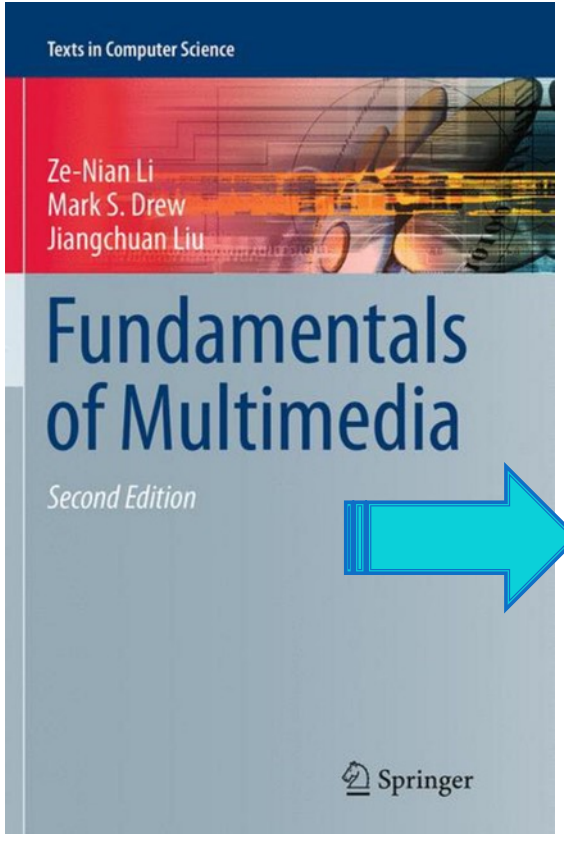
Background Photo:
Opera Garnier Ceiling

MULTIMEDIA

Dr Siba HAIDAR • INFO430 • 2019-2020

Textbook: Fundamentals of Multimedia • Z.-N. Li et al.

Course Outline



1. Introduction to multimedia
2. Digital representation of graphics and images
3. Colors in images and video
4. Fundamental Concepts in Video
5. Lossless compression algorithms
6. Lossy compression algorithms (JPEG)
7. Video Coding (MPEG)
8. Introduction to Image Processing

Background Photo:
The Starry Night

IMAGE COMPRESSION STANDARDS

Chapter Outline

- Image Compression Standards
- chapter 9 in textbook

1. The JPEG Standard
2. The JPEG2000 Standard
3. The JPEG-LS Standard
4. Bi-level Image Compression Standards

Background Photo:
The Milk Maid

The JPEG Standard

Why compress?

Multimedia image data	Grayscale image	Color image	HDTV video frame	Medical image	Super High Definition (SHD) image
Size/duration	512 × 512	512 × 512	1280 × 720	2048 × 1680	2048 × 2048
Bits/pixel or bits/sample	8 bpp	24 bpp	12 bpp	12 bpp	24 bpp
Uncompressed size (B for bytes)	262 KB	786 KB	1.3 MB	5.16 MB	12.58 MB
Transmission bandwidth (b for bits)	2.1 Mb/image	6.29 Mb/ image	8.85 Mb/ frame	41.3 Mb/ image	100 Mb/ image
Transmission time (56 K modem)	42 seconds	110 seconds	158 seconds	12 min.	29 min.
Transmission time (780 Kb DSL)	3 seconds	7.9 seconds	11.3 seconds	51.4 seconds	2 min.

Redundancy and Relevancy of images

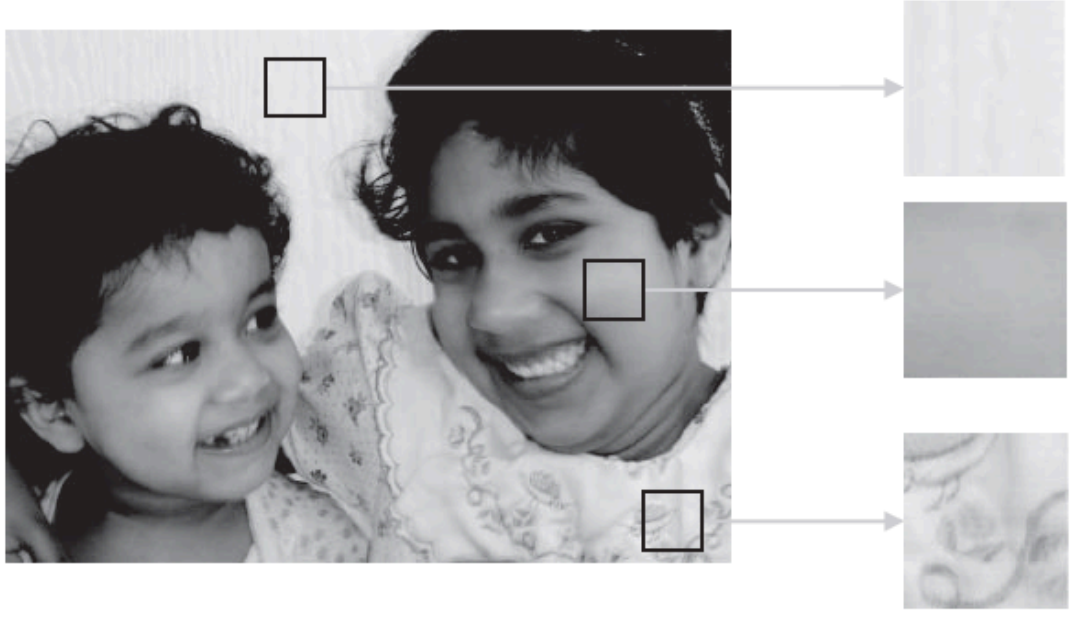
- Irrelevancy reduction:
  - Information associated with some pixels might be irrelevant and can be removed. Two categories of irrelevancy:
    - Visual irrelevancy or
    - Application-specific irrelevancy

- Visual: when image density exceeds the limits of display or viewing

- Application: an entire region of the image is unneeded, like in medical images or military
    - Can be either heavily compressed or excluded.

