Chapter 9

Image Compression Standards

9.1 The JPEG Standard

- JPEG is an image compression standard that was developed by the "Joint Photographic Experts Group". JPEG was formally accepted as an international standard in 1992.

- JPEG is a **lossy** image compression method. It employs a **transform coding** method using the DCT (*Discrete Cosine Transform*).

- An image is a function of i and j (or conventionally x and y) in the *spatial domain*.

  The 2D DCT is used as one step in JPEG in order to yield a frequency response which is a function $F(u, v)$ in the *spatial frequency domain*, indexed by two integers u and v.

Observations for JPEG Image Compression

- The effectiveness of the DCT transform coding method in JPEG relies on 3 major observations:

Observation 1: Useful image contents change relatively slowly across the image, i.e., it is unusual for intensity values to vary widely several times in a small area, for example, within an 8×8 image block.

- much of the information in an image is repeated, hence "spatial redundancy".

Observations for JPEG Image Compression (cont'd)

Observation 2: Psychophysical experiments suggest that humans are much less likely to notice the loss of very high spatial frequency components than the loss of lower frequency components.

- the spatial redundancy can be reduced by largely reducing the high spatial frequency contents.

Observation 3: Visual acuity (accuracy in distinguishing closely spaced lines) is much greater for gray ("black and white") than for color.

- chroma subsampling (4:2:0) is used in JPEG.

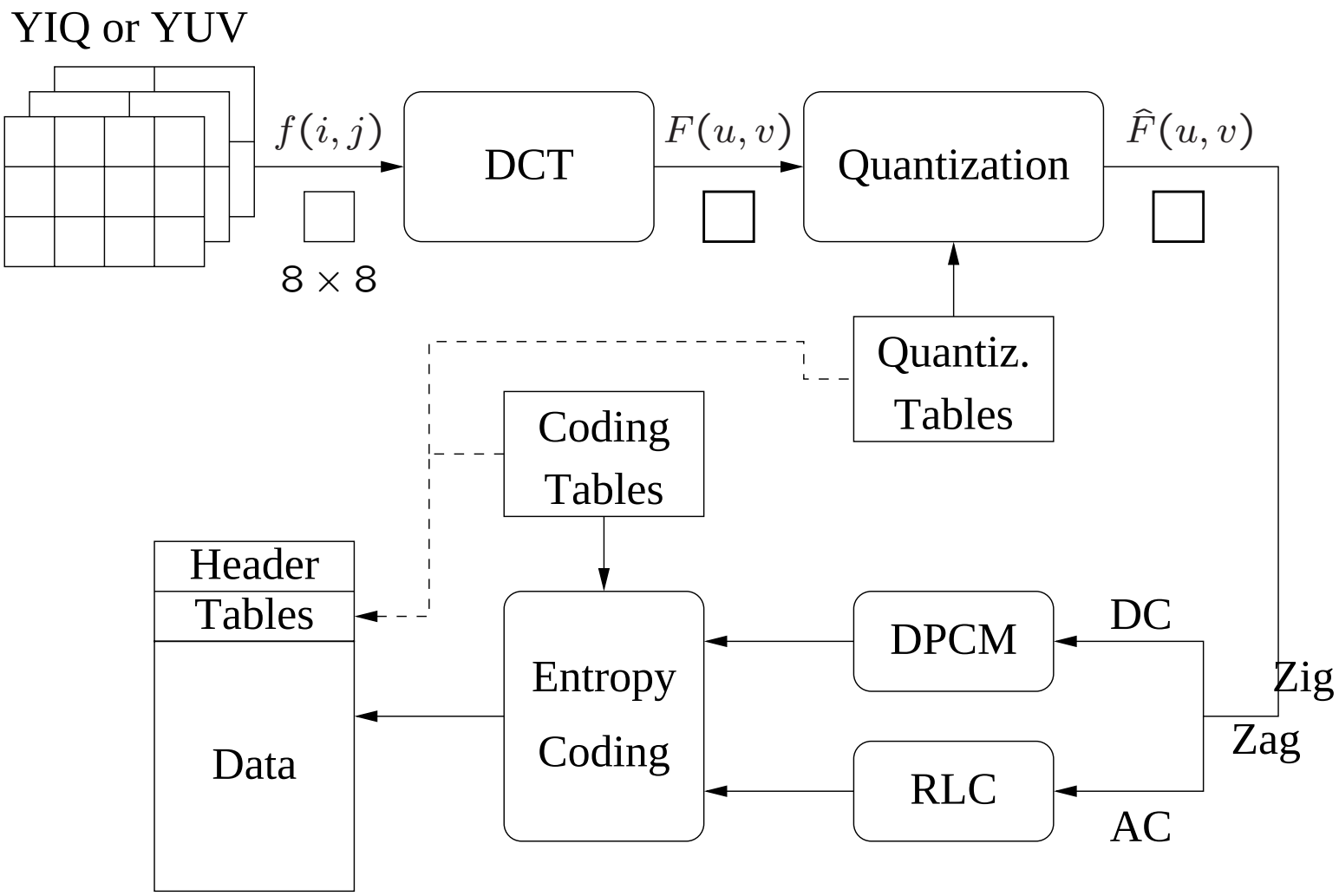


Fig. 9.1: Block diagram for JPEG encoder.

9.1.1 Main Steps in JPEG Image Compression

- Transform RGB to YIQ or YUV and subsample color.
- DCT on image blocks.
- Quantization.
- Zig-zag ordering and run-length encoding.
- Entropy coding.

DCT on image blocks

- Each image is divided into 8×8 blocks. The 2D DCT is applied to each block image $f(i, j)$, with output being the DCT coefficients $F(u, v)$ for each block.

- Using blocks, however, has the effect of isolating each block from its neighboring context. This is why JPEG images look choppy ("blocky") when a high *compression ratio* is specified by the user.

Quantization

$$\widehat{F}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{9.1}$$

- $F(u,v)$ represents a DCT coefficient, $Q(u,v)$ is a "quantization matrix" entry, and $\widehat{F}(u,v)$ represents the *quantized DCT coefficients* which JPEG will use in the succeeding entropy coding.
  - **The quantization step is the main source for loss in JPEG compression**.
  - The entries of $Q(u,v)$ tend to have larger values towards the lower right corner. This aims to introduce more loss at the higher spatial frequencies — a practice supported by Observations 1 and 2.
  - Table 9.1 and 9.2 show the default $Q(u,v)$ values obtained from psychophysical studies with the goal of maximizing the compression ratio while minimizing perceptual losses in JPEG images.

Table 9.1 The Luminance Quantization Table

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

Table 9.2 The Chrominance Quantization Table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

An 8×8 block from the Y image of 'Lena'

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

$f(i,j)$

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u,v)$

Fig. 9.2: JPEG compression for a smooth image block.

