Chapter-2: Image Compression (JPEG), Entropy Coding and Brief Intro to Video Compression

JPEG Image Compression Standard

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.

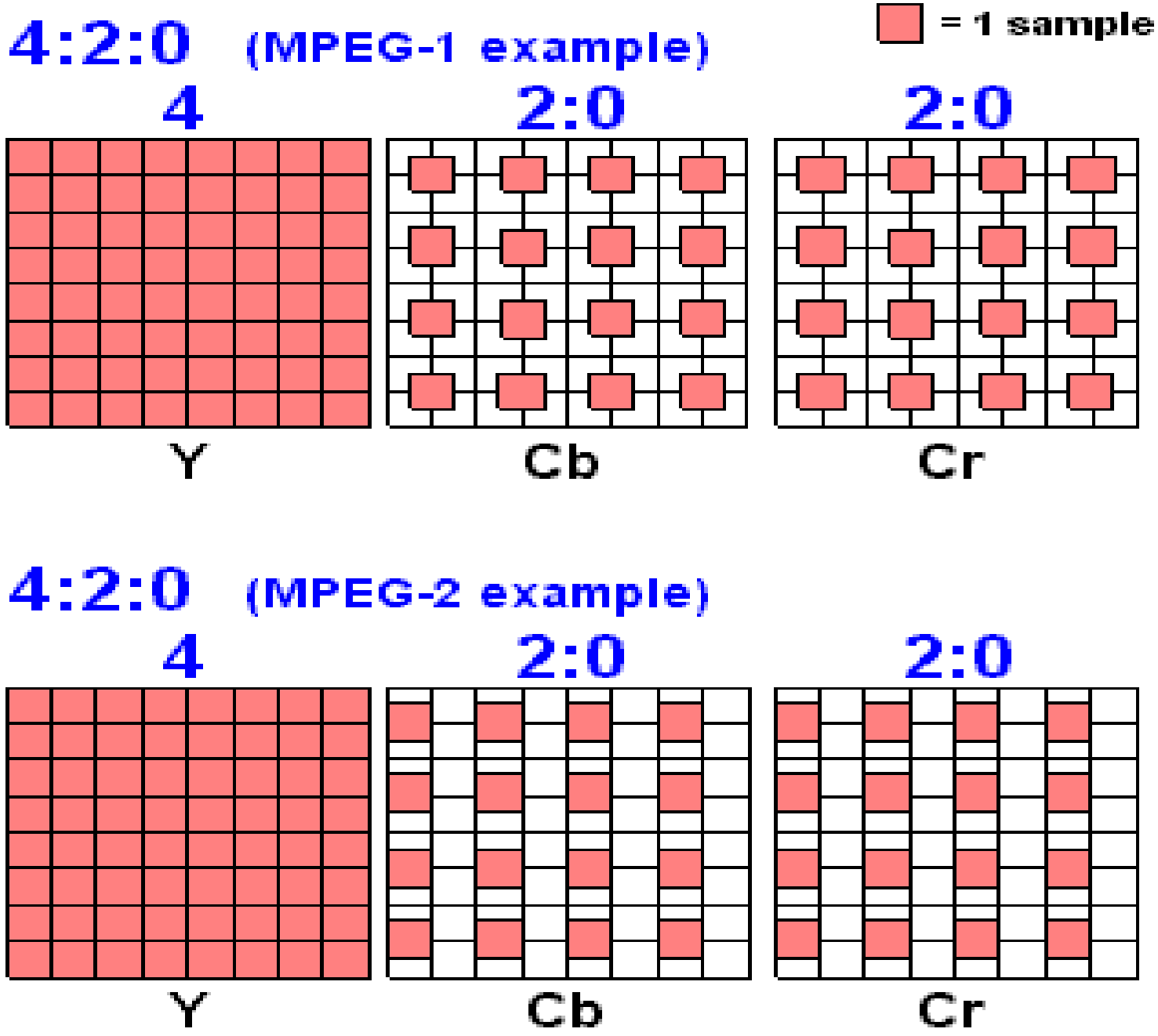
RGB <-> YCbCr Conversion

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

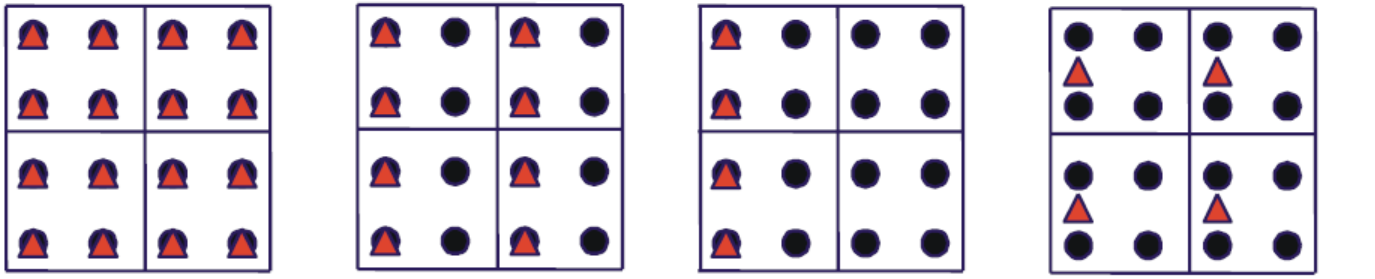
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & -0.001 & 1.402 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.772 & 0.001 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

Note: Cb ~ Y-B, Cr ~ Y-R, are known as color difference signals.

Please look at the Color Format section in Chapter-1


= 1 sample
4:2:0 (MPEG-1 example)
4
2:0
2:0
Y
Cb
Cr
4:2:0 (MPEG-2 example)
4
2:0
2:0
Y
Cb
Cr

Chrominance Subsampling



4:4:4
For every 2x2 Y Pixels
4 Cb & 4 Cr Pixel
(No subsampling)

4:2:2
For every 2x2 Y Pixels
2 Cb & 2 Cr Pixel
(Subsampling by 2:1 horizontally only)

4:1:1
For every 4x1 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 4:1 horizontally only)

4:2:0
For every 2x2 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 2:1 both horizontally and vertically)