Redundancy Reduction

- Images has statistical redundancy
- Why ? → Because pixels are not random but highly correlated either locally or globally
- We can benefit from entropy-coding in the compression process

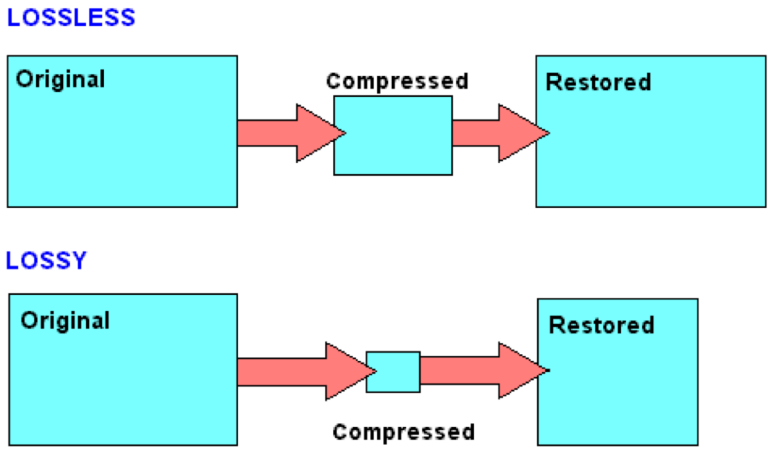


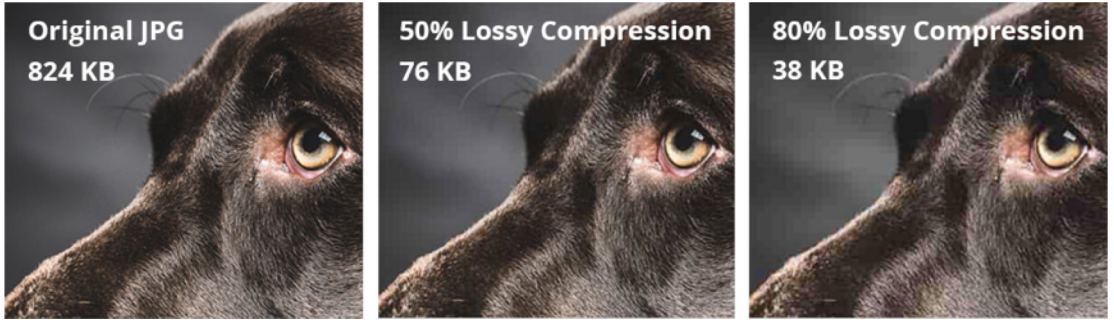
Types of redundancy

- **Spatial**: pixel intensities in a region
- **Spectral**: in Freq. domain, some frequencies dominate over others
- **Temporal**: correlation from frame to frame, will be covered in Video compression

The JPEG Standard

- image compression standard
- by "Joint Photographic Experts Group"
- international standard in 1992
- **lossy** image compression method
- **transform coding**
  - DCT (*Discrete Cosine Transform*)
  - image → function of $f(i, j)$ in *spatial domain*
  - 2D DCT yields frequency response $F(u, v)$ in *spatial frequency domain*





Observations for JPEG Image Compression

- effectiveness of DCT transform coding method relies on 3 major observations
- Observation 1:
  - useful image contents change relatively slowly across the image
  - 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
- → **spatial redundancy**