32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\widehat{F}(u,v)$

512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\tilde{F}(u,v)$

199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

$\tilde{f}(i,j)$

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

Fig. 9.2 (cont'd): JPEG compression for a smooth image block.

Another 8×8 block from the Y image of 'Lena'

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

$f(i,j)$

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

$F(u,v)$

Fig. 9.3: JPEG compression for a textured image block.

-5	-4	9	-5	2	1	1	0
-11	-5	-2	0	1	0	0	-1
3	-6	4	0	-3	-1	0	1
0	1	-1	0	1	0	0	0
0	0	-1	0	0	1	0	0
0	-1	1	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\widehat{F}(u,v)$

-80	-44	90	-80	48	40	51	0
-132	-60	-28	0	26	0	0	-55
42	-78	64	0	-120	-57	0	56
0	17	-22	0	51	0	0	0
0	0	-37	0	0	109	0	0
0	-35	55	-64	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\tilde{F}(u,v)$

70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

$\tilde{f}(i,j)$

0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

Fig. 9.3 (cont'd): JPEG compression for a textured image block.

Run-length Coding (RLC) on AC coefficients

- RLC aims to turn the $\widehat{F}(u,v)$ values into sets $\{$*#-zeros-to-skip , next non-zero value*$\}$.
- To make it most likely to hit a long run of zeros: a *zig-zag scan* is used to turn the 8×8 matrix $\widehat{F}(u,v)$ into a *64-vector*.

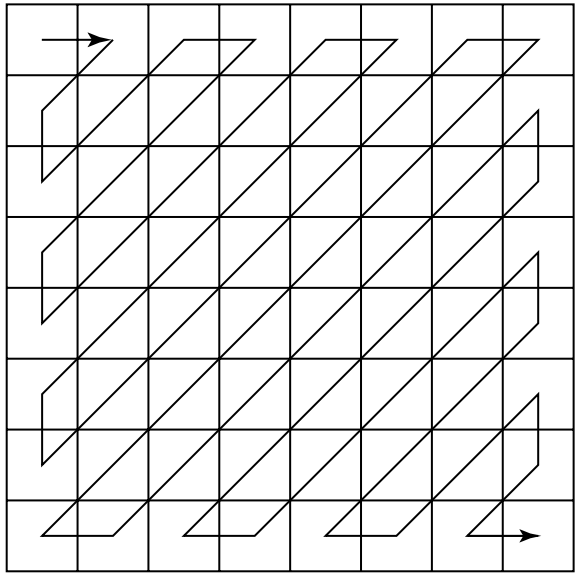


Fig. 9.4: Zig-Zag Scan in JPEG.

DPCM on DC coefficients

- The DC coefficients are coded separately from the AC ones. *Differential Pulse Code Modulation (DPCM)* is the coding method.

- If the DC coefficients for the first 5 image blocks are 150, 155, 149, 152, 144, then the DPCM would produce 150, 5, -6, 3, -8, assuming $d_i = DC_{i+1} - DC_i$, and $d_0 = DC_0$.

Entropy Coding

- The DC and AC coefficients finally undergo an entropy coding step to gain a possible further compression.
- Use DC as an example: each DPCM coded DC coefficient is represented by (SIZE, AMPLITUDE), where SIZE indicates how many bits are needed for representing the coefficient, and AMPLITUDE contains the actual bits.
- In the example we're using, codes 150, 5, -6, 3, -8 will be turned into

  (8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111) .

- SIZE is Huffman coded since smaller SIZEs occur much more often. AMPLITUDE is not Huffman coded, its value can change widely so Huffman coding has no appreciable benefit.

Table 9.3 Baseline entropy coding details – size category.

SIZE	AMPLITUDE
1	-1, 1
2	-3, -2, 2, 3
3	-7..-4, 4..7
4	-15..-8, 8..15
.	.
.	.
.	.
10	-1023..-512, 512..1023

9.1.2 Four Commonly Used JPEG Modes

- Sequential Mode — the default JPEG mode, implicitly assumed in the discussions so far. Each graylevel image or color image component is encoded in a single left-to-right, top-to-bottom scan.
- Progressive Mode.
- Hierarchical Mode.
- Lossless Mode — discussed in Chapter 7, to be replaced by JPEG-LS (Section 9.3).

Progressive Mode

Progressive JPEG delivers low quality versions of the image quickly, followed by higher quality passes.

1. **Spectral selection**: Takes advantage of the "spectral" (spatial frequency spectrum) characteristics of the DCT coefficients: higher AC components provide detail information.

   Scan 1: Encode DC and first few AC components, e.g., AC1, AC2.

   Scan 2: Encode a few more AC components, e.g., AC3, AC4, AC5.

   ⋮

   Scan k: Encode the last few ACs, e.g., AC61, AC62, AC63.

Progressive Mode (Cont'd)

2. **Successive approximation**: Instead of gradually encoding spectral bands, all DCT coefficients are encoded simultaneously but with their most significant bits (MSBs) first.

   Scan 1: Encode the first few MSBs, e.g., Bits 7, 6, 5, 4.

   Scan 2: Encode a few more less significant bits, e.g., Bit 3.

   ⋮

   Scan m: Encode the least significant bit (LSB), Bit 0.

Hierarchical Mode

- The encoded image at the lowest resolution is basically a compressed low-pass filtered image, whereas the images at successively higher resolutions provide additional details (differences from the lower resolution images).

- Similar to Progressive JPEG, the Hierarchical JPEG images can be transmitted in multiple passes progressively improving quality.