● Y Pixel ▲ Cb and Cr Pixel

4:2:0 is the most common format

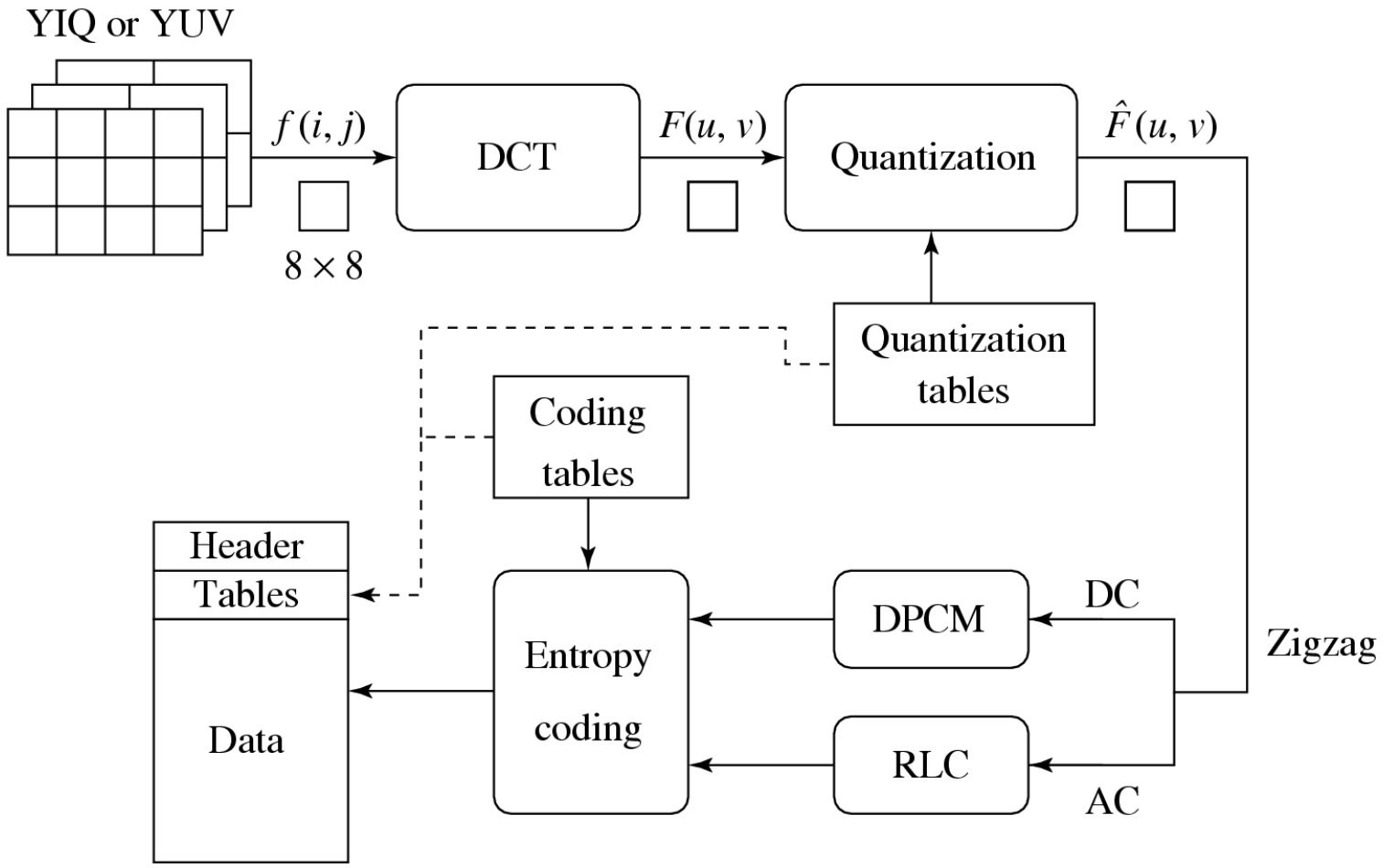


Fig. 9.1: Block diagram for JPEG encoder.

9.1.1 Main Steps in JPEG Image Compression

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

Introduction to Transform Coding

- **The rationale behind transform coding**:

  If $\mathbf{Y}$ is the result of a linear transform $\mathbf{T}$ of the input vector $\mathbf{X}$ in such a way that the components of $\mathbf{Y}$ are much less correlated, then $\mathbf{Y}$ can be coded more efficiently than $\mathbf{X}$.

- If most information is accurately described by the first few components of a transformed vector, then the remaining components can be coarsely quantized, or even set to zero, with little signal distortion.

- Discrete Cosine Transform (DCT) will be studied.

Spatial Frequency and DCT

- *Spatial frequency* indicates how many times pixel values change across an image block.

The DCT formalizes this notion with a measure of how much the image contents change in correspondence to the number of cycles of a cosine wave per block.

The role of the DCT is to *decompose* the original signal into low frequency and high frequency components; the role of the IDCT is to *reconstruct* (re-compose) the signal.

In most images, there are a lot of low frequency components, but mush less high frequency components. Thus, we can retain the low frequency and get rid of many high frequency components in the DCT domain.

Definition of DCT:

Given an input function $f(i, j)$ over two integer variables i and j (a piece of an image), the 2D DCT transforms it into a new function $F(u, v)$, with integer u and v running over the same range as i and j. The general definition of the transform is:

(8.15)

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

where $i, u = 0, 1, \ldots, M-1$; $j, v = 0, 1, \ldots, N-1$; and the constants $C(u)$ and $C(v)$ are determined by

(8.16)

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

2D Discrete Cosine Transform (2D DCT):

$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j) \qquad (8.17)$$

where $i, j, u, v = 0, 1, \ldots, 7$, and the constants $C(u)$ and $C(v)$ are determined by Eq. (8.5.16).

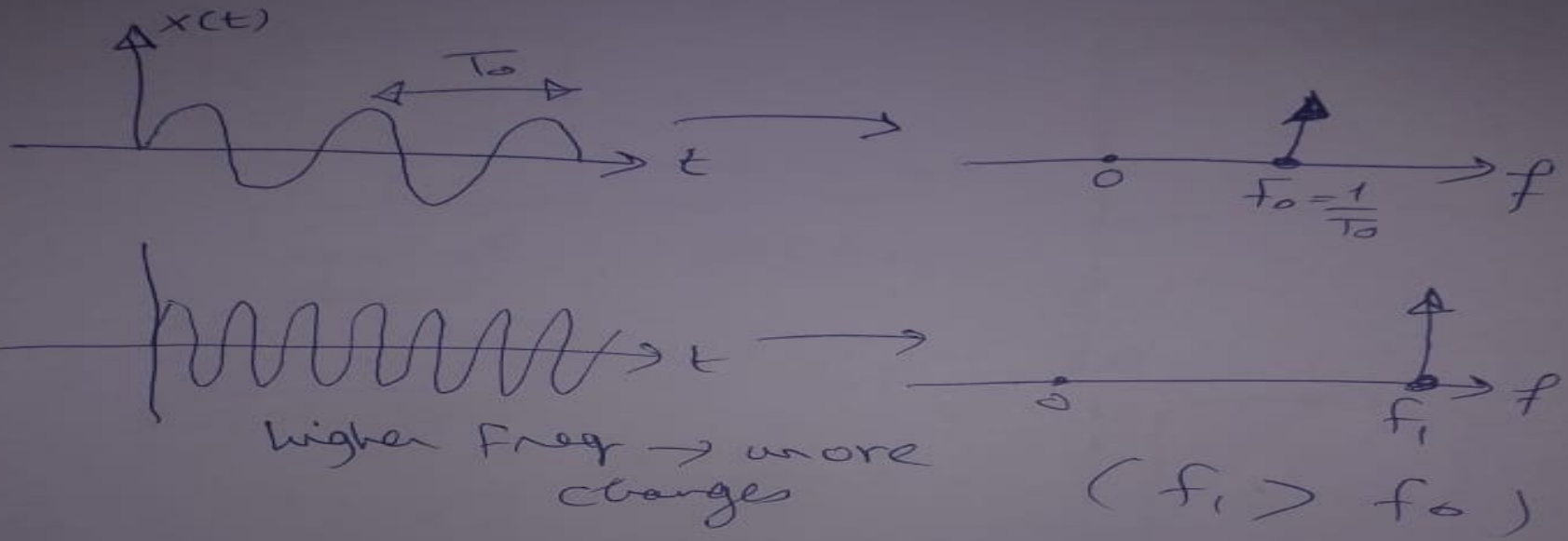
2D Inverse Discrete Cosine Transform (2D IDCT):

The inverse function is almost the same, with the roles of $f(i, j)$ and $F(u, v)$ reversed, except that now $C(u)C(v)$ must stand inside the sums:

$$\tilde{f}(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)C(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v) \qquad (8.18)$$

where $i, j, u, v = 0, 1, \ldots, 7$.

(DCT2) and frequency domain



higher Freq → more changes

$(f_1 > f_0)$

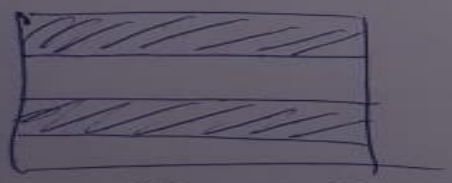


Image ①

↳ changes in vertical direction

$f_v > 0$

$f_h = 0$

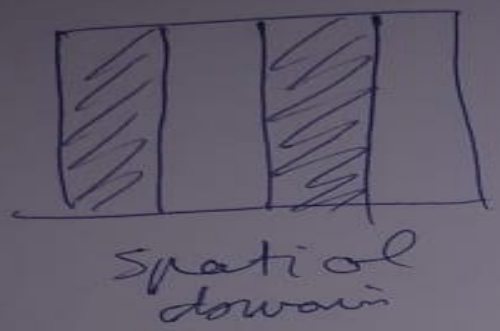
Image ②

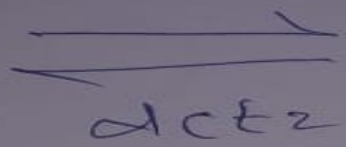
changes in horiz. direction

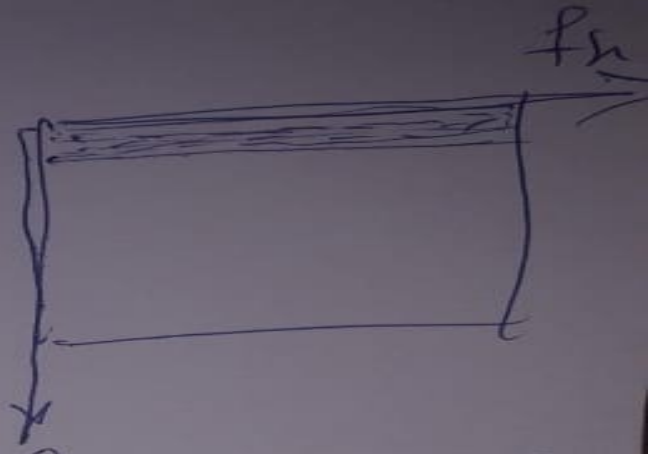
$f_v = 0$

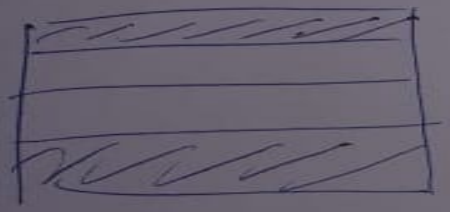
$f_h > 0$

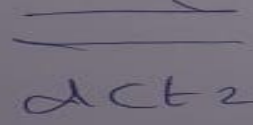
(2)

