} \leq H(S) + p_{\max}$
  - If $p_{\max} \geq 0.5$, then $l_{\text{avg}} \leq H(S) + p_{\max} + 0.086$
  - If probabilities are powers of two: $H(S) = l_{\text{avg}}$ (the ideal limit)
  - For N-th extension: $H(S) \leq l_{\text{avg}} \leq H(S) + \frac{1}{N}$
- The complement of a Huffman code is also a valid Huffman code
- Extensions reduce redundancy; symbol grouping improves efficiency

	Bit-Serial Decoding	**Lookup-Table-Based Decoding**
Input rate	Fixed (bit-by-bit)	Variable (based on max codeword length L)
Output rate	Variable (depends on codeword length)	Fixed (one symbol per access)
Decoding method	Traverse Huffman tree until a leaf is reached	Direct lookup using L-bit table
Memory usage	Low	High (table size 2^L rows)
Speed	Slower (bitwise)	Faster (constant-time lookup)
Use case	Software or resource-constrained systems	Hardware, high-speed decoding

4.3.2 Differential Coding & Lossless JPEG

Predictive coding transforms the original intensity distribution into a residual distribution with lower entropy, by replacing each pixel with its differential. **Differential = actual value − predicted value**:

$$r = y - x, \quad \text{with } y = f(a, b, c)$$

where x is the pixel of interest, and a, b, c are neighbouring pixels (left, above, top-left).

Residual coding (Lossless JPEG)

- Each residual r is split into two parts:
  - **Category**: number of bits needed to represent $|r|$ in binary (Huffman-coded)
    - Category $0 \leftrightarrow r = 0$
    - Category c $(c \geq 1) \leftrightarrow r \in [-2^c + 1,\ -2^{c-1}] \cup [2^{c-1},\ 2^c - 1]$
  - **Magnitude**: actual value of r encoded in binary
    Most Significant Bit (MSB) = 1 if $r > 0$; otherwise one's complement
- **Codeword** = Huffman(category) + magnitude
- **Residual range**: $[-2^{15} + 1, 2^{15}]$

Efficiency reason:
residuals are small and centred around 0 → more low-category codes → shorter Huffman output.

4.3.3 Block-based Coding

- **Spatial-domain**: the pixels are grouped into blocks and then compressed in the spatial domain
- **Transform-domain**: the pixels are grouped into blocks and then transformed via DCT/DFT/DHT/KLT to another domain

4.3.4 DCT-Based Coding (Low Activity Region)

- Preprocessing: subtract 128 → zero-mean block
- Transform: apply DCT
- **Quantisation**: $z[k,l] = \text{round}\left(\frac{y[k,l]}{q[k,l]}\right)$, $q[k,l]$ from matrix Q (HVS sensitivity + bitrate)
- Encode quantised Z using entropy coder
- Decode: inverse quantisation $\hat{z}[k,l] = z[k,l]\,q[k,l]$ + IDCT + level shift + block recombination

4.3.5 JPEG Coding (High Activity Region)

Basic Idea Image block → DCT → frequency coefficients → split into DC and AC terms:

- **DC**: top-left DCT coefficient
- **AC**: remaining 63 coefficients

DC and AC handled separately due to statistical differences:

- **DC**: varies smoothly → differential coding across blocks (DC differential: $[-2047, 2048]$)
- **AC**: many zeros → run-length + Huffman coding within blocks (AC coefficient: $[-1023, 1024]$)

DC Coefficients (inter-block) The first DCT coefficient of each block is the DC coefficient. Differential coding is applied between adjacent blocks, encoded as a **(size, amplitude)** pair.

AC Coefficients (in-block) Other values (AC coefficients) are encoded as **(run/size, amplitude)**:

- **Run**: the number of consecutive zeros before a non-zero AC coefficient (maximum run is 15)
- **Size**: category of the non-zero value (same definition as DC)
- **Encoding**: run/size is Huffman-coded, amplitude is stored in binary

Special cases:

- If run-length exceeds 15, use **(15/0)** to represent 15 zeros; cascade if needed
- The last AC coefficient must be non-zero (otherwise cannot mark end of block)
- If all remaining coefficients are zeros, use special symbol **0/0** to indicate **EOB** (End of Block)

Ordering

- **Conventional**: raster scan (row-wise)
- **Zig-zag**: prioritises low frequencies, better compression efficiency