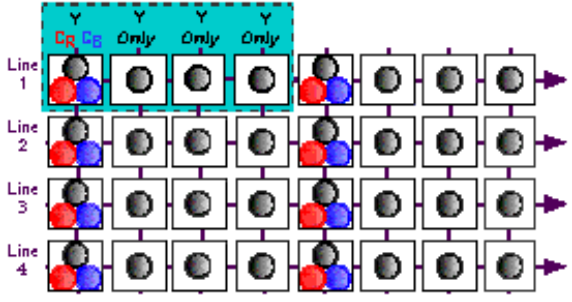
Observations for JPEG Image Compression

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

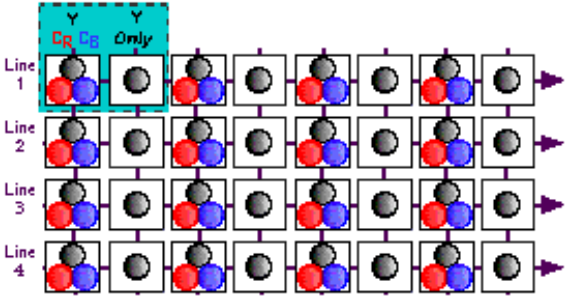
Observations for JPEG Image Compression

- 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

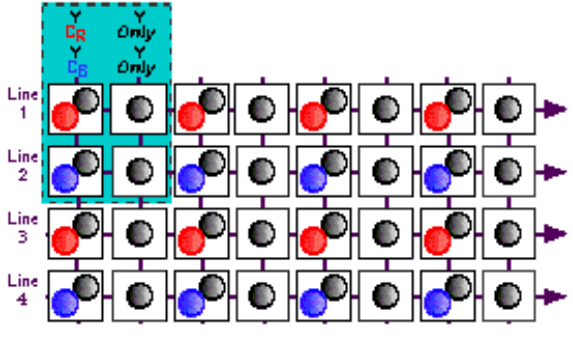
Luma Sampling & Chroma Sub-Sampling



4:1:1



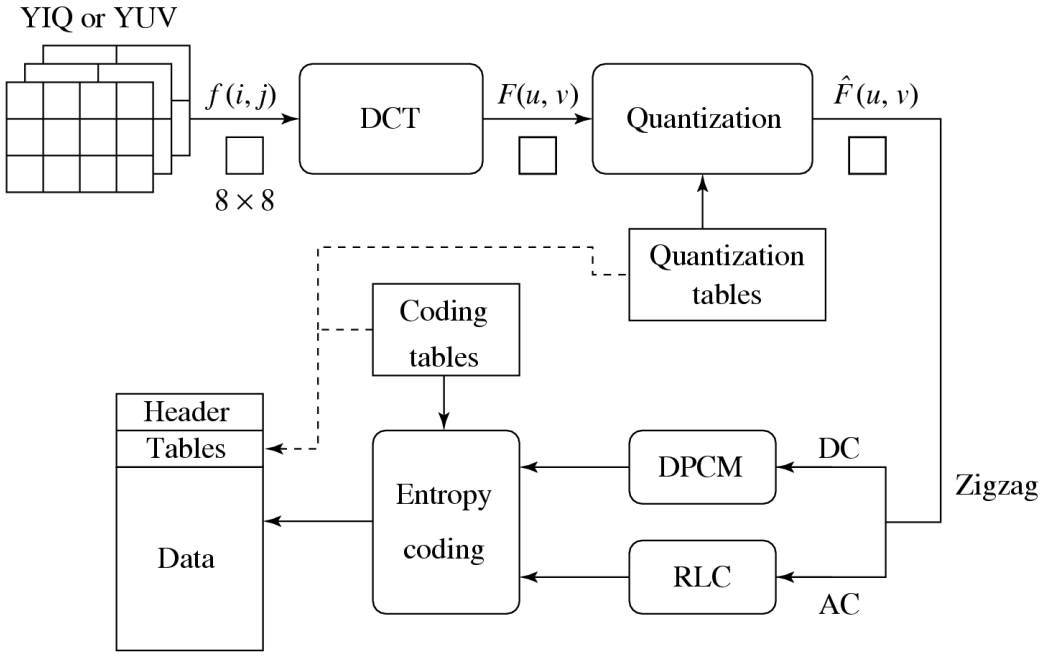
4:2:2



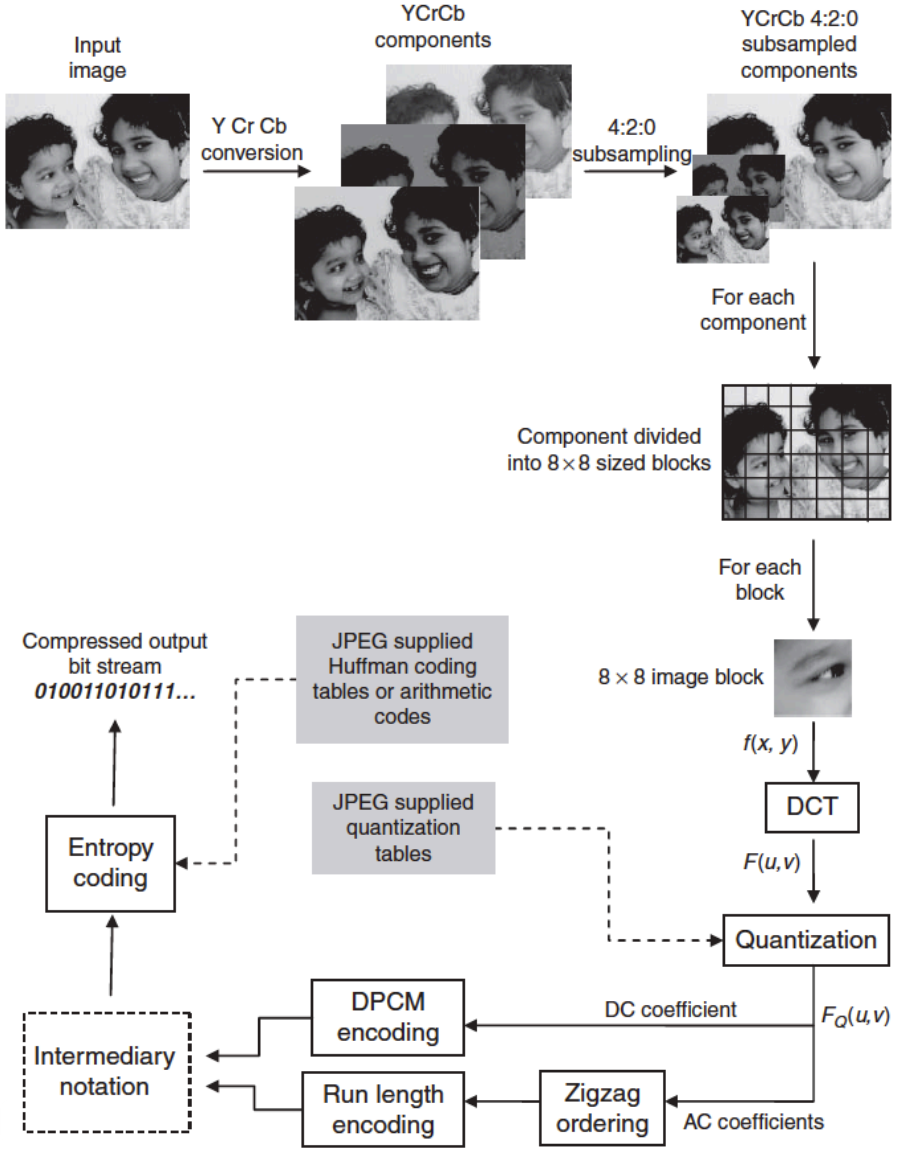
4:2:0

Block diagram for JPEG encoder

- 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



Example


Input image
Y Cr Cb conversion
YCrCb components
4:2:0 subsampling
YCrCb 4:2:0 subsampled components
For each component
Component divided into 8×8 sized blocks
For each block
8 × 8 image block
f(x, y)
DCT
F(u,v)
Quantization
JPEG supplied quantization tables
JPEG supplied Huffman coding tables or arithmetic codes
Compressed output bit stream
010011010111...
Entropy coding
Intermediary notation
DPCM encoding
DC coefficient
$F_Q(u,v)$
Run length encoding
Zigzag ordering
AC coefficients

Spatial Frequency and DCT

- ***spatial frequency*** → indicates how many times pixel values change across an image block
- DCT formalizes this notion
  - how much the image contents change in correspondence to the number of cycles of a cosine wave per block
  - → *decompose* original signal into its DC and AC components;
- IDCT → *reconstruct* signal

- given an input function f(i, j) over 2 ints i & j (piece of an image)
  - 2D DCT transforms it into a new function F(u, v)
  - with integer u and v running over same range as i & j
- general definition:

$$F(u,v)=\frac{2C(u)C(v)}{\sqrt{MN}}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\cos\frac{(2i+1)\cdot u\pi}{2M}\cdot\cos\frac{(2j+1)\cdot v\pi}{2N}\cdot f(i,j)$$

  - where
  - i, u = 0, 1, . . . ,M − 1;
  - j, v = 0, 1, . . . ,N − 1;
  - and the constants C(u) and C(v):

$$C(\xi)=\begin{cases}\frac{\sqrt{2}}{2} & if \quad \xi=0,\\ 1 & otherwise.\end{cases}$$

DCT on image blocks

- each image is divided into **8 × 8** blocks
- 2D DCT applied to each block image $f(i, j)$ → output DCT coefficients $F(u, v)$ for each block
- using blocks → bad 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
- F → 8x8
- F(0,0) → **DC** component
  - usually highest values because energy in natural photographs is concentrated among the lowest frequencies
- remaining F(u, v) are called **AC** components
- then they are quantized using a table supplied by JPEG

2D DCT

$$F(u,v)=\frac{C(u)C(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}f(i,j)$$

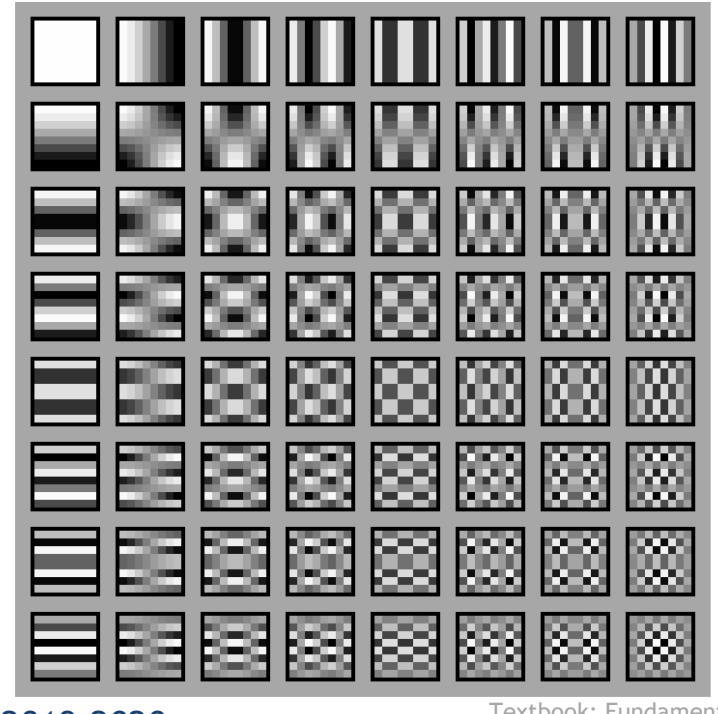
- where i, j, u, v = 0, 1, . . . , 7, and
- constants C(u) and C(v) are determined by previous slide