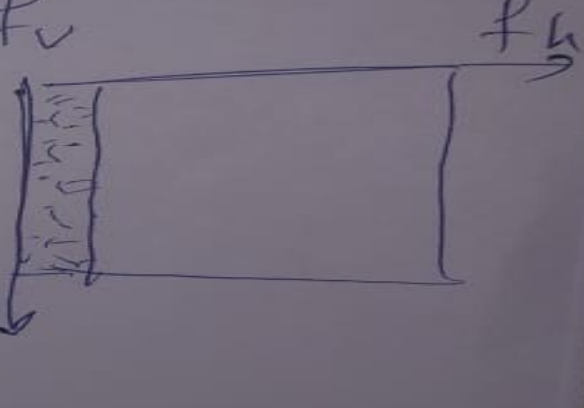


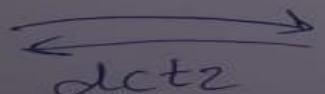


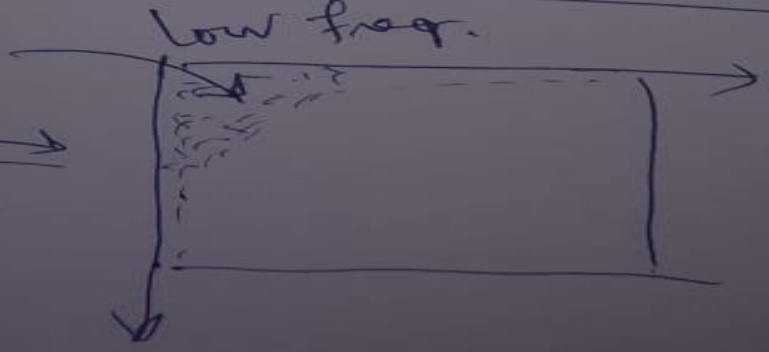




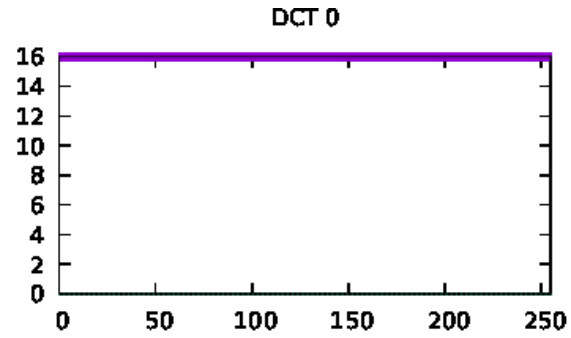


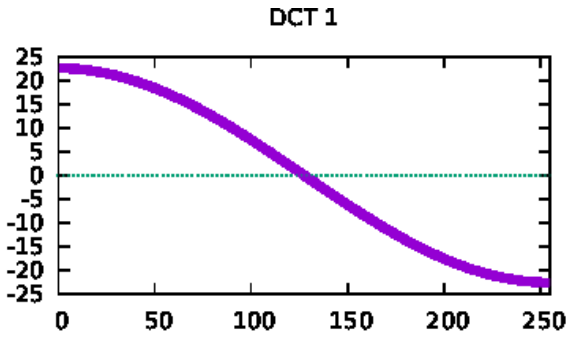


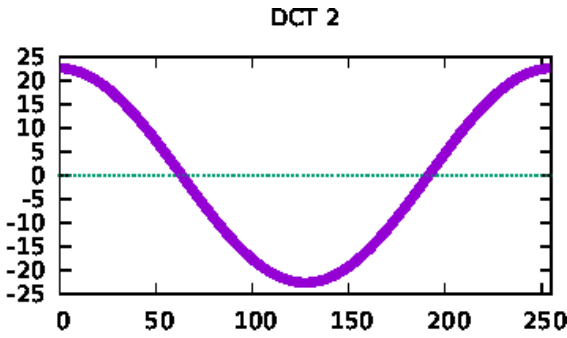


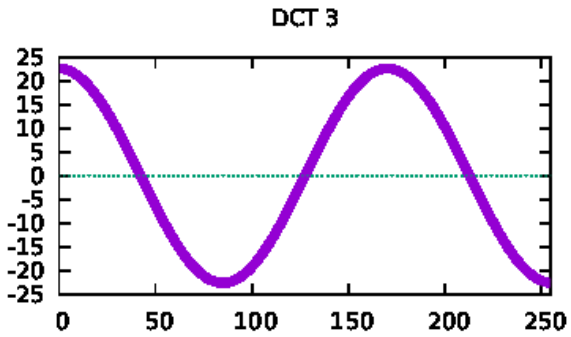


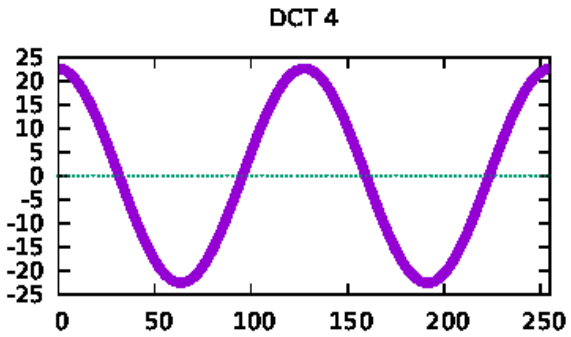
1-dimensional DCT basis

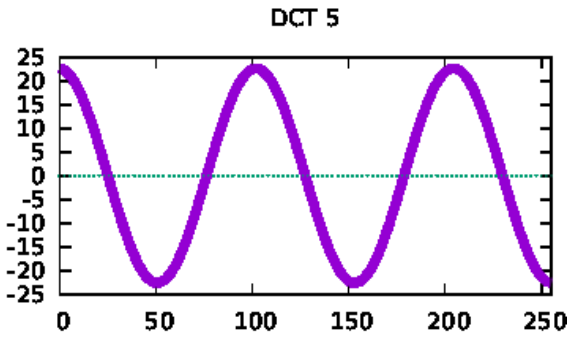


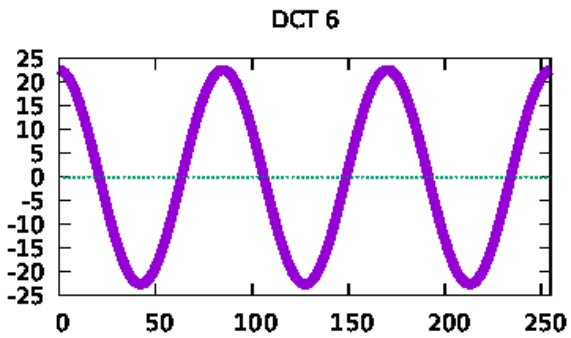


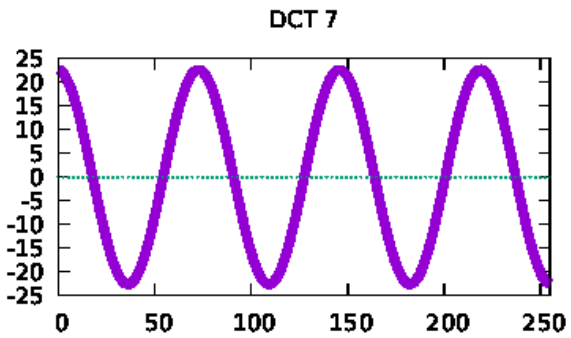


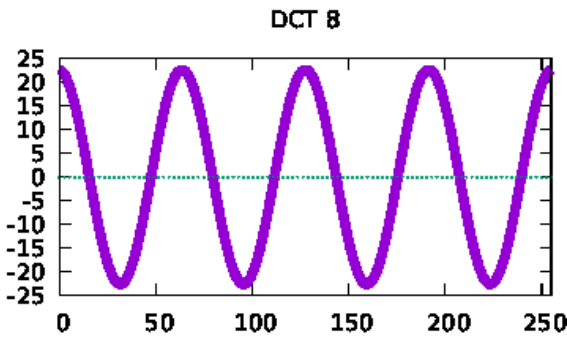




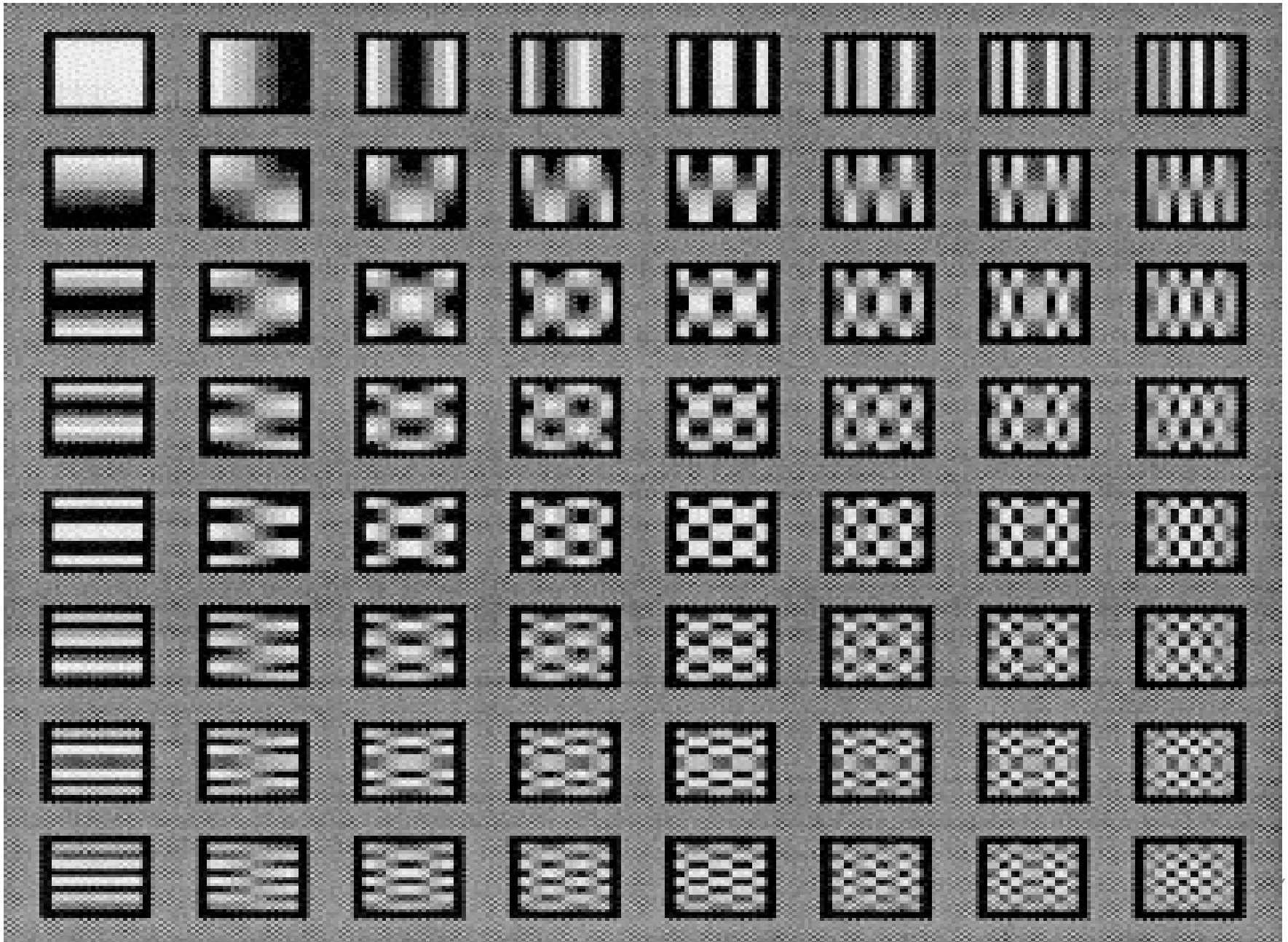








2-dimensional DCT2 basis



Experience the power of the DCT2 compression

```
clear;
 %x=imread('baboon.bmp');
x=imread('person.bmp');
%Transform the image into gray (for many applications gray is
    sufficient)
y= rgb2gray(x);
y=im2double(y);     %from unsigned integer (0-255) to double
    (0-1)
%This is necessary for image transforms.
%The 2-dimensional DCT
ydct=dct2(y);
subplot(2,1,2), imshow(ycomp);
```

```
figure(1);
subplot(2,1,1), imshow(y);
subplot(2,1,2), imshow(ydct);
size1=size(ydct);
ksize1=size1(1)*size1(2)
%This gives the total number of coefficients before compression
%We take only a small portion of the DCT coefficients (K*K)
K=60;    %We can play with K and see the result each time
yzeros=zeros(size1(1),size1(2));
ydct1=yzeros;
ydct1(1:K,1:K)=ydct(1:K,1:K);
%The size reduction (compression%)
compression=100-100*(K*K)/ksize1
ycomp=idct2(ydct1);      %Inverse DCT2
figure(2);
subplot(2,1,1), imshow(y);