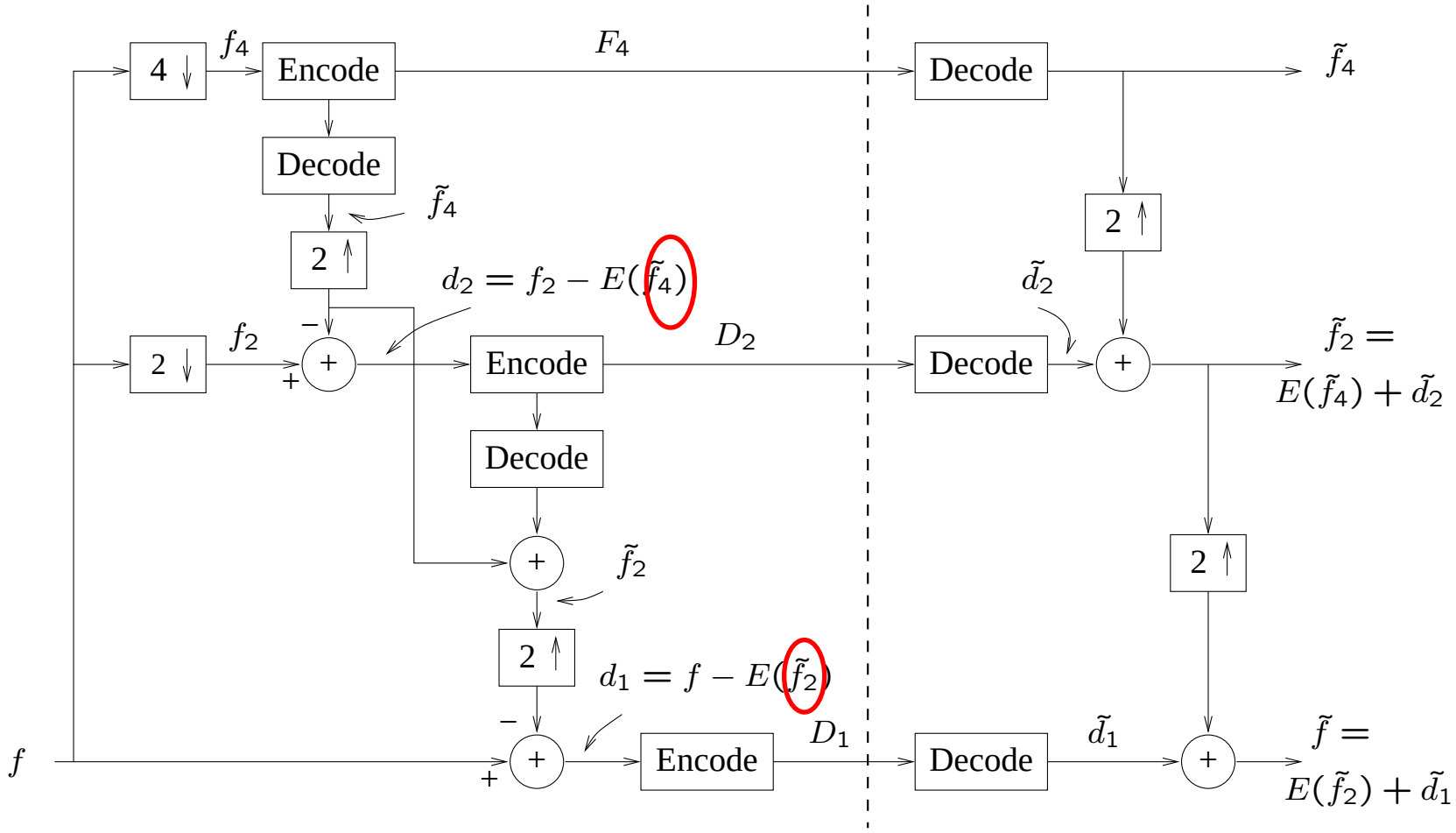


Fig. 9.5: Block diagram for Hierarchical JPEG.

Encoder for a Three-level Hierarchical JPEG

1. Reduction of image resolution:

   Reduce resolution of the input image f (e.g., 512×512) by a factor of 2 in each dimension to obtain f_2 (e.g., 256×256). Repeat this to obtain f_4 (e.g., 128×128).

2. Compress low-resolution image f_4:

   Encode f_4 using any other JPEG method (e.g., Sequential, Progressive) to obtain F_4.

3. Compress difference image d_2:

   (a) Decode F_4 to obtain $\tilde{f}_4$. Use any interpolation method to expand $\tilde{f}_4$ to be of the same resolution as f_2 and call it $E(\tilde{f}_4)$.

   (b) Encode difference $d_2 = f_2 - E(\tilde{f}_4)$ using any other JPEG method (e.g., Sequential, Progressive) to generate D_2.

4. Compress difference image d_1:

   (a) Decode D_2 to obtain $\tilde{d}_2$; add it to $E(\tilde{f}_4)$ to get $\tilde{f}_2 = E(\tilde{f}_4) + \tilde{d}_2$ which is a version of f_2 after compression and decompression.

   (b) Encode difference $d_1 = f - E(\tilde{f}_2)$ using any other JPEG method (e.g., Sequential, Progressive) to generate D_1.

Decoder for a Three-level Hierarchical JPEG

1. Decompress the encoded low-resolution image F_4:
   - Decode F_4 using the same JPEG method as in the encoder to obtain $\tilde{f}_4$.
2. Restore image $\tilde{f}_2$ at the intermediate resolution:
   - Use $E(\tilde{f}_4) + \tilde{d}_2$ to obtain $\tilde{f}_2$.
3. Restore image $\tilde{f}$ at the original resolution:
   - Use $E(\tilde{f}_2) + \tilde{d}_1$ to obtain $\tilde{f}$.

9.1.3 A Glance at the JPEG Bitstream

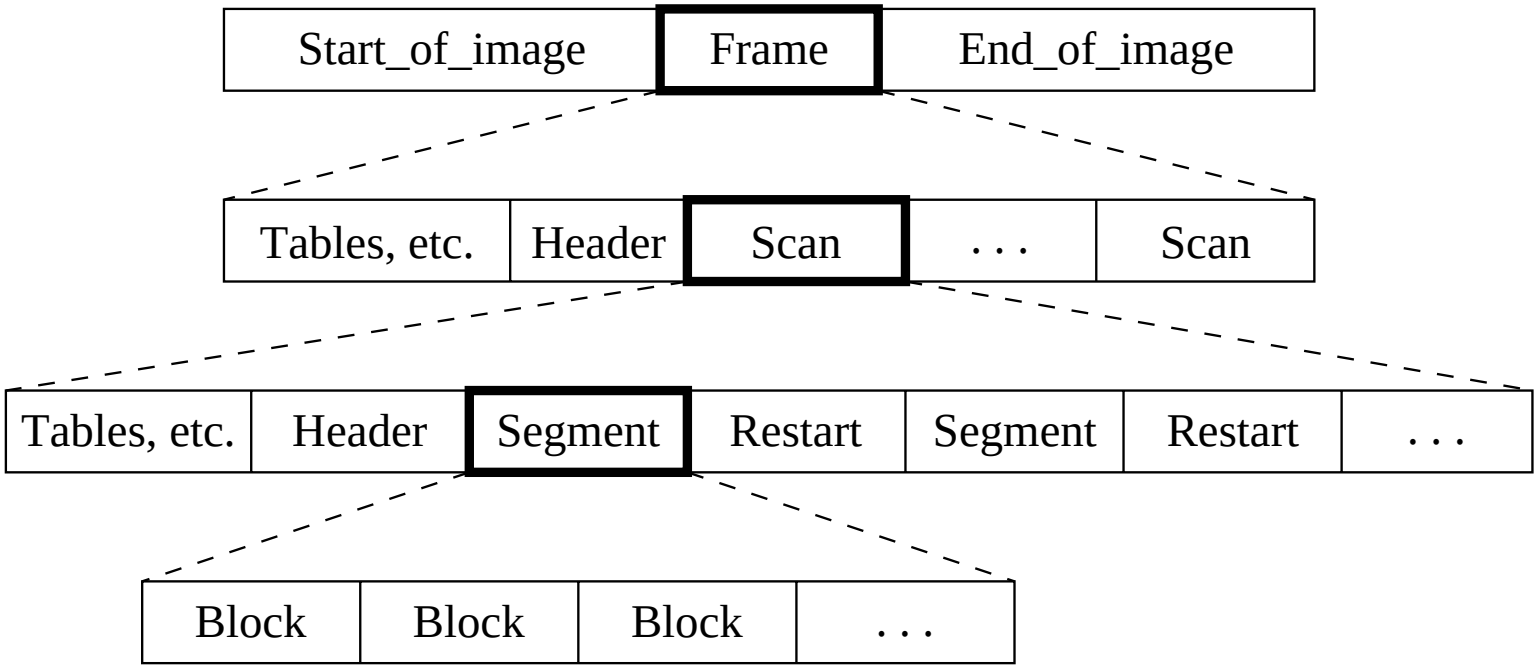


Fig. 9.6: JPEG bitstream.

