

UNIT 4

Image Coding and Compression

(9 Hours) — Image Processing (CAC404)

⚠ 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.

4.1 Need for Image Compression

Image compression reduces the amount of data required to represent a digital image by removing redundant information. Raw images are huge: a 1024×1024 , 24-bit colour image needs 3 MB; a 90-minute video at 640×480 , 24 bits, 30 frames/s needs ≈ 83 GB. Compression is essential for:

- **Storage** — more images on a disk, in archives and in databases (e.g., medical PACS).
- **Transmission** — reducing bandwidth and time on the Internet, mobile, satellite and broadcast links.
- **Processing** — less data to move and handle.

Compression ratio $C = n_1 / n_2$ (n_1 = bits in original, n_2 = bits in compressed). **Relative data redundancy** $R = 1 - 1/C$.

Example 4.1 — Compression ratio

A 256×256 8-bit image is compressed to 16,384 bytes. Find C and R.

Solution

$n_1 = 256 \times 256 = 65,536$ bytes. $C = 65,536 / 16,384 = 4$ (written 4:1). $R = 1 - 1/4 = 0.75 \rightarrow 75\%$ of the original data is redundant.

4.2 Types of Redundancy

- **Coding redundancy** — fixed-length codes (e.g., 8 bits for every gray level) when some levels are far more probable than others. Removed by **variable-length codes** (shorter codes for frequent levels): Huffman, arithmetic.

- **Interpixel (spatial / temporal) redundancy** — neighbouring pixels (and successive video frames) are highly correlated, so a pixel's value can be predicted from its neighbours. Removed by run-length coding, predictive coding, transforms.
- **Psychovisual redundancy** — information the eye does not notice (fine colour detail, high-frequency detail). Removing it is irreversible → **lossy** compression, done by quantization.

For a code with length $l(r_k)$ for gray level r_k : $L_{avg} = \sum l(r_k) \cdot p(r_k)$ bits/pixel; with fixed 8-bit code, $L_{avg} = 8$ and $C = 8 / L_{avg}$.

4.3 Image Compression Model

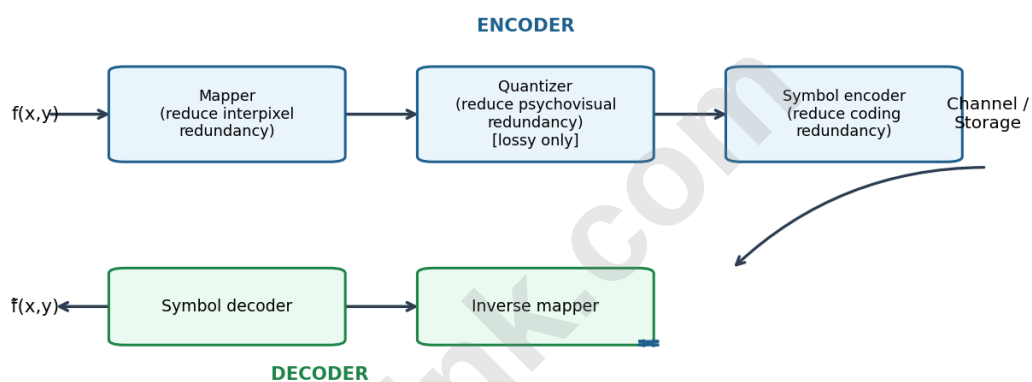


Figure 4.1 — General image compression system: encoder and decoder

- **Mapper** — converts the image into a (usually reversible) form that reduces interpixel redundancy: run-lengths, prediction errors, transform coefficients.
- **Quantizer** — reduces the accuracy of the mapper output according to a fidelity criterion; **irreversible**; omitted in lossless compression.
- **Symbol encoder** — assigns a (variable-length) code to the quantizer output; reduces coding redundancy; reversible.