```

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 neighbouring context. This is why JPEG images look choppy ("blocky") when a high *compression ratio* is specified by the user.

Quantization

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

- $F(u, v)$ represents a DCT coefficient, $Q(u, v)$ is a "quantization matrix" entry, and $\hat{F}(u,v)$ 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

Is there only One quantization matrix standard?

No…!!, different digital image capturing devices may use different matrices:

http://www.impulseadventure.com/photo/jpeg-quantization.html

Extremely useful site

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

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

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

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

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. 9.2: JPEG compression for a smooth 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

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

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

The Quality Factor

- For most current JPEG encoders, we can choose a quality factor from 1 to 100.

- The method of controlling the compression quality is by scaling the quantization table. For example,

$$F_q(u,v) = \text{round}(F(u,v)^* \text{ (quality)} / Q(u,v))$$

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*.

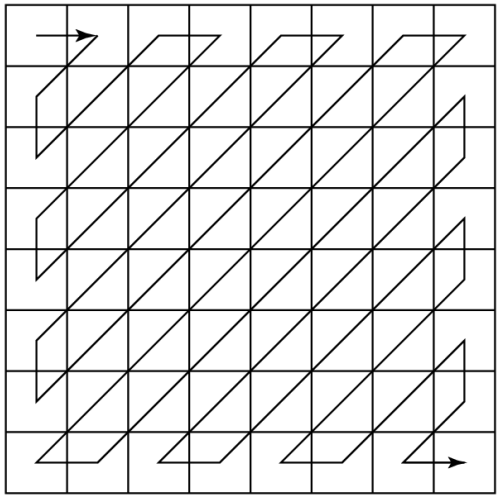
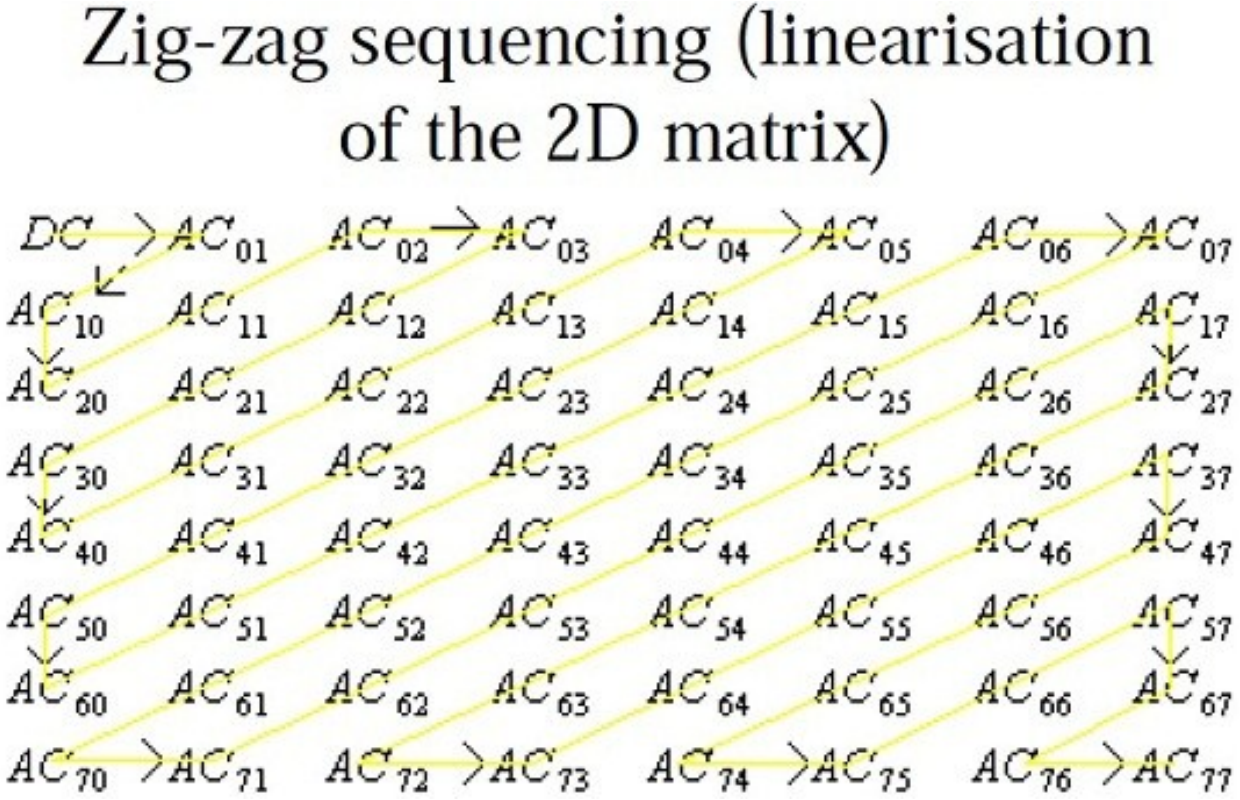