- 2D Inverse Discrete Cosine Transform (2D IDCT):
  - inverse function almost same
  - with f(i, j) & F(u, v) reversed
  - & C(u)C(v) inside sums:

$$\tilde{f}(i,j)=\sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C(u)C(v)}{4}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}F(u,v)$$

  - where i, j, u, v = 0, 1, . . . , 7

- graphical illustration of 8×8 2D DCT basis



Quantization

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

- F(u, v) → DCT coefficient
- Q(u, v) → "quantization matrix" entry
- $\hat{F}(u,v)$ → quantized DCT coefficients

- quantization step → **main source for loss in JPEG compression**
  - entries of Q(u, v) tend to have larger values towards the lower right corner
  - introduce more loss at the higher spatial frequencies — a practice supported by Observations 1 and 2

- Default 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

- Default 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

JPEG compression for a smooth image block

- 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)

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

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

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

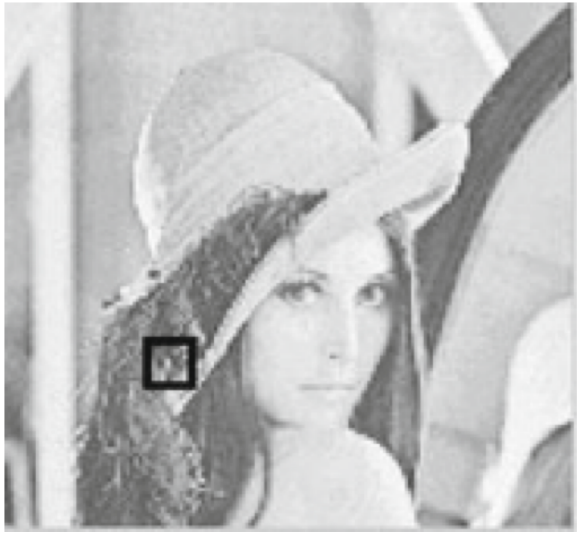
$\tilde{f}(i, j)$

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

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

JPEG compression for another 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)

JPEG compression for another 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

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

−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

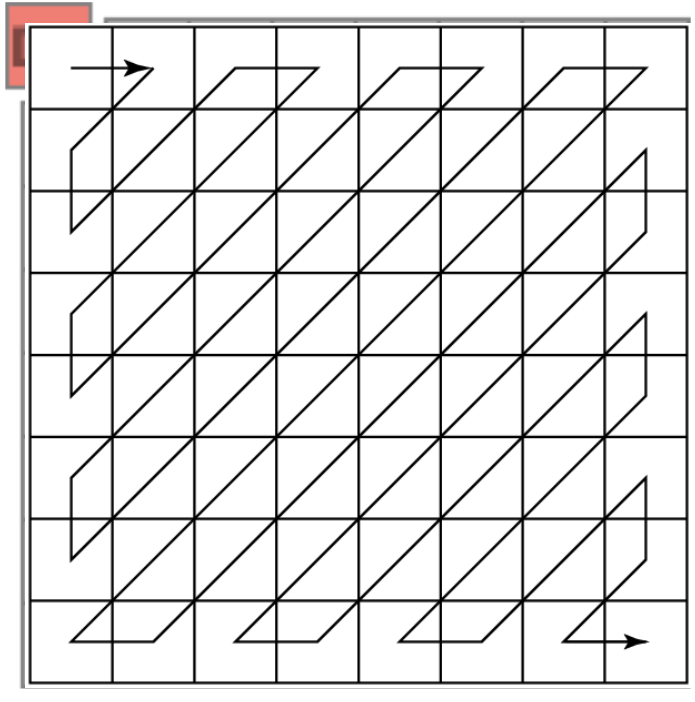
$(i, j) = f(i, j) - \tilde{f}(i, j)$

Run-length Coding (RLC) on AC coefficients

- RLC aims to turn the $\hat{F}(u,v)$ 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 $\hat{F}(u,v)$ into a *64-vector*

- 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),
  - SIZE indicates how many bits are needed for representing the coefficient
  - 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

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

Background Photo:
The Milk Maid

Example 2

DCT

178	187	183	175	178	177	150	183
191	174	171	182	176	171	170	188
199	153	128	177	171	167	173	183
195	178	158	167	167	165	166	177
190	186	158	155	159	164	158	178
194	184	137	148	157	158	150	173
200	194	148	151	161	155	148	167
200	195	172	159	159	152	156	154

Pixel values $f(x, y)$

1359	46	61	26	38	–21	–5	–18
31	–35	–25	–11	13	10	12	–3
13	20	–17	–14	–11	–7	6	5
–5	5	2	–8	–11	–26	8	–4
10	15	–10	–16	–21	–7	8	7
–6	1	0	7	5	–7	–1	–3
–13	–8	1	10	8	4	–3	–4
–5	–5	–2	5	5	0	0	–3

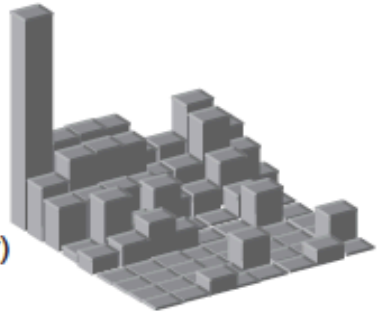
DCT values $F(u,v)$

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

Quantization table

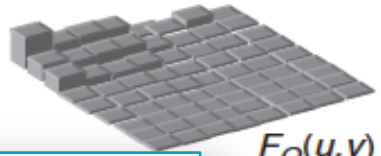
$f(x, y)$

DCT



$F(u,v)$

Quantization

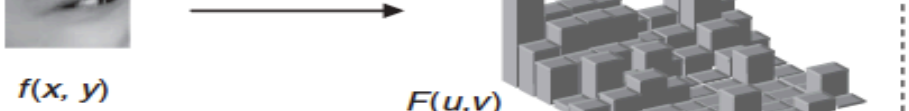


$F_Q(u,v)$

85	4	6	2	2	–1	0	0
3	–3	–2	–1	1	0	0	0
1	2	–1	–1	0	0	0	0
0	0	0	0	0	0	0	0
1	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

$$F_Q(u,v) = \left\lceil \frac{F(u,v)}{Q(u,v)} \right\rceil, \text{ where } Q(u,v) \text{ is the value in the quantization table}$$

DCT

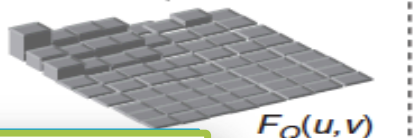


$f(x, y)$

$F(u,v)$

85	4	6	2	2	–1	0	0
3	–3	–2	–1	1	0	0	0
1	2	–1	–1	0	0	0	0
0	0	0	0	0	0	0	0
1	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