Fidelity Criteria (for lossy compression)

- **Objective:** Root-mean-square error $e_{rms} = [(1/MN) \sum \sum (f(x,y) - \hat{f}(x,y))^2]^{1/2}$; Peak signal-to-noise ratio **PSNR** = $10 \cdot \log_{10} ((L-1)^2 / MSE)$ dB.
- **Subjective:** human observers rate the image (excellent, fine, passable, marginal, inferior, unusable).

Example 4.1b — PSNR

An 8-bit image reconstructed after compression has $MSE = 25$. Find the PSNR.

Solution

$\text{PSNR} = 10 \log_{10}(255^2 / 25) = 10 \log_{10}(65025/25) = 10 \log_{10}(2601) = 10 \times 3.415 = \mathbf{34.15 \text{ dB}}$.
(Above $\approx 30 \text{ dB}$ usually looks good.)

4.4 Elements of Information Theory

- **Self-information** of an event E with probability $P(E)$: $I(E) = -\log_2 P(E)$ bits. Rare events carry more information.
- **Entropy** (average information per symbol) of a source with symbol probabilities p_j : $H = -\sum p_j \log_2 p_j$ bits/symbol.
- **Shannon's first (noiseless coding) theorem**: the average code length can never be less than the entropy: $L_{\text{avg}} \geq H$. Entropy is the lower bound for lossless coding.
- **Coding efficiency** $\eta = H / L_{\text{avg}}$.
- **Channel capacity** — maximum rate at which information can be transmitted reliably over a channel; **rate-distortion theory** gives the minimum rate required for a given allowed distortion (lossy coding).

Maximum entropy occurs when all symbols are equally likely ($H = \log_2 N$ for N symbols, e.g., 8 bits for 256 equiprobable gray levels). Real images have entropy well below 8 bits/pixel, which is why they can be compressed.

4.5 Error-Free (Lossless) Compression

The reconstructed image is **identical** to the original ($\hat{f} = f$). Compression ratios are modest (typically 2:1 to 3:1). Required for medical, legal and archival images, text, executables.

4.5.1 Run-Length Coding (RLC)

Replaces a *run* of identical consecutive values by a (value, run length) pair. Effective for binary images, graphics and fax documents with long runs; it can *expand* data when runs are short (noisy images).

Example 4.2a — Run-length coding

Encode the pixel row W W W W B B B W W W W W W B B (W = white, B = black).

Solution

Runs: (W, 4), (B, 3), (W, 7), (B, 2). 16 pixels are represented by 4 pairs (8 numbers); in a binary image only the run lengths and the first colour are needed: first = W, runs = 4, 3, 7, 2.

Gray-level example: 5 5 5 5 7 7 9 $\rightarrow$ (5, 4), (7, 2), (9, 1).

4.5.2 Huffman Coding

The most popular technique for removing coding redundancy; yields the **smallest possible average code length** for symbol-by-symbol coding with known probabilities. The code is **prefix-free** (no code word is the prefix of another) and uniquely decodable instantaneously.

Procedure: (1) Sort symbols by probability. (2) Combine the two least probable symbols into one node whose probability is their sum. (3) Repeat until a single node (probability 1) remains — this builds a binary tree. (4) Label the two branches at each node 0 and 1. (5) The code of a symbol is the sequence of labels from the root to its leaf.

Example 4.2 — Huffman code

A source has six symbols with probabilities $a_1 = 0.4$, $a_2 = 0.3$, $a_3 = 0.1$, $a_4 = 0.1$, $a_5 = 0.06$, $a_6 = 0.04$. Find a Huffman code, L_{avg} , entropy and efficiency.

Solution

Merge steps: $(0.04 + 0.06) = 0.10$; then two 0.10s merge to 0.20; $0.20 + 0.10 = 0.30$; $0.30 + 0.30 = 0.60$; $0.60 + 0.40 = 1.0$.