9.2 The JPEG2000 Standard

- Design Goals:
  - To provide a better rate-distortion tradeoff and improved subjective image quality.
  - To provide additional functionalities lacking in the current JPEG standard.
- The JPEG2000 standard addresses the following problems:
  - **Lossless and Lossy Compression**: There is currently no standard that can provide superior lossless compression and lossy compression in a single bitstream.

- **Low Bit-rate Compression**: The current JPEG standard offers excellent rate-distortion performance in mid and high bit-rates. However, at bit-rates below 0.25 bpp, subjective distortion becomes unacceptable. This is important if we hope to receive images on our web-enabled ubiquitous devices, such as web-aware wristwatches and so on.
- **large Images**: The new standard will allow image resolutions greater than 64K by 64K without tiling. It can handle image size up to $2^{32} - 1$.
- **Single Decompression Architecture**: The current JPEG standard has 44 modes, many of which are application specific and not used by the majority of JPEG decoders.

- **Transmission in Noisy Environments**: The new standard will provide improved error resilience for transmission in noisy environments such as wireless networks and the Internet.
- **Progressive Transmission**: The new standard provides seamless quality and resolution scalability from low to high bit-rate. The target bit-rate and reconstruction resolution need not be known at the time of compression.
- **Region of Interest Coding**: The new standard allows the specification of Regions of Interest (ROI) which can be coded with superior quality than the rest of the image. One might like to code the face of a speaker with more quality than the surrounding furniture.

- **Computer Generated Imagery**: The current JPEG standard is optimized for natural imagery and does not perform well on computer generated imagery.
- **Compound Documents**: The new standard offers metadata mechanisms for incorporating additional non-image data as part of the file. This might be useful for including text along with imagery, as one important example.

- In addition, JPEG2000 is able to handle up to 256 channels of information whereas the current JPEG standard is only able to handle three color channels.

Properties of JPEG2000 Image Compression

- Uses Embedded Block Coding with Optimized Truncation (EBCOT) algorithm which partitions each subband LL, LH, HL, HH produced by the wavelet transform into small blocks called "code blocks".

- A separate scalable bitstream is generated for each code block $\Longrightarrow$ improved error resilience.

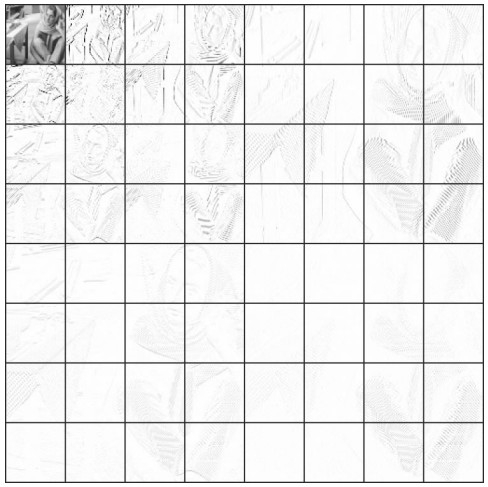


Fig. 9.7: Code block structure of EBCOT.

Main Steps of JPEG2000 Image Compression

- Embedded Block coding and bitstream generation.
- Post compression rate distortion (PCRD) optimization.
- Layer formation and representation.

Embedded Block Coding and Bitstream Generation

1. Bitplane coding.

2. Fractional bitplane coding.

1. Bitplane Coding

- Uniform dead zone quantizers are used with successively smaller interval sizes. Equivalent to coding each block one bitplane at a time.

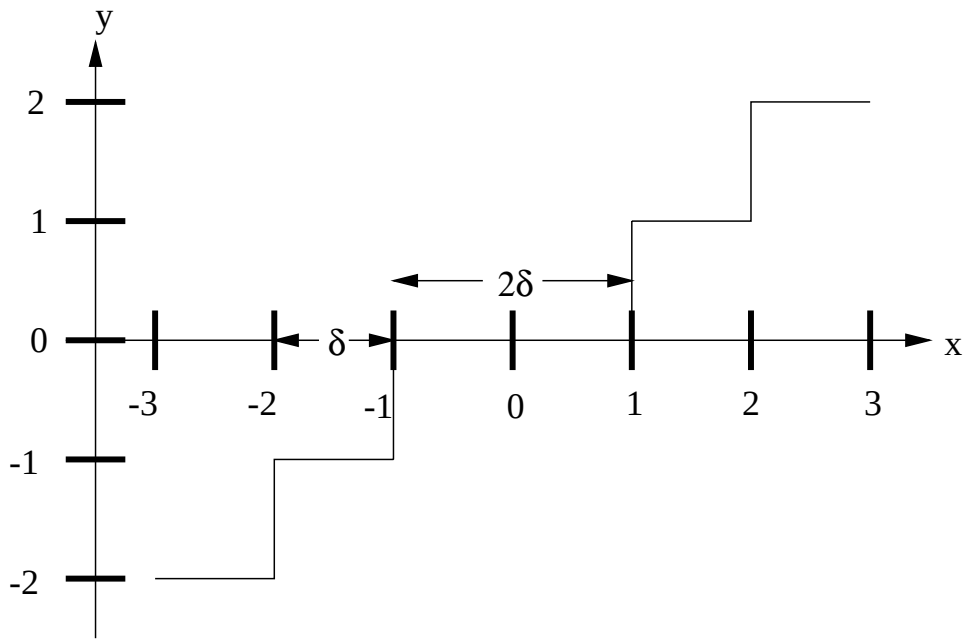


Fig. 9.8: Dead zone quantizer. The length of the dead zone is 2δ. Values inside the dead zone are quantized to 0.

- Blocks are further divided into a sequence of 16×16 sub-blocks.

- The significance of sub-blocks are encoded in a significance map σ^P where $\sigma^p(B_i[\mathbf{j}])$ denote the significance of sub-block $B_i[\mathbf{j}]$ at bitplane P.

- A quad-tree structure is used to identify the significance of sub-blocks one level at a time.