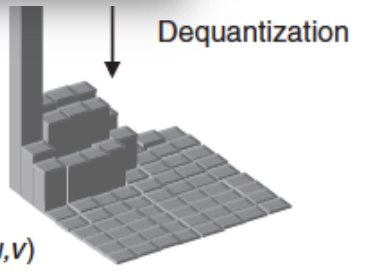
Quantization



$F_Q(u,v)$

encoding ... next slide

Dequantization



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

1360	44	60	32	48	–40	0	0
36	–36	–28	–19	26	0	0	0
14	26	–16	–24	0	0	0	0
0	0	0	0	0	0	0	0
18	22	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

Inverse DCT

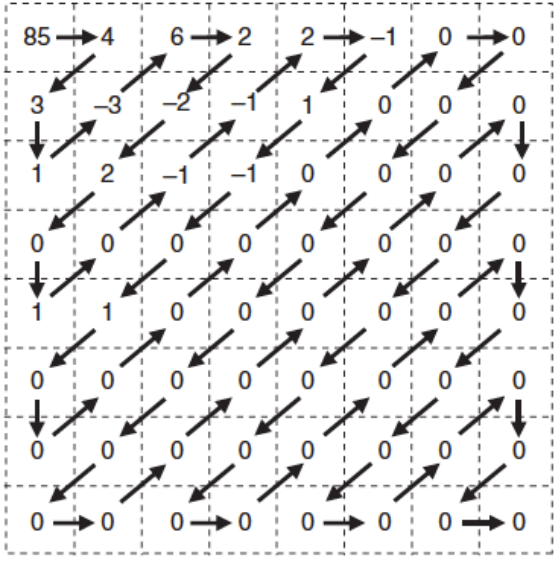
$\hat{f}(x, y)$

192	185	178	152	193	162	155	190
181	172	162	154	187	164	159	192
183	168	150	158	181	167	162	190
198	177	150	155	177	169	161	182
202	180	148	148	171	167	159	175
193	176	145	141	162	162	156	170
195	184	155	145	159	156	150	164
209	200	170	154	160	153	144	156

Encoding …

- Intermediary step:
  - DC is encoded using DPCM (error calculation)
  - AC are calculated using run length (remember there are a lot of ZEROS)
  - ZIZGAG scan AC → longer runs of zeros compared with raster scan

- Zigzag ordering



DC coefficient = 85

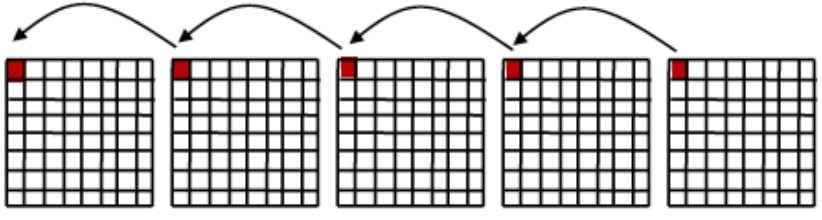
AC coefficient stream

4 3 1 -3 6 2 -2 2 0 1 0 -1 -1 2
-1 1 -1 0 1 0 0 0 0 0 0 0 0 0 ...

Steps continued

For DC

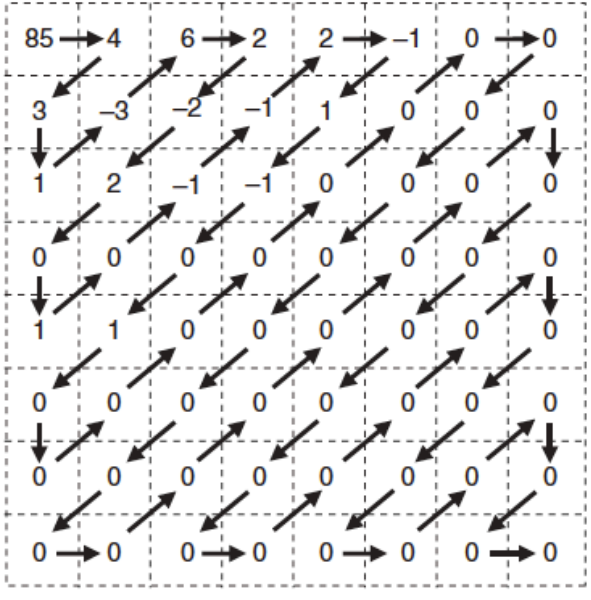
- - ce between 2 DC block values is encoded in 2 notations:
- <size><amplitude>
- Example: DC=85 and DC=82 → diff = 3
- <2><3>: 2 bits to represent value 3



For AC

- only non-zero coefficients are used
- <runlength, size> <Amplitude of non-zero>
  - runlength: number of zero AC that precedes it in the ZigZag
  - size: nb. Of bits to represent the amplitude
- 1's complement for negative numbers

AC & DC coefficients



DC coefficient = 85

AC coefficient stream

4 3 1 -3 6 2 -2 2 0 1 0 -1 -1 2
-1 1 -1 0 1 0 0 0 0 0 0 0 0 0 ...

Intermediary stream

<2><3> <0,3><4> <0,2><3> <0,1><1> <0,2><-3> <0,3><6>
<0,2><2> <0,2><-2> <0,2><2> <1,1><1> <1,1><-1> <0,1><-1>
<0,2><2> <0,1><-1> <0,1><1> <0,1><-1> <1,1><1> EOB

Last steps

- For both DC and AC
- First part <> → use Huffman
- Second part <> → use non-prefixed representation

Intermediary symbol	Binary representation of first symbol (prefixed Huffman Codes)	Binary representation of second symbol (non-prefixed variable integer codes)
<2> <3>	011	11
<0,3> <4>	100	100
<0,2> <3>	01	11
<0,1> <1>	00	1
<0,2> <−3>	01	00
<0,3> <6>	100	110
<0,2> <2>	01	10
<0,2> <−2>	01	01
<0,2> <2>	01	10
<1,1> <1>	11	1
<1,1> <−1>	11	0
<0,1> <−1>	00	0
<0,2> <2>	01	10
<0,1> <−1>	00	0
<0,1> <1>	00	1
<0,1> <−1>	00	0
<1,1> <1>	11	1
EOB	1010	

Binary Stream:

011111001000111001010010011001100101011011111000001100000010001111010

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 previous chapter 5

Progressive Mode

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

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

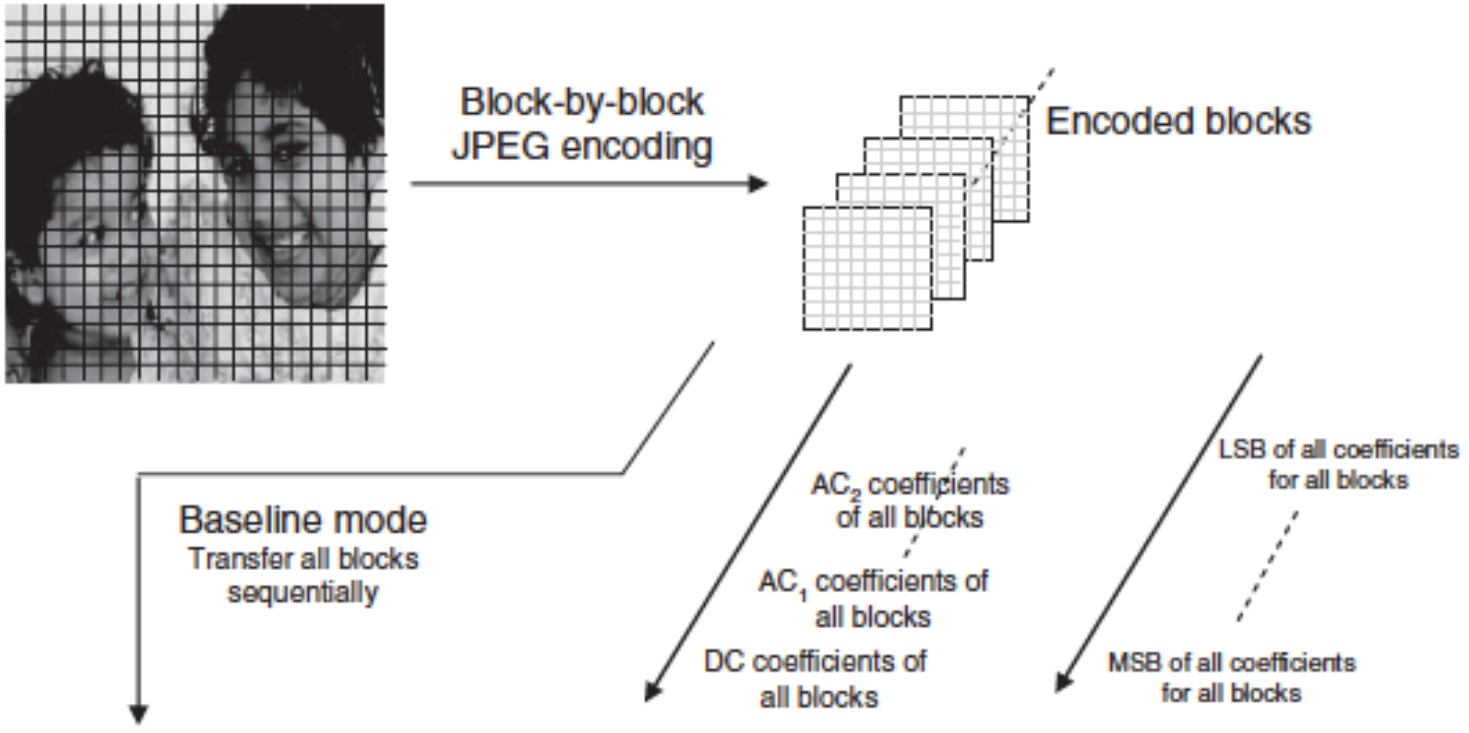
- Scan 1: Encode the first few MSBs, e.g., Bits 7, 6, 5, 4.
- Scan 2: Encode a few more less significant bits, e.g., Bit 3.
- ...
- Scan m: Encode the least significant bit (LSB), Bit 0.

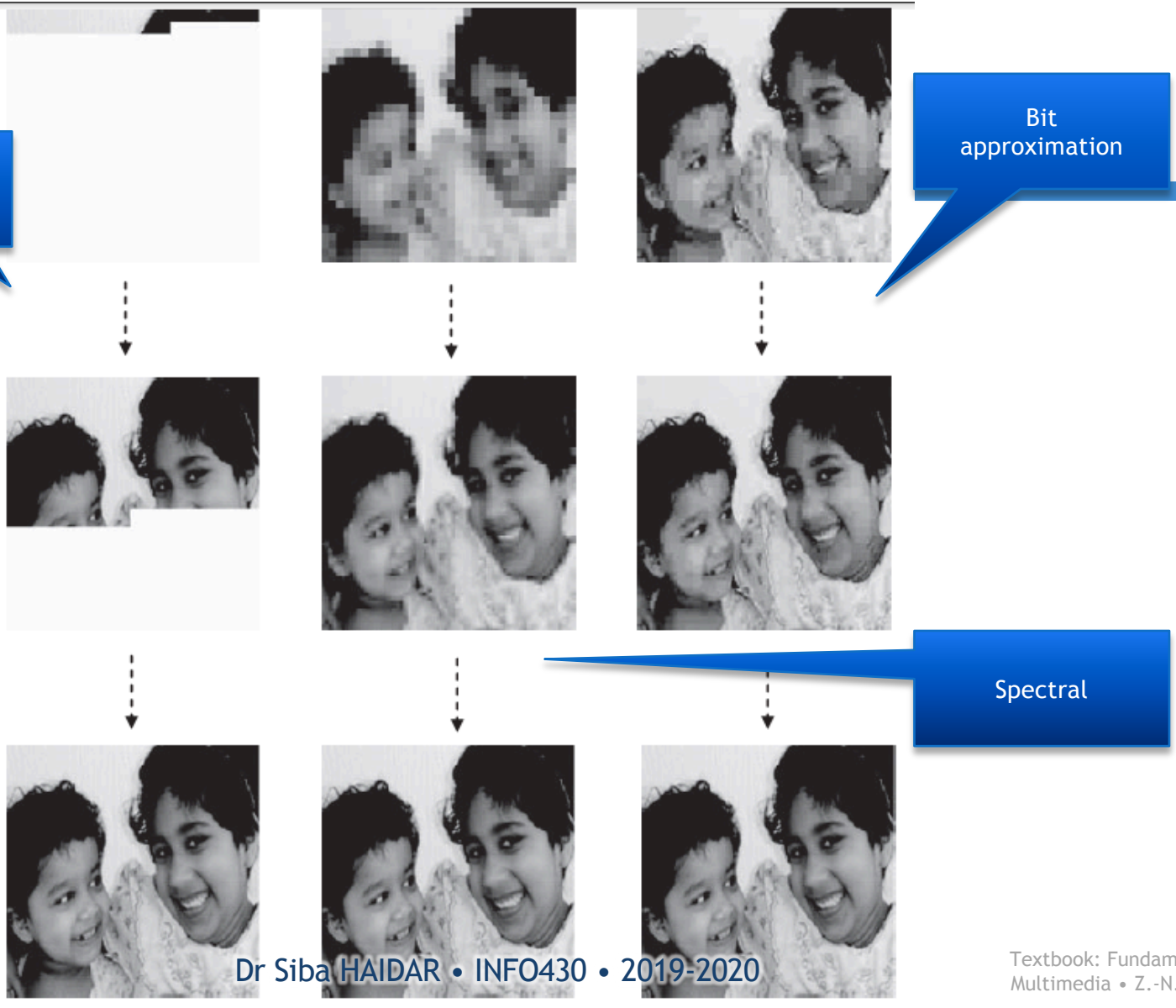
Progressive Transmission of DCT-based images

- **Spectral selection:**
- First scan: send all DC of all blocks
- Second scan: send 1st AC coefficient
- Third scan: send 2nd AC and so on

- **Successive bit approximation**
- All coefficients in one scan but bit by bit manner
- First scan: send MSB of all coef
- Second scan: send second MSB and so on.