One valid code (tie-breaking can change the individual words but not the lengths' sum): $a_1 \rightarrow 0$, $a_2 \rightarrow 10$, $a_3 \rightarrow 1110$, $a_4 \rightarrow 1111$, $a_5 \rightarrow 1101$, $a_6 \rightarrow 1100$.

$L_{\text{avg}} = 0.4 \times 1 + 0.3 \times 2 + 0.1 \times 4 + 0.1 \times 4 + 0.06 \times 4 + 0.04 \times 4 = \mathbf{2.20 \text{ bits/symbol}}$.

Entropy $H = -\sum p \log_2 p = \mathbf{2.14 \text{ bits/symbol}}$. Efficiency $\eta = 2.14/2.20 = \mathbf{97.4 \%}$.

A fixed-length code needs 3 bits/symbol, so $C = 3 / 2.2 = 1.36$.

Huffman tree for Example 4.2 (0 = upper/left branch, 1 = lower/right)

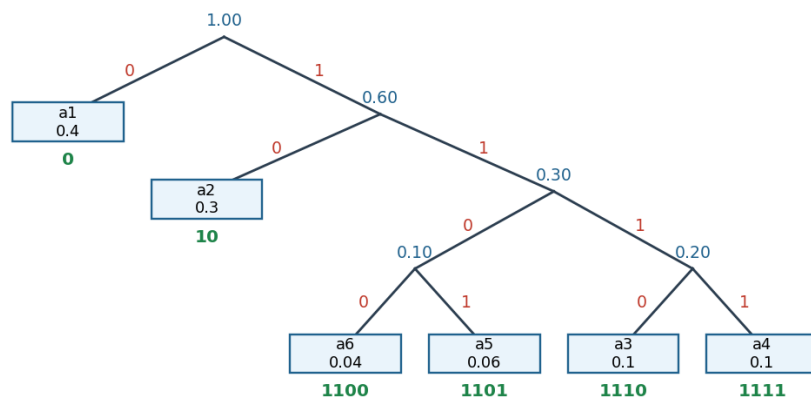


Figure 4.2 — Huffman tree for Example 4.2

4.5.3 Arithmetic Coding

Encodes an entire message as a single fractional number in $[0, 1)$. The interval is repeatedly narrowed according to the probability of each symbol; the final number identifies the message. It can use a *non-integer* number of bits per symbol, so it approaches the entropy more closely than Huffman (used in JPEG-2000, JBIG, H.264 CABAC).

4.5.4 LZW (Dictionary) Coding

Lempel–Ziv–Welch builds a dictionary of strings dynamically while encoding; repeated strings are replaced by their dictionary index. No probabilities are needed. Used in GIF, TIFF and PDF.

Example 4.3 — LZW

Encode ABABABA with an initial dictionary $A = 1$, $B = 2$.

Solution

Read A, then B: AB is not in the dictionary $\rightarrow$ output 1 (A), add $AB = 3$. Read A: BA is new $\rightarrow$ output 2 (B), add $BA = 4$. Read B: AB is found, so continue; read A: ABA is new $\rightarrow$ output 3 (AB), add $ABA = 5$. Read B: AB found; read A: ABA found; end of input $\rightarrow$ output 5 (ABA).
Output code: **1, 2, 3, 5** (4 codes for 7 symbols).

4.5.5 Predictive Coding (Lossless DPCM) and Bit-Plane Coding

- **Lossless predictive coding:** predict each pixel from previously coded neighbours, $\hat{f}_n = \text{round}(\sum \alpha_i f_{n-i})$; encode only the **prediction error** $e_n = f_n - \hat{f}_n$. Errors are small and peaked around zero $\rightarrow$ low entropy $\rightarrow$ Huffman coding is efficient. Example: 100, 102, 103, 103, 105 with $\hat{f}_n = f_{n-1}$ gives 100, 2, 1, 0, 2.

- **Bit-plane coding:** decompose the image into binary planes and compress each (e.g., with run-length / JBIG); Gray code reduces the number of transitions between planes.