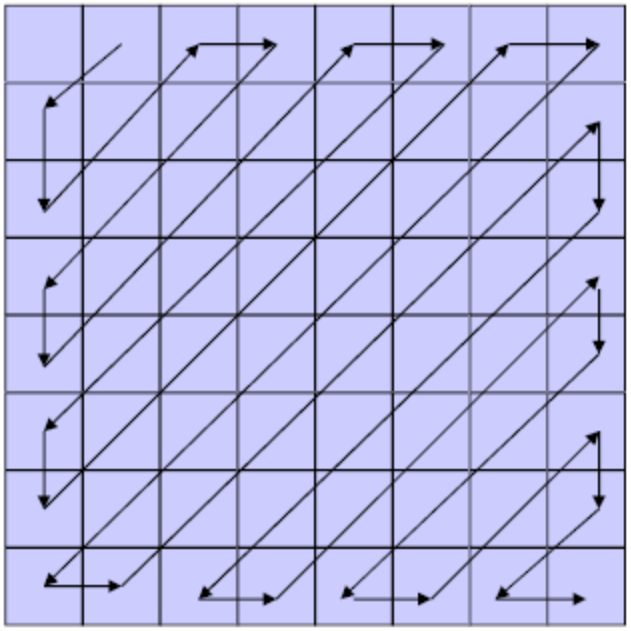
Fig. 9.4: Zig-Zag Scan in JPEG.

Additional View of Zig-Zig Sequencing



AC Coefficients Encoding

- AC coefficients are not differentially encoded.
- Instead, we first do run-length coding.



Zig-zag scanning

16	11	10	16	0	0	0	0
-12	-12	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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

The run length code:

(0,11), (0,-12), (1, -12),
(0, 10), (0,16), (0, 0)

Indicates all are 0 from here

Example

qdct =

2	5	0	-2	0	-1	0	0
9	1	-1	2	0	1	0	0
14	1	-1	0	-1	0	0	0
3	-1	-1	-1	0	0	0	0
2	-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

Run-length symbol representation:
{2,(0,5),(0,9),(0,14),(0,1),(1,-2),(0,-1),(0,1),(0,3),(0,2),(0,-1),(0,-1),(0,2),(1,-1),(2,-1), (0,-1), (4,-1),(0,-1),(0,1),EOB}

EOB: End of block, one of the symbol that is assigned a short Huffman codeword

AC Coefficients Encoding (cont)

- For AC coefficients, we now have a bunch of symbols like

  (zero-run length, value)

- The "value" has large number of possible values in [-1023, 1023]. Direct Huffman coding is infeasible.
- Instead, we generate symbols like

  (zero-run length, size, value)

  Huffman coded 1's complement

More details on AC

After quantization, most of the AC coefficients will be zero; thus, only the nonzero AC coefficients need to be coded.

AC ; This range is divided into 10 size categories.

Each AC coefficient can be described by the pair

(***run/size***, ***amplitude***).

From this pair of values, only the first (the ***run/size***) is Huffman coded.

Assume an AC coefficient is preceded by **six zeros** and has a value of **-18**.

-18 falls into category 5.

The one's complement of -18 is 01101.

Hence, this coefficient is represented by (6/5, 01101).

The pair (6/5) is Huffman coded, and the 5-bit value of-18 is appended to that code.

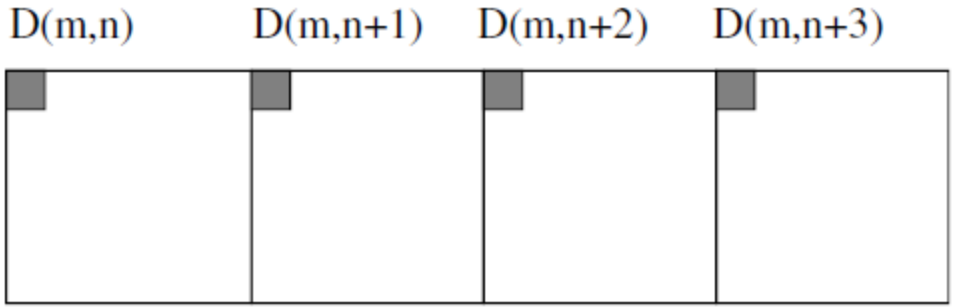
If the Huffman codeword for (6/5) is 1101, then the codeword for –18 is 110101101.

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 $d_0 = DC_0$.

DC Coefficients Encoding



We compute the difference of each two successive quantized coefficients

$$
\begin{aligned}
D(m,n) - 0 &\Rightarrow d(m,n) \\
D(m,n+1) - D(m,n) &\Rightarrow d(m,n+1) \\
D(m,n+2) - D(m,n+1) &\Rightarrow d(m,n+2) \\
D(m,n+3) - D(m,n+2) &\Rightarrow d(m,n+3)
\end{aligned}
$$

DC coefficients coding 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.

DC Coefficients Encoding

- Then, we encode each DC difference value by (size, coefficient)

1	-1, 1
2	-3, -2, 2, 3
3	-7 …-4, 4, …,7
4	
…	
11	-2047,…,-1024,1024,…,2047

The Size Table

7 will be coded as (3, 7)

- The first number 3 is Huffman coded.
- The second 7 is encoded as 1's complement 111. (-7 will be 000)

Can we encode the dc difference value directly?

Coding of Quantized DCT Coefficients

- DC coefficient: Predictive coding
  - The DC value of the current block is predicted from that of the previous block, and the error is coded using Huffman coding
- AC Coefficients: Runlength coding
  - Many high frequency AC coefficients are zero after first few low-frequency coefficients
  - Runlength Representation:
    - Ordering coefficients in the zig-zag order
    - Specify how many zeros before a non-zero value
    - Each symbol=(length-of-zero, non-zero-value)
  - Code all possible symbols using Huffman coding
    - More frequently appearing symbols are given shorter codewords
    - For more details on the actual coding table, see Handout (Sec.8.5.3 in [Gonzalez02]
- One can use default Huffman tables or specify its own tables.
- Instead of Huffman coding, arithmetic coding can be used to achieve higher coding efficiency at an added complexity.

Coding of DC Symbols

- Example:
  - Current quantized DC index: 2
  - Previous block DC index: 4
  - Prediction error: -2
  - The prediction error is coded in two parts:
    - Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code using a Huffman code (JPEG Default DC Code)
      - DC= -2 is in category "2", with a codeword "100"
    - Which position it is in that category, using a fixed length code, length=category number
      - "-2" is the number 1 (starting from 0) in category 2, with a fixed length code of "01".
      - The overall codeword is "10001"

JPEG Tables for Coding DC

TABLE 8.17 JPEG coefficient coding categories.