- The tree structure is constructed by identifying the sub-blocks with leaf nodes, i.e., $B_i^0[\mathbf{j}] = B_i[\mathbf{j}]$. The higher levels are built using recursion: $B_i^t[\mathbf{j}] = \cup_{\mathbf{z} \in \{0,1\}^2} B_i^{t-1}[2\mathbf{j} + \mathbf{z}]$, $0 \leq t \leq T$.

Bitplane Coding Primitives

Four different primitive coding methods that employ context based arithmetic coding are used:

- **Zero Coding**: Used to code coefficients on each bitplane that are not yet significant.
  - Horizontal: $h_i[\mathbf{k}] = \sum_{z \in \{1,-1\}} \sigma_i[k_1 + z, k_2]$, with $0 \leq h_i[\mathbf{k}] \leq 2$.
  - Vertical: $v_i[\mathbf{k}] = \sum_{z \in \{1,-1\}} \sigma_i[k_1, k_2 + z]$, with $0 \leq v_i[\mathbf{k}] \leq 2$.
  - Diagonal: $d_i[\mathbf{k}] = \sum_{z_1, z_2 \in \{1,-1\}} \sigma_i[k_1 + z_1, k_2 + z_2]$, with $0 \leq d_i[\mathbf{k}] \leq 4$.

Table 9.4 Context assignment for the zero coding primitive.

	LL, LH and HL subbands			HH subband	
Label	$h_i[\mathbf{k}]$	$v_i[\mathbf{k}]$	$d_i[\mathbf{k}]$	$d_i[\mathbf{k}]$	$h_i[\mathbf{k}] + v_i[\mathbf{k}]$
0	0	0	0	0	0
1	0	0	1	0	1
2	0	0	> 1	0	> 1
3	0	1	x	1	0
4	0	2	x	1	1
5	1	0	0	1	> 1
6	1	0	> 0	2	0
7	1	> 0	x	2	> 0
8	2	x	x	> 2	x

- **Run-length coding**: Code runs of 1-bit significance values. Four conditions must be met:
  - Four consecutive samples must be insignificant.
  - The samples must have insignificant neighbors.
  - The samples must be within the same sub-block.
  - The horizontal index k_1 of the first sample must be even.

- **Sign coding**: Invoked at most once when a coefficients goes from being insignificant to significant.
  - The sign bits $\chi_i[\mathbf{k}]$ from adjacent samples contains substantial dependencies.
  - The conditional distribution of $\chi_i[\mathbf{k}]$ is assumed to be the same as $-\chi_i[\mathbf{k}]$.
  - $\bar{h}_i[\mathbf{k}]$ be 0 if both horizontal neighbors are insignificant, 1 if at least one horizontal neighbor is positive, or -1 if at least one horizontal neighbor is negative.
  - $\bar{v}_i[\mathbf{k}]$ defined similarly for vertical neighbors.
  - If $\widehat{\chi}_i[\mathbf{k}]$ is the sign prediction, the binary symbol coded using the relevant context is $\chi_i[\mathbf{k}] \cdot \widehat{\chi}_i[\mathbf{k}]$.

Table 9.5 Context assignment for the sign coding primitive

Label	$\widehat{\chi}_i[\mathbf{k}]$	$\bar{h}_i[\mathbf{k}]$	$\bar{v}_i[\mathbf{k}]$
4	1	1	1
3	1	0	1
2	1	-1	1
1	-1	1	0
0	1	0	0
1	1	-1	0
2	-1	1	-1
3	-1	0	-1
4	-1	-1	-1

- **Magnitude refinement**: Code the value of $\nu_i^p[\mathbf{k}]$ given that $\nu_i[\mathbf{k}] \geq 2^{p+1}$.

  - $\tilde{\sigma}_i[\mathbf{k}]$ is changes from 0 to 1 after the magnitude refinement primitive is first applied to $s_i[\mathbf{k}]$.

  - $v_i^p[\mathbf{k}]$ is coded with context 0 if $\tilde{\sigma}[\mathbf{k}] = h_i[\mathbf{k}] = v_i[\mathbf{k}] = 0$, with context 1 if $\tilde{\sigma}_i[\mathbf{k}] = 0$ and $h_i[\mathbf{k}] + v_i[\mathbf{k}] \neq 0$, and with context 2 if $\tilde{\sigma}_i[\mathbf{k}] = 1$.

2. Fractional Bitplane Coding

- Divides code block samples into smaller subsets having different statistics.
- Codes one subset at a time starting with the subset expecting to have the largest reduction in distortion.
- Ensures each code block has a finely embedded bitstream.
- Four different passes are used: forward significance propagation pass ($\mathcal{P}_1^p$); reverse significance propagation pass ($\mathcal{P}_2^p$); magnitude refinement pass ($\mathcal{P}_3^p$); and normalization pass ($\mathcal{P}_4^p$).

Forward Significance Propagation Pass

- Sub-block samples are visited in scan-line order and insignificant samples and samples that do not satisfy the neighborhood requirement are skipped.
- For the LH, HL, and LL subbands, the neighborhood requirement is that at least one of the horizontal neighbors has to be significant.
- For the HH subband, the neighborhood requirement is that at least one of the four diagonal neighbors is significant.

Reverse Significance Propagation Pass

- This pass is identical to $\mathcal{P}_1^p$ except that it proceeds in the reverse order. The neighborhood requirement is relaxed to include samples that have at least one significant neighbor in any direction.

Magnitude Refinement Pass

- This pass encodes samples that are already significant but have not been coded in the previous two passes. Such samples are processed with the magnitude refinement primitive.

Magnitude Refinement Pass

- The value $\nu_i^p[\mathbf{k}]$ of all samples not considered in the previous three coding passes are coded using the sign coding and run-length coding primitives as appropriate. If a sample is found to be significant, its sign is immediately coded using the sign coding primitive.

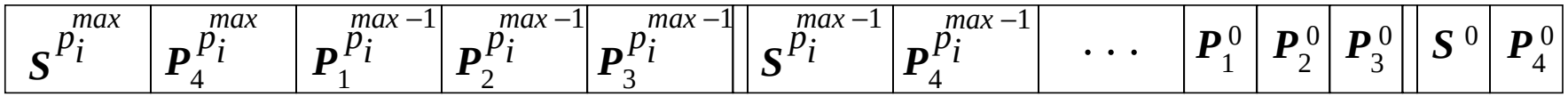


Fig. 9.9: Appearance of coding passes and quad tree codes in each block's embedded bitstream.

Post Compression Rate Distortion (PCRD) Optimization

- Goal:
  - Produce a truncation of the independent bitstream of each code block in an optimal way such that distortion is minimized, subject to the bit-rate constraint.
- For each truncated embedded bitstream of code block B_i having rate $R_i^{n_i}$ with distortion $D_i^{n_i}$ and truncation point n_i, the overall distortion of the reconstructed image is (assuming distortion is additive)

$$D = \sum_i D_i^{n_i} \tag{9.3}$$

- The optimal selection of truncation points n_i can be formulated into a minimization problem subject to the following constraint:

$$R = \sum_i R_i^{n_i} \leq R^{max} \tag{9.5}$$

- For some λ, any set of truncation point $\{n_i^\lambda\}$ that minimizes

$$(D(\lambda) + \lambda R(\lambda)) = \sum_i \left(D_i^{n_i^\lambda} + \lambda R_i^{n_i^\lambda} \right) \tag{9.6}$$

  is optimal in the rate-distortion sense.