4.6 Lossy Compression

Allows some loss of information ($f \neq \hat{f}$) to obtain much higher compression ratios (10:1 to 100:1 or more) with acceptable visual quality. The key step is **quantization**.

- **Lossy predictive coding (DPCM):** the prediction error is quantized before coding; the predictor uses *reconstructed* past values so encoder and decoder stay in step. Too coarse a quantizer produces slope overload and granular noise.
- **Transform coding:** divide the image into $n \times n$ blocks, apply a reversible linear transform (DCT, Walsh–Hadamard, KLT), quantize and encode the coefficients. Energy is packed into few coefficients; most small coefficients quantize to zero. Used in JPEG and MPEG.
- **Wavelet coding:** multi-resolution decomposition without blocking artefacts (Section 4.9).

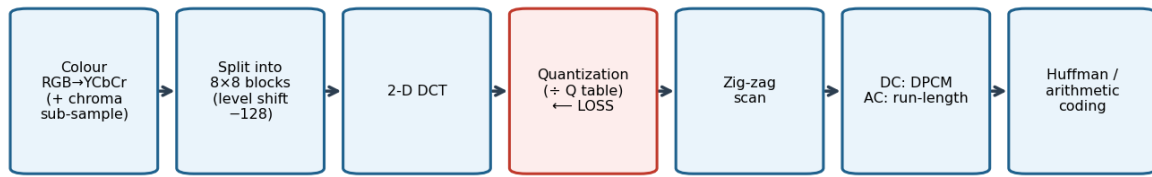
Aspect	Lossless	Lossy
Reconstruction	Exact	Approximate
Compression ratio	Low ($\approx 2-3 : 1$)	High (10 – 100 : 1)
Methods	RLC, Huffman, LZW, arithmetic	DCT/JPEG, wavelet, MPEG, vector quantization
Used for	Medical, text, archives	Photos, web, video, streaming

4.7 Image Compression Standards

Standard	Type	Application
CCITT Group 3 / Group 4	Lossless, 1-D / 2-D run-length + Huffman	Fax and bi-level document images
JBIG / JBIG2	Lossless (and lossy) arithmetic, bit-plane	Bi-level images
JPEG	Lossy (DCT); lossless mode available	Continuous-tone still images
JPEG 2000	Wavelet-based; lossless and lossy	Still images; medical, digital cinema
MPEG-1 / -2 / -4, H.261 / H.263 / H.264	Lossy, DCT + motion compensation	Video: VCD, DVD / TV, streaming

4.7.1 JPEG (Joint Photographic Experts Group)

JPEG baseline encoder



Decoder: entropy decode → de-quantize → inverse DCT → +128 → YCbCr→RGB

Figure 4.3 — Block diagram of the baseline JPEG compression

- **1. Colour transform:** RGB → YCbCr; Cb and Cr are often down-sampled by 2 (4:2:0), since the eye is less sensitive to colour detail.
- **2. Blocking and level shift:** divide into 8×8 blocks; subtract 128 from each pixel to centre values around 0.
- **3. DCT:** 2-D DCT on each block → 1 DC (average) and 63 AC coefficients.
- **4. Quantization:** divide each coefficient by the corresponding entry of a quantization table Q and round: $F_q(u, v) = \text{round}[F(u, v) / Q(u, v)]$. Larger Q for high frequencies → many zeros. This is the **only lossy step**; scaling Q controls the quality factor.
- **5. Zig-zag scan:** orders the 64 coefficients from low to high frequency, grouping zeros at the end.
- **6. Entropy coding:** DC coefficients are DPCM-coded (difference with previous block); AC coefficients are run-length coded as (zero-run, value) pairs with an end-of-block (EOB) symbol; then Huffman coded.