Range	DC Difference Category	AC Category
0	0	N/A
1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

TABLE 8.18 JPEG default DC code (luminance).

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Coding of AC Coefficients

- Example:
  - First symbol (0,5)
    - The value '5' is represented in two parts:
    - Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code the "(runlength, category)" using a Huffman code (JPEG Default AC Code)
      - AC=5 is in category "3",
      - Symbol (0,3) has codeword "100"
    - Which position it is in that category, using a fixed length code, length=category number
      - "5" is the number 5 (starting from 0) in category 3, with a fixed length code of "101".
      - The overall codeword for (0,5) is "100101"
  - Second symbol (0,9)
    - '9' in category '4', (0,4) has codeword '1011','9' is number 9 in category 4 with codeword '1001' -> overall codeword for (0,9) is '10111001'
  - ETC

JPEG Tables for Coding AC (Run,Category) Symbols

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	1111111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

TABLE 8.19 JPEG default AC code (luminance) *(continues on next page).*

Entropy Coding Introduction

Huffman Coding

LZ Coding

Arithmetic Coding

(www.image.ee.cityu.edu.hk/~kbchan/thesis)

(1) Huffman Coding :-

(*) In 1952 Huffman demonstrated a means for constructing compact (efficient) variable-length codes, and his method remains the most commonly used today (JPEG and MPEG standards). http://www.impulseadventure.com/photo/jpeg-huffman-coding.html

(*) The construction of a Huffman code involves 2 stages :- (i) source reduction
(ii) codeword construction

(i) source reduction stage :-

(*) The two least probable symbols are combined, leaving an alphbet with one less symbol. The combined symbol has a probability = sum of probs. of the 2 original symbols. The source reduction process continues.

2 least probable symbols, and create symbol with summed probability. Eventually, there'll be only 2 symbols, and the source reduction stage is complete.

ex:-

source	$P(s_i)$	$I(s_i)$
S_1	0.3	1.737
S_2	0.25	2
S_3	0.2	2.232
S_4	0.15	2.737
S_5	0.1	3.322

* The symbols are arranged in order of priority. So, the 1st step is to combine the 2 lowest symbols.
* In each step we'll designate a combined symbol by combining the suffixes of its components, e.g., S_4 and $S_5 \Rightarrow S_{45}$
* After each combination, the new set of symbols is re-ordered according to the new probabilities and we proceed -.

original		step-1		step-2		step-3	
S_1	0.3	S_1	0.3	$S_{3(45)}$	0.45	S_{12}	0.55
S_2	0.25	S_2	0.25	S_1	0.3	$S_{3(45)}$	0.45
S_3	0.2	S_{45}	0.25	S_2	0.25		
S_4	0.15	S_3	0.2				
S_5	0.1						

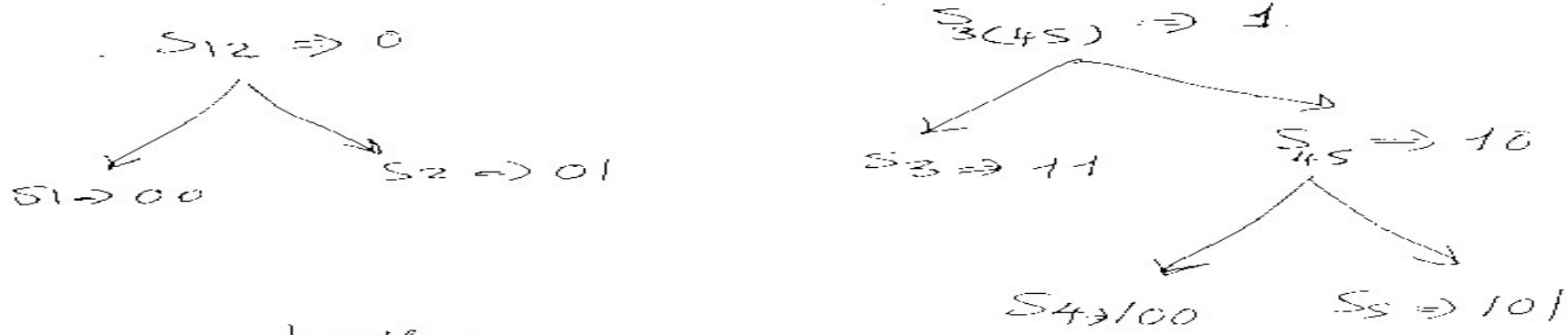
Source reduction is performed until only 2 symbols remain, in this case S_{12}, $S_{3(45)}$.

At this stage, we do stage-2 ⇒ code allocation ⇒ With the remaining 2 symbols, we allocate the codes 0 and 1.

Conventionally, we use 0 to represent the higher priority and 1 the lower priority.

⊛ Now, we split each composite symbol, by appending 0 and 1 to the code of the composite symbol, and we continue this process until all the composite symbols

have been reduced to the only two symbols of the alphabet. We now have a code for each member of the alphabet :-



So, we have :-

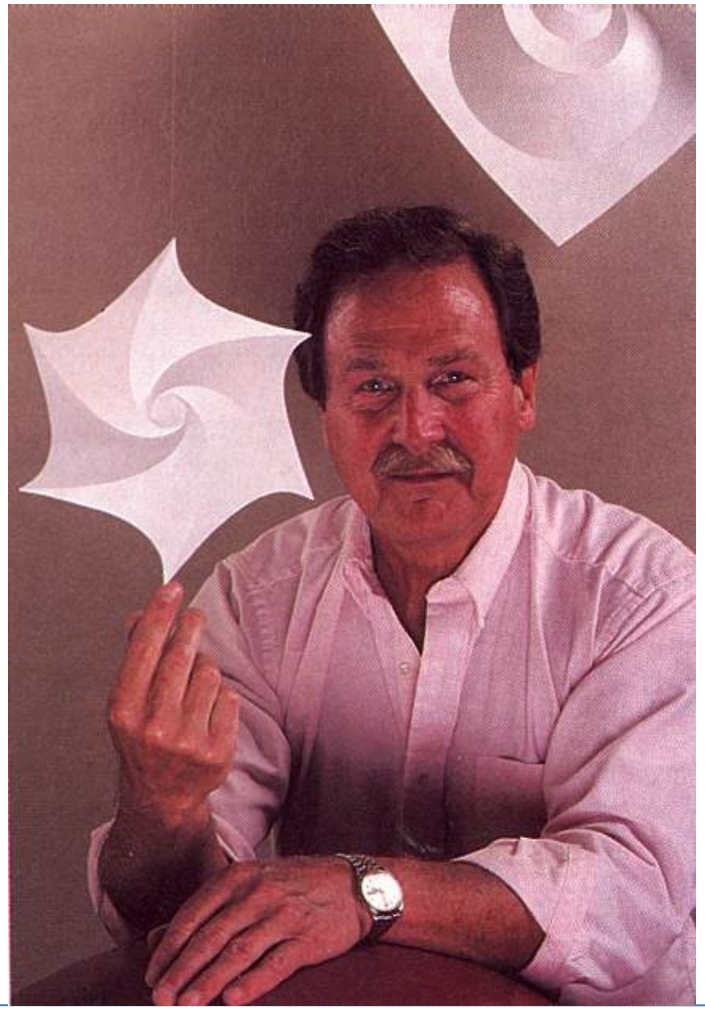
S1	0 0
S2	0 1
S3	1 1
S4	1 0 0
S5	1 0 1

⊛ This is not the only possible Huffman code for this source (since 0, 1 could be interchanged).

⊛ It can be shown that it is not possible to generate a code that is both uniquely decodable and more efficient than Huffman.

Huffman Encoding

- **Huffman coding** is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman while he was a Ph.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".



(D. Huffman)

Dangers of Variable-length coding :-

* A properly constructed variable-length code is efficient because it is derived from the probability distribution of the symbols.
* This is fine, provided we do know the probabilities and that the source is stationary (i.e., the symbol probs. don't change).
* What if the probs. change ?

Returning to the 4-symbol alphabet example with (0.7, 0.1, 0.1, 0.1) prob.
we know we can code this source with a fixed length code of 2 bits/symbol, irrespective of the probability distribution of the alphabet.

* When we used a variable-length code, we were able to get a 1.5 bits/symbol.
* If the probs change to (0.1, 0.1, 0.1, 0.7), and used the same variable-length code, and for a typical group of 10 symbols,

we get a 2.7 bits/symbol, but (1.5)
worse than the simple fixed-length code.