- The distortion-rate slopes given by the ratios

$$S_i^{j_k} = \frac{\Delta D_i^{j_k}}{\Delta R_i^{j_k}} \tag{9.7}$$

  is strictly decreasing.

- This allows the optimization problem be solved by a simple selection through an enumeration $j_1 < j_2 < \cdots$ of the set of feasible truncation points.

$$n_i^{\lambda} = \max\left\{ j_k \in \mathcal{N}_i | S_i^{j_k} > \lambda \right\} \tag{9.8}$$

Layer Formation and Representation

- JPEG2000 offers both resolution and quality scalability through the use of a layered bitstream organization and a two-tiered coding strategy.

- The first tier produces the embedded block bit-streams while the second tier compresses block summary information.

- The quality layer $\mathcal{Q}_1$ contains the initial $R_i^{n_i^1}$ bytes of each code block B_i and the other layers $\mathcal{Q}_q$ contain the incremental contribution $L_i^q = R_i^{n_i^q} - R^{n_i^{q-1}} \geq 0$ from code block B_i.

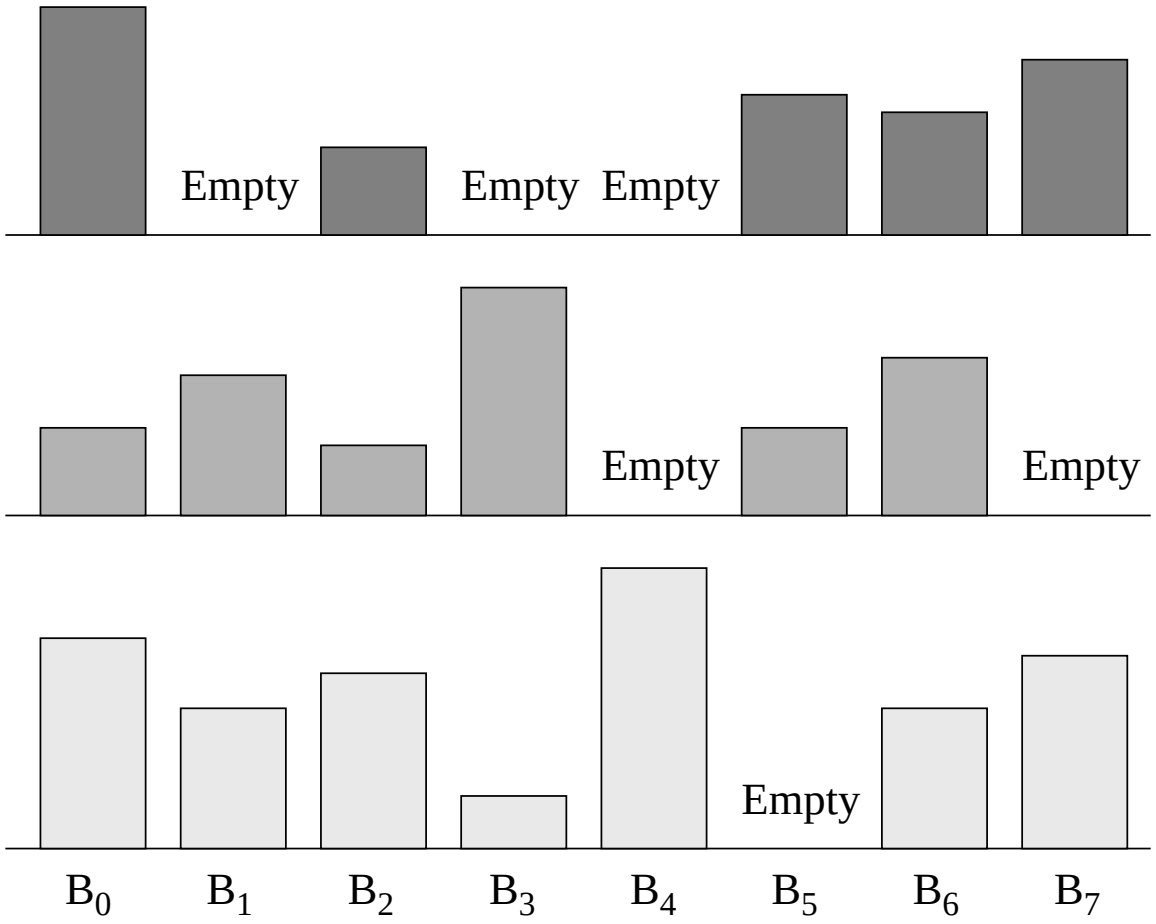
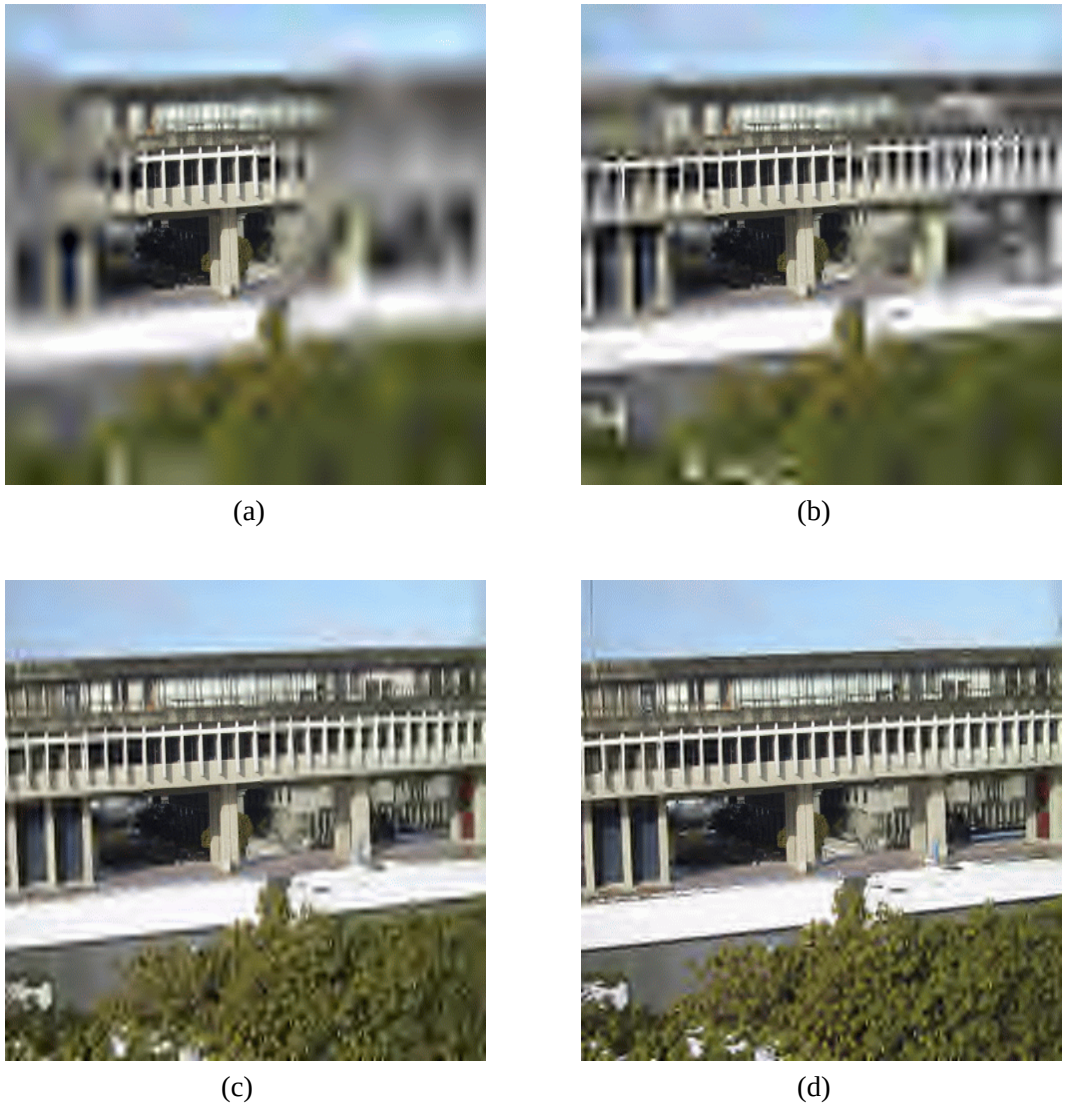