Standard luminance quantization table (quality 50):

0	1	2	3	4	5	6	7
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Example 4.4 — JPEG quantization

After the DCT of a block, the DC coefficient is $F(0,0) = -415.38$ and $F(0,1) = -30.19$. With $Q(0,0) = 16$ and $Q(0,1) = 11$, find the quantized values and the values recovered by the decoder.

Solution

$F_q(0,0) = \text{round}(-415.38/16) = \text{round}(-25.96) = -26$; decoder recovers $-26 \times 16 = -416$ (error 0.62).

$F_q(0,1) = \text{round}(-30.19/11) = \text{round}(-2.74) = -3$; decoder recovers $-3 \times 11 = -33$ (error 2.8).

Larger Q steps $\rightarrow$ larger error, but fewer bits.

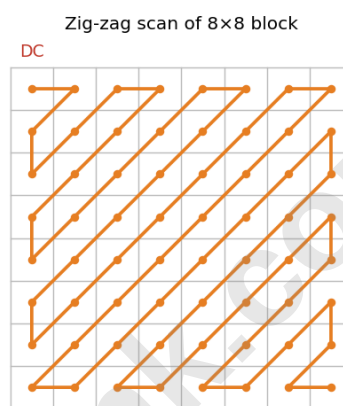
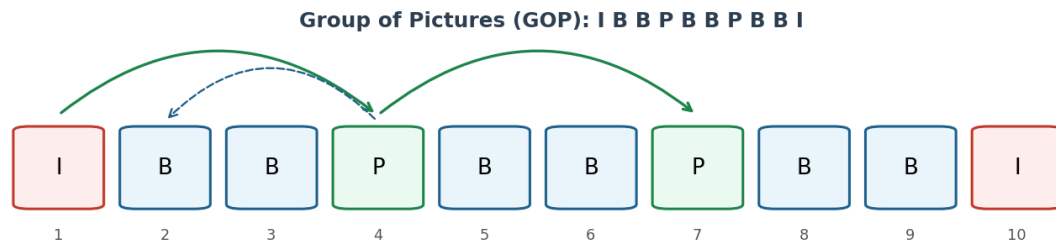


Figure 4.4 — Zig-zag ordering of the quantized coefficients

*Note: JPEG at high compression shows **blocking artefacts** (visible 8×8 squares) and **ringing/mosquito noise** near edges.*

4.7.2 MPEG (Moving Picture Experts Group)

Video has strong **temporal redundancy** between successive frames. MPEG exploits this with **motion-compensated prediction** plus DCT coding of the residual. Frames are divided into 16×16 **macroblocks**; for each, the encoder searches the reference frame for the best-matching block (**motion estimation**) and transmits only a **motion vector** and the prediction error (coded with DCT, as in JPEG).



Green arrows: forward prediction (P from previous I/P). Dashed: B frames use past AND future reference frames.

Figure 4.5 — MPEG group of pictures with I, P and B frames

Frame	Name	Coding
I	Intra-coded	Coded independently like a JPEG image; random-access points; largest size
P	Predicted	Predicted from the previous I or P frame using motion compensation; medium size
B	Bi-directional	Predicted from past and future I/P frames; highest compression; needs frame re-ordering

Standard	Typical use	Bit rate
MPEG-1	Video CD, MP3 audio (layer III)	≈ 1.5 Mbps
MPEG-2	Digital TV, DVD	4 – 15 Mbps
MPEG-4 (incl. H.264/AVC)	Internet streaming, mobile, Blu-ray; object-based coding	Wide range (kbps – Mbps)

4.8 Pixel Coding Methods — Summary

Method	Redundancy removed	Lossless / Lossy
Huffman, arithmetic	Coding	Lossless
Run-length, LZW	Interpixel (repetition)	Lossless
Predictive / DPCM	Interpixel (correlation)	Both
Transform (DCT) + quantization	Interpixel + psychovisual	Lossy
Wavelet	Interpixel + psychovisual	Both

4.9 Wavelet-Based Image Compression

A **wavelet transform** decomposes a signal into scaled and shifted versions of a short oscillating **mother wavelet**. Unlike the Fourier/DCT, wavelets are localised in **both space and frequency** — a multi-resolution representation (approximation + details at several scales).