⊛ Thus, if we are sure of the probability distribution of a source, a variable-length code can offer substantial advantages. If the assumed probabilities are wrong, or the source distribution changes, the variable length code can do more harm than good!

(1.2) Modified Huffman codes :-

⊛ In Image processing, we usually have typical ranges (0 → 255)
(-254 → 255)
(-510 → 511)
} quantized values

⊛ If the probability distribution of each symbol (quantized value) is known, it is possible to construct a Huffman code for any number of values. However

Basic Video Compression Techniques

Introduction to Video Compression

- A video consists of a time-ordered sequence of frames, i.e., images.

- An obvious solution to video compression would be predictive coding based on previous frames.

  Compression proceeds by subtracting images: subtract in time order and code the residual error.

- It can be done even better by searching for just the right parts of the image to subtract from the previous frame.

Video Compression with Motion Compensation

- Consecutive frames in a video are similar — temporal redundancy exists.

- **Temporal redundancy** is exploited so that not every frame of the video needs to be coded independently as a new image.

  The difference between the current frame and other frame(s) in the sequence will be coded — small values and low entropy, good for compression.

- Steps of Video compression based on *Motion Compensation (MC)*:

  1. Motion Estimation (motion vector search).
  2. MC-based Prediction.
  3. Derivation of the prediction error, i.e., the difference.

Motion Compensation

- Each image is divided into *macroblocks* of size N x N.
  - By default, N = 16 for luminance images. For chrominance images, N = 8 if 4:2:0 chroma subsampling is adopted.
- Motion compensation is performed at the macroblock level.
  - The current image frame is referred to as *Target Frame*.
  - A match is sought between the macroblock in the Target Frame and the most similar macroblock in previous and/or future frame(s) (referred to as *Reference frame(s)*).
  - The displacement of the reference macroblock to the target macroblock is called a *motion vector* $\mathbf{MV}$.
  - Figure 10.1 shows the case of *forward prediction* in which the Reference frame is taken to be a previous frame.

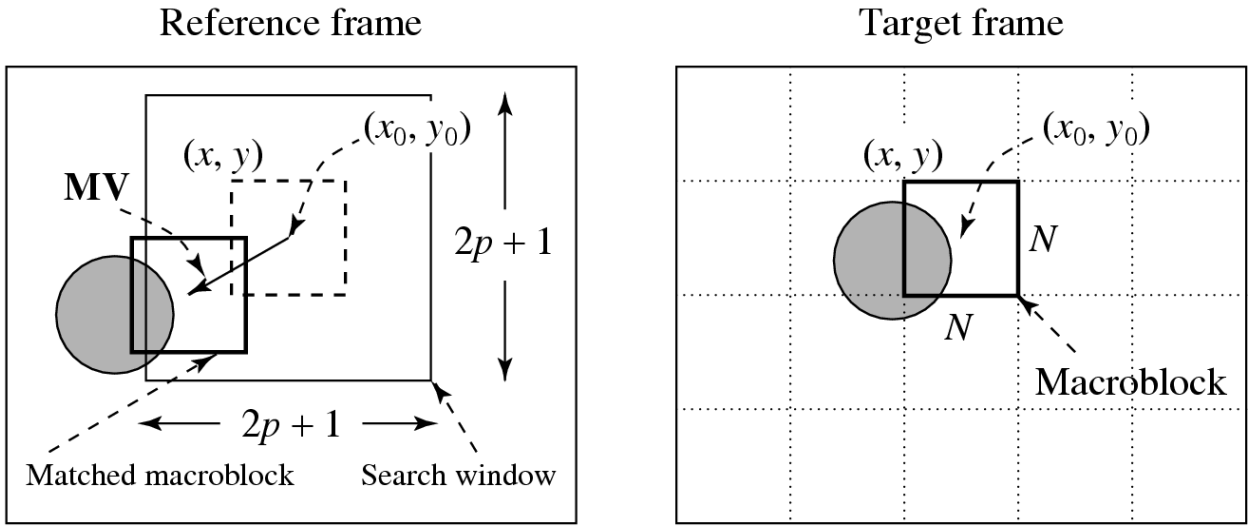


Fig. 10.1: Macroblocks and Motion Vector in Video Compression.

- MV search is usually limited to a small immediate neighborhood — both horizontal and vertical displacements in the range $[-p, p]$.

  This makes a search window of size $(2p + 1) \times (2p + 1)$.

Search for Motion Vectors

- The difference between two macroblocks can then be measured by their *Mean Absolute Difference (MAD)*:

$$MAD(i,j) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\left|C(x+k,y+l) - R(x+i+k,y+j+l)\right| \quad (10.1)$$

N — size of the macroblock,

k and l — indices for pixels in the macroblock,

i and j — horizontal and vertical displacements,

$C(x+k, y+l)$ — pixels in macroblock in Target frame,

$R(x+i+k, y+j+l)$ — pixels in macroblock in Reference frame.

- The goal of the search is to find a vector (i, j) as the motion vector $\mathbf{MV} = (\mathbf{u}, \mathbf{v})$, such that $MAD(i, j)$ is minimum:

$$(u,v) = \left[\ (i,j)\ |MAD(i,j)\ \textit{is minimum},\ i \in [-p,p], j \in [-p,p]\right] \quad (10.2)$$

Sequential Search

- **Sequential search**: sequentially search the whole $(2p + 1)$ x $(2p + 1)$ window in the Reference frame (also referred to as Full search).
  - a macroblock centered at each of the positions within the window is compared to the macroblock in the Target frame pixel by pixel and their respective *MAD* is then derived using Eq. (10.1).
  - The vector (i, j) that offers the least MAD is designated as the $\mathbf{MV}$ (u, v) for the macroblock in the Target frame.

Hierarchical Search

- The search can benefit from a hierarchical (Wavelet: multiresolution) approach in which initial estimation of the motion vector can be obtained from images with a significantly reduced resolution.
- Figure 10.3: a three-level hierarchical search in which the original image is at Level 0, images at Levels 1 and 2 are obtained by down-sampling from the previous levels by a factor of 2, and the initial search is conducted at Level 2.

  Since the size of the macroblock is smaller and p can also be proportionally reduced, the number of operations required is greatly reduced.

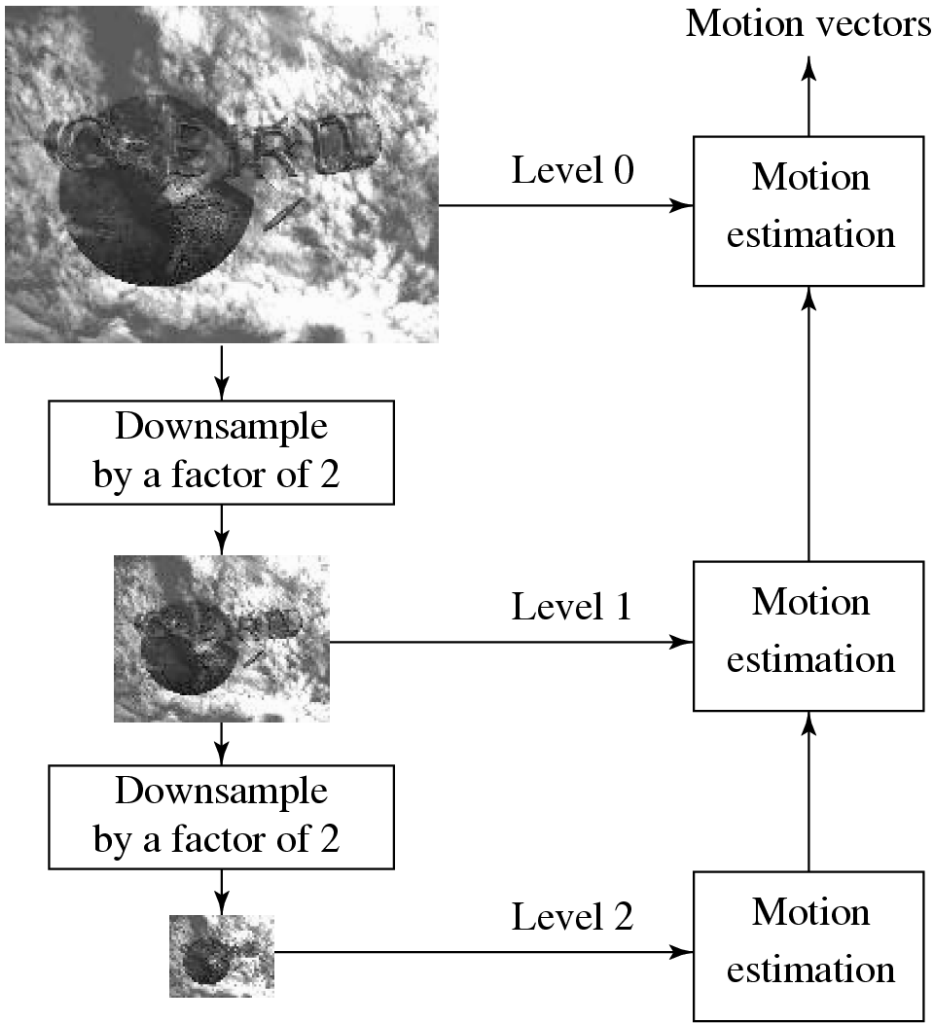


Fig. 10.3: A Three-level Hierarchical Search for Motion Vectors.