Fig. 9.10: Three quality layers with eight blocks each.

Region of Interest Coding in JPEG2000

- Goal:
  - Particular regions of the image may contain important information, thus should be coded with better quality than others.
- Usually implemented using the MAXSHIFT method which scales up the coefficients within the ROI so that they are placed into higher bit-planes.
- During the embedded coding process, the resulting bits are placed in front of the non-ROI part of the image. Therefore, given a reduced bit-rate, the ROI will be decoded and refined before the rest of the image.



(a) (b) (c) (d)

Fig. 9.11: Region of interest (ROI) coding of an image using a circularly shaped ROI. (a) 0.4 bpp, (b) 0.5 bpp, (c) 0.6bpp, and (d) 0.7 bpp.

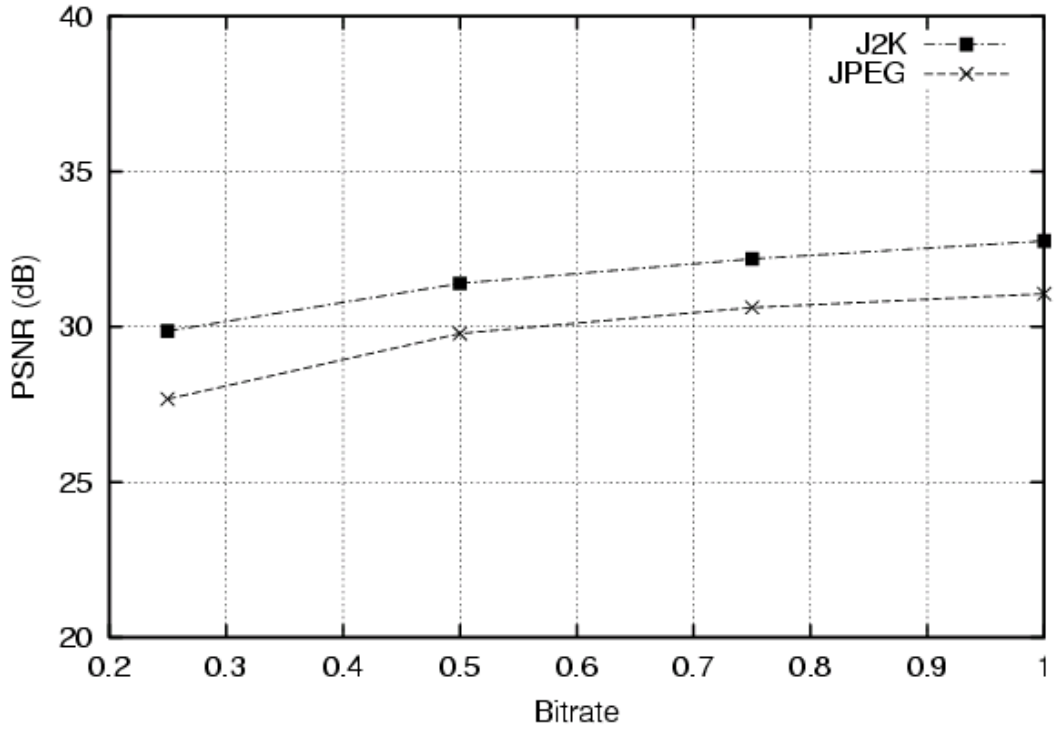
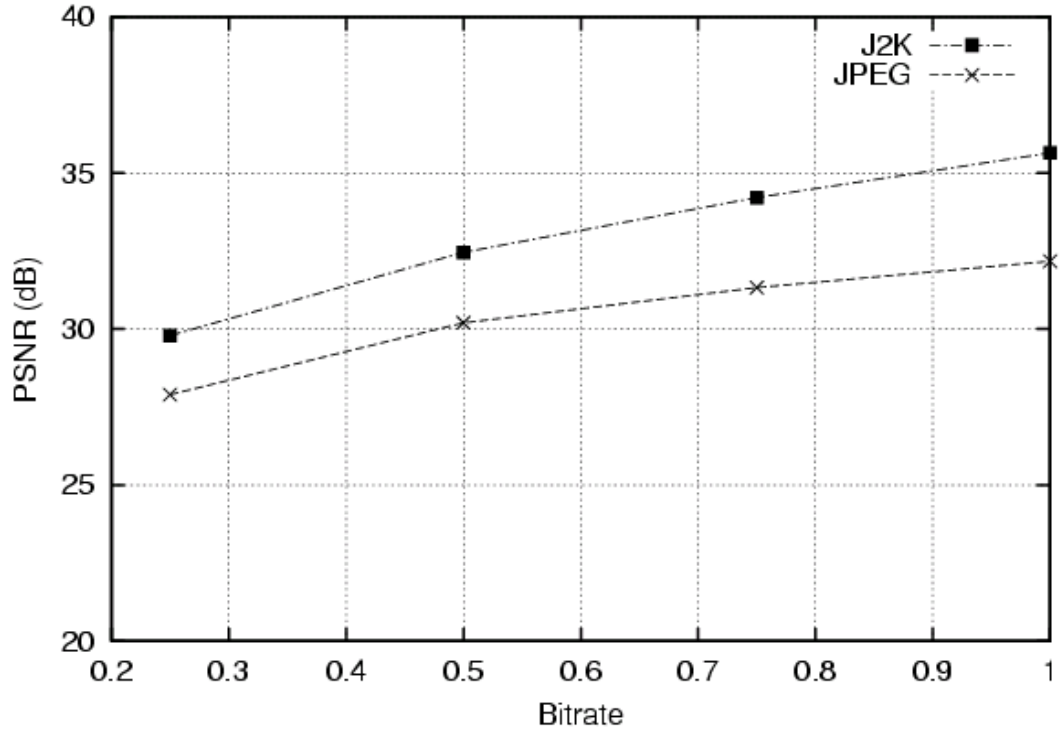