Progressive Transmission in JPEG




baseline
Bit approximation
Spectral

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.

A Glance at the JPEG Bitstream

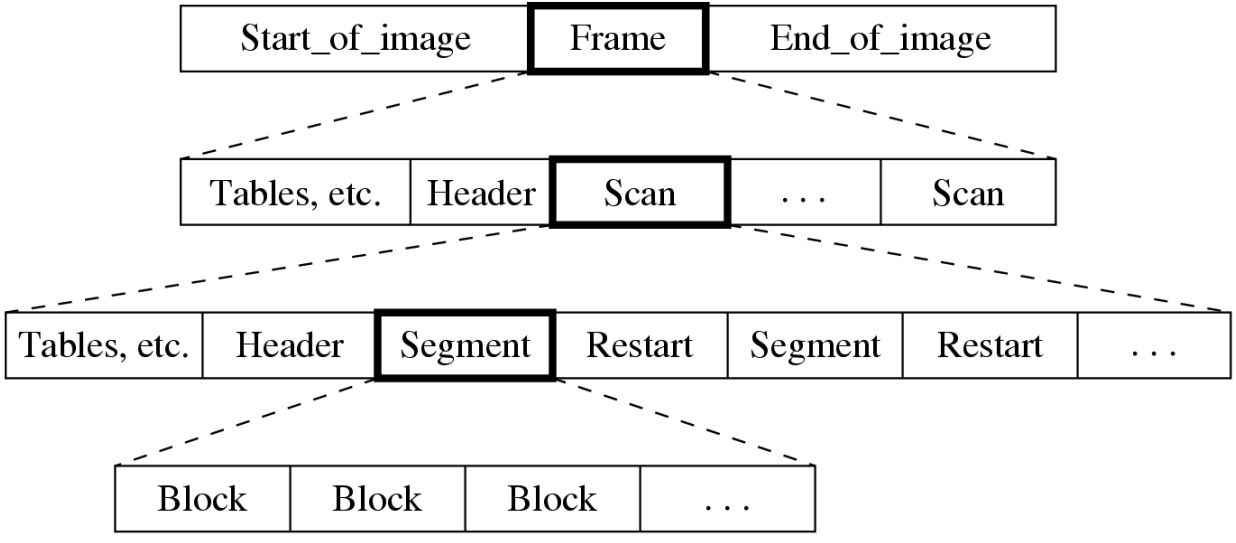


Fig. 9.6: JPEG bitstream.

JPEG drawbacks

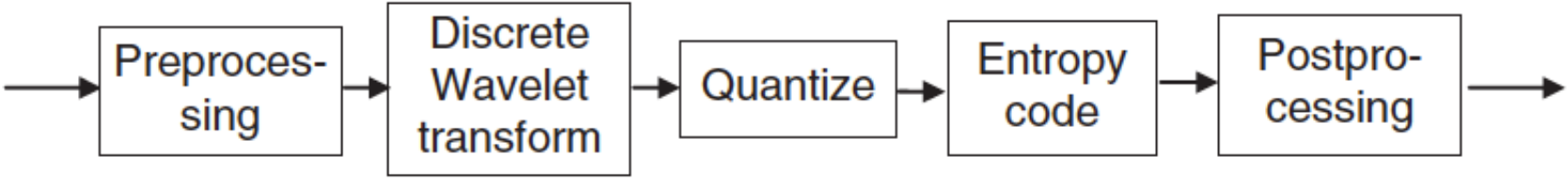
- poor low bit compression → at low bit rates, the perceived distortion becomes unacceptable
- does not allow random access to bit stream
- large image handling → JPEG does not allow compression of images larger than 64K by 64K
- transmission in noisy environments (especially in wireless) which was not taken into account in the standard.
- not suited for computer generated images and documents (it was developed for natural tone images)
- this all led to the development of Jpeg2000

Background Photo:
The Milk Maid

The JPEG 2000 Standard

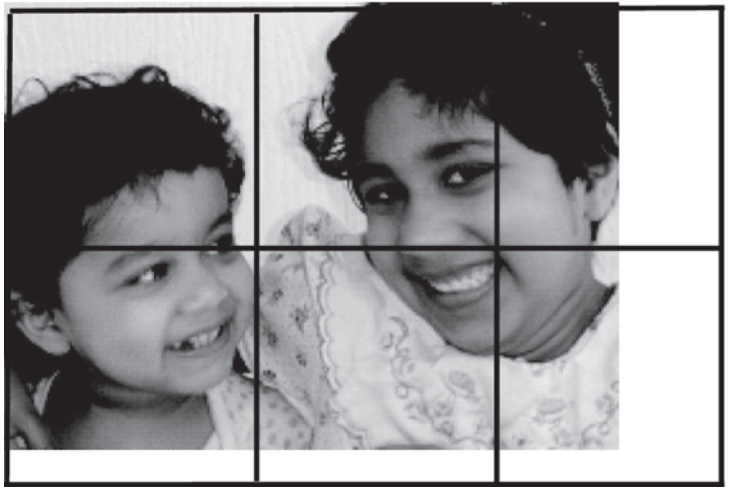
DWT

- JPEG 2000 used Discrete Wavelet Transform
- It's better than DCT since it distributes energy among all coefficients
- DCT works on 8x8 blocks while DWT on the whole image



I. Preprocessing

- 1) **Tiling**: splitting the image into rectangular but equal sized blocks
- Each will be DWTed independently so encoding and decoding will be faster
- Better memory management due to smaller sizes
- optional