In 2-D, a pair of low-pass (L) and high-pass (H) filters is applied along rows and then columns followed by down-sampling by 2, giving four subbands: **LL** (approximation), **LH**, **HL** (horizontal / vertical detail) and **HH** (diagonal). The LL band is decomposed again, producing a pyramid of subbands.

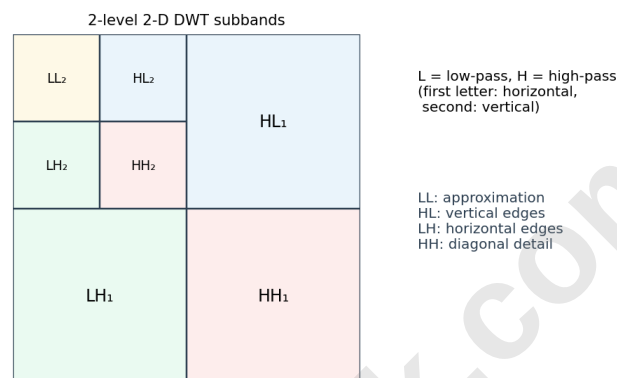


Figure 4.6 — Two-level 2-D wavelet decomposition (LL_2 holds most of the energy)

Example 4.5 — 1-D Haar wavelet

Decompose the signal {9, 7, 3, 5} with the Haar wavelet (average = $(a + b)/2$, detail = $(a - b)/2$).

Solution

Level 1: pairs (9, 7) → average 8, detail 1; (3, 5) → average 4, detail -1. Averages {8, 4}, details {1, -1}.

Level 2: (8, 4) → average 6, detail 2. Transform = {6, 2, 1, -1}.

Reconstruction: $6 + 2 = 8$, $6 - 2 = 4$; then $8 + 1 = 9$, $8 - 1 = 7$; $4 + (-1) = 3$, $4 - (-1) = 5$ ✓. Small detail values can be quantized to zero to compress.

Wavelet compression steps: (1) apply the DWT (several levels); (2) quantize the coefficients (a different step size for each subband; a dead-zone around 0); (3) entropy-code the result (e.g., EZW, SPIHT, EBCOT arithmetic coding in JPEG 2000).

- **Advantages over JPEG:** no blocking artefacts; better quality at high compression ratios; progressive (embedded) transmission with resolution and quality scalability; region-of-interest coding; both lossless and lossy modes in one codestream.

- **Drawbacks:** higher computational cost and memory use; less hardware support than JPEG.

Key Takeaways

- ✓ Compression removes coding, interpixel and psychovisual redundancy; $C = n_1/n_2$, $R = 1 - 1/C$.
- ✓ Entropy $H = -\sum p \log_2 p$ is the lower bound for lossless L_{avg} ; Huffman gives optimal prefix codes (Example: $L_{\text{avg}} = 2.2$ bits).
- ✓ Lossless: RLC, Huffman, arithmetic, LZW, lossless predictive. Lossy: quantization + transform / wavelet.
- ✓ JPEG: YCbCr $\rightarrow$ 8×8 DCT $\rightarrow$ quantize (lossy) $\rightarrow$ zig-zag $\rightarrow$ DPCM/RLC $\rightarrow$ Huffman.
- ✓ MPEG: I, P, B frames; motion estimation/compensation on 16×16 macroblocks plus DCT.
- ✓ Wavelet coding (JPEG 2000): multi-resolution subbands LL, LH, HL, HH; no blocking; progressive.
- ✓ Quality measures: MSE, PSNR = $10 \log_{10}((L-1)^2/\text{MSE})$, subjective ratings.

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

— End of Unit 4 —