Fig. 9.12: Performance comparison for JPEG and JPEG2000 on different image types. (a): Natural images.



(b)

Fig. 9.12 (Cont'd): Performance comparison for JPEG and JPEG2000 on different image types. (b): Computer generated images.

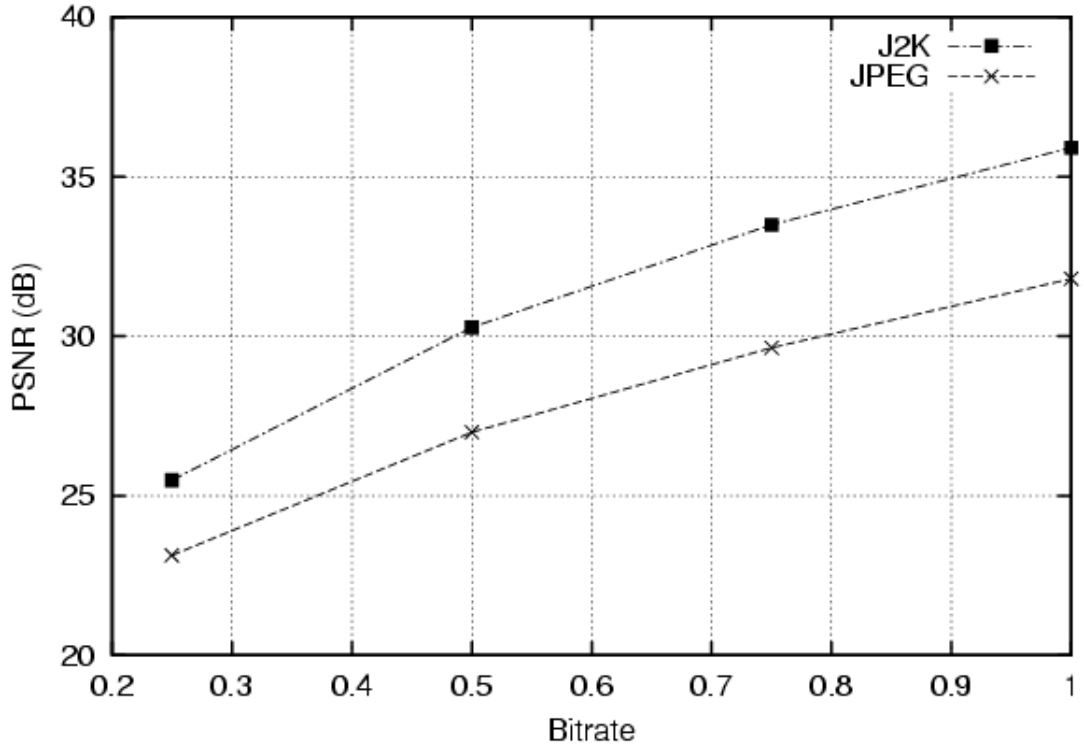


Fig. 9.12 (Cont'd): Performance comparison for JPEG and JPEG2000 on different image types. (c): Medical images.

(a)

Fig. 9.13: Comparison of JPEG and JPEG2000. (a) Original image.

(b)

(c)

Fig. 9.13 (Cont'd): Comparison of JPEG and JPEG2000. (b) JPEG (left) and JPEG2000 (right) images compressed at 0.75 bpp. (c) JPEG (left) and JPEG2000 (right) images compressed at 0.25 bpp.

9.3 The JPEG-LS Standard

- JPEG-LS is in the current ISO/ITU standard for lossless or "near lossless" compression of continuous tone images.
- It is part of a larger ISO effort aimed at better compression of medical images.
- Uses the LOCO-I (LOw COmplexity LOssless COmpression for Images) algorithm proposed by Hewlett-Packard.
- Motivated by the observation that complexity reduction is often more important than small increases in compression offered by more complex algorithms.

Main Advantage: Low complexity!

- The LOCO-I algorithm makes uses of *context modelling*.
- The idea of context modelling is to take advantage of the structure within the input source – the conditional probabilities.

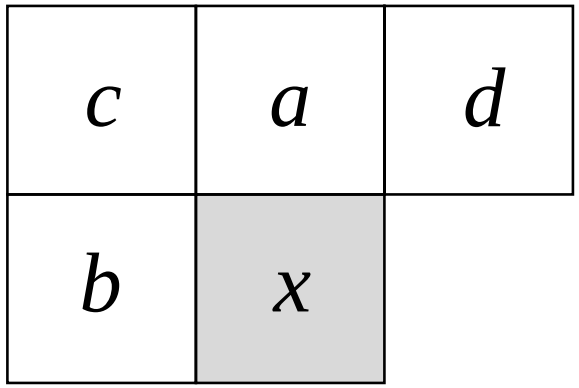


Fig. 9.14: JPEG-LS Context Model.

9.4 JBIG and JBIG-2: Bi-level Image Compression Standards

- Main Goal: Enables the handing of documents in electronic form.
- Primarily used to code scanned images of printed or hand-written text, computer generated text, and facsimile transmissions.
- JBIG is a lossless compression standard. It also offers progressive encoding/decoding capability, the resulting bitstream contains a set of progressively higher resolution images.
- JBIG-2 introduces *model-based coding* – similar to context-based coding. It supports lossy compressions well.

9.5 Further Explorations

- **Text books:**
  - *The JPEG Still Image Compression Standard* by Pennebaker and Mitchell
  - *JPEG2000: Image Compression Fundamentals, Standards, and Practice* by Taubman and Marcellin
  - *Image and Video Compression Standards: Algorithms and Architectures, 2nd ed.* by Bhaskaren and Konstantinides
- Interactive JPEG demo, and comparison of JPEG and JPEG2000
- **Web sites:** $\longrightarrow$ Link to Further Exploration for Chapter 9.. including:
  - JPEG and JPEG2000 links, source code, etc.
  - Original paper for the LOCO-I algorithm
  - Introduction and source code for JPEG-LS, JBIG, JBIG2