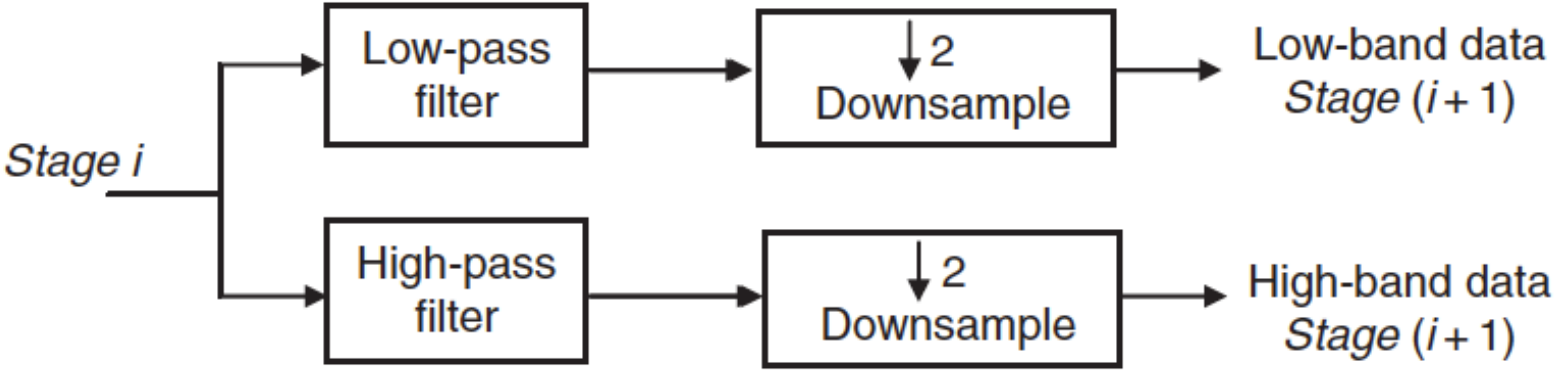
I. Preprocessing

- 2) **YCrCb conversion**
  - Convert to YCrCb color space to take advantage of human vision characteristics
  - Each channel is then processed independently with different tolerances

- 3) **Level offsetting**: shifting the DC levels
  - Done by subtracting a constant value from all the pixels
  - DWT requires the dynamic range of pixel values to be centered around 0
  - For n-bit representation, values should be centered from $-2^{(n-1)}$ to $2^{(n-1)}-1$

The DWT

- A low pass and high pass are applied to the image
- Each result is the same size of the image resulting in double original size
- So we down sample by 2 each one, so we keep same size.



The DWT

- Since the image is 2-D, we can use Mallat's algorithm to use 1-D filters
- Filter the rows first (low and high) using 1-D
- Then each result is filtered along the columns using also 1-D
- Result: 4 bands: LL, LH, HL, HH

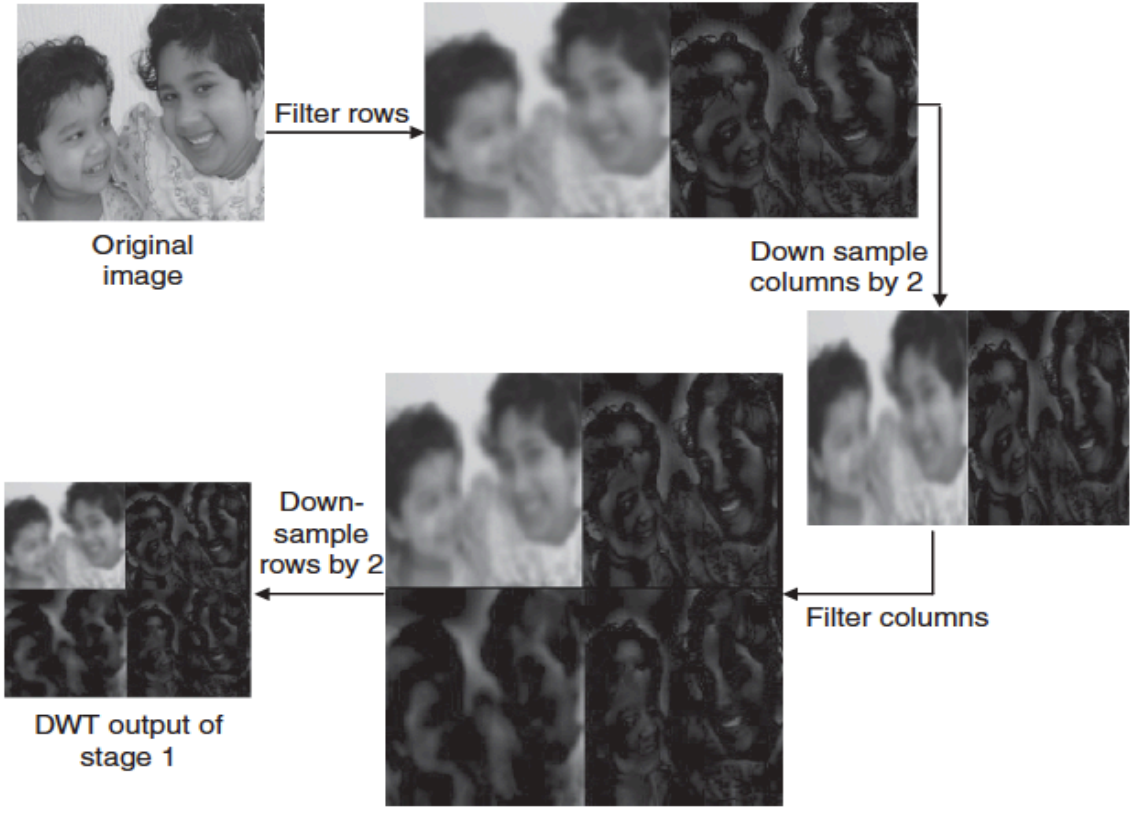
LL—Low subbands of the filtering in both dimensions, rows and columns

HL—High subbands after row filtering and low subbands after column filtering

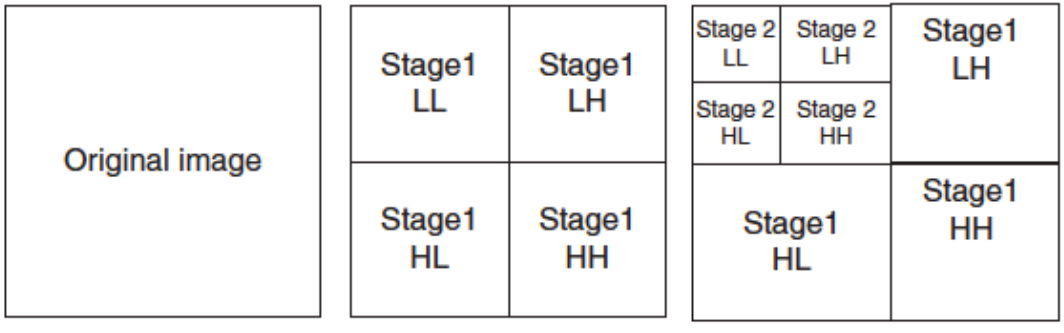
LH—Low subbands after row filtering and high subbands after column filtering

HH—High subbands after both row and column filtering

DWT


Filter rows
Original image
Down sample columns by 2
Filter columns
Down-sample rows by 2
DWT output of stage 1

Each time, we take LL and apply another level of DWT



JPEG 2000 allows up to 32 stages but usually is 4 to 8

Advantages of JPEG2000

- Encode once, Platform dependent decoding
  - We can decode at several frequency or spatial resolutions depending on the display and bandwidth available
- Working with compressed images:
  - Imaging operations can be performed on the compressed version (crop, flip, rotate,…)
- Region of interest encoding