Table 10.1 Comparison of Computational Cost of Motion Vector Search based on examples

Search Method	OPS_per_second for 720×480 at 30 fps	
	$p = 15$	$p = 7$
Sequential search	29.89×10^9	7.00×10^9
2D Logarithmic search	1.25×10^9	0.78×10^9
3-level Hierarchical search	0.51×10^9	0.40×10^9

10.4 H.261

- **H.261**: An earlier digital video compression standard, its principle of MC-based compression is retained in all later video compression standards.
  - The standard was designed for videophone, video conferencing and other audiovisual services over ISDN.
  - The video codec supports bit-rates of p x 64 kbps, where p ranges from 1 to 30 (Hence also known as p * 64).
  - Require that the delay of the video encoder be less than 150 msec so that the video can be used for real-time bidirectional video conferencing.

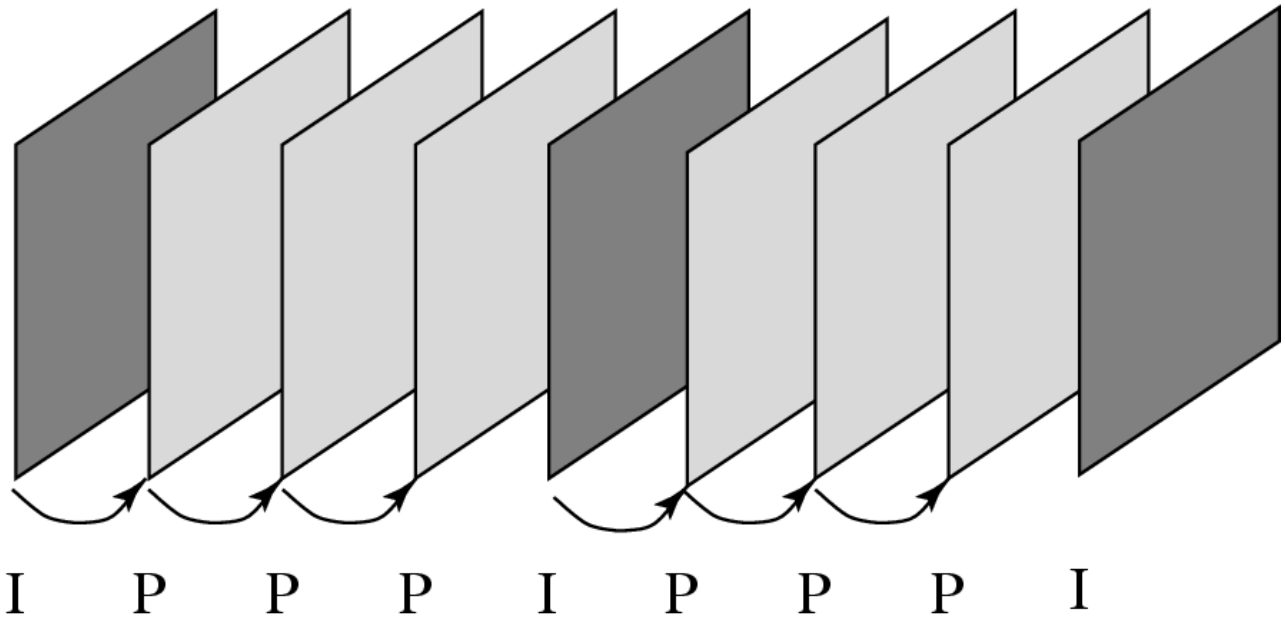


Fig. 10.4: H.261 Frame Sequence.

H.261 Frame Sequence

- Two types of image frames are defined: Intra-frames (**I-frames**) and Inter-frames (**P-frames**):
  - I-frames are treated as independent images. Transform coding method similar to JPEG is applied within each I-frame, hence "Intra".
  - P-frames are not independent: coded by a forward predictive coding method (prediction from a previous P-frame is allowed — not just from a previous I-frame).
  - **Temporal redundancy removal** is included in P-frame coding, whereas I-frame coding performs only **spatial redundancy removal**.
  - To avoid propagation of coding errors, an I-frame is usually sent a couple of times in each second of the video.

- Motion vectors in H.261 are always measured in units of full pixel and they have a limited range of ± 15 pixels, i.e., $p = 15$.

Intra-frame (I-frame) Coding

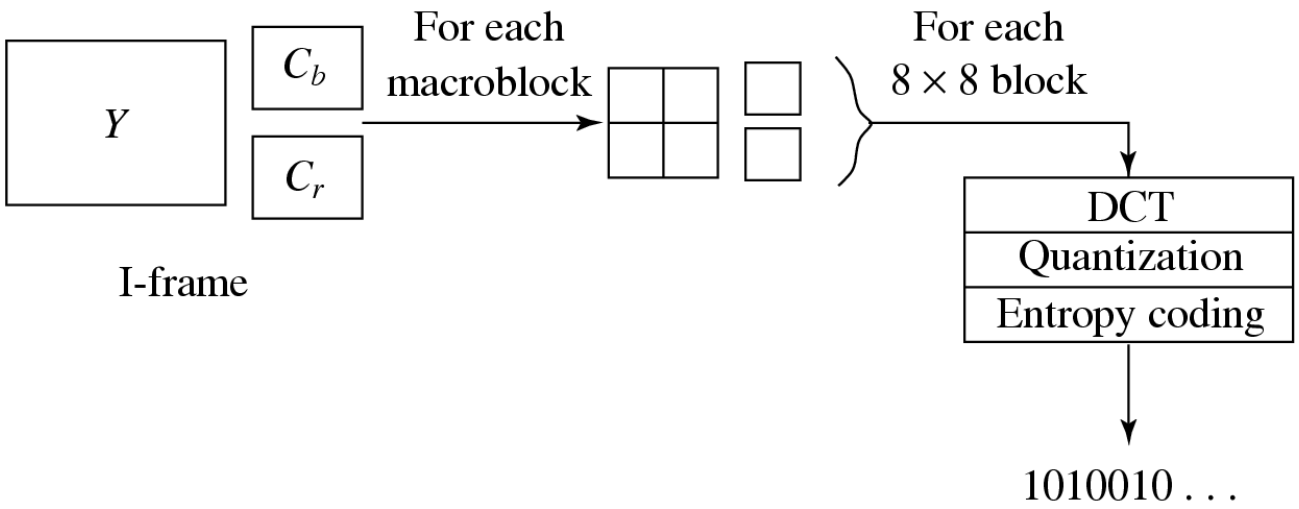
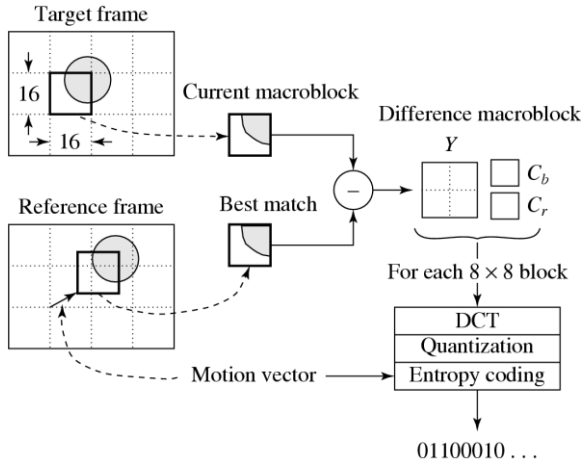


Fig. 10.5: I-frame Coding.

- **Macroblocks** are of size 16 x 16 pixels for the Y frame, and 8 x 8 for Cb and Cr frames, since 4:2:0 chroma subsampling is employed. A macroblock consists of four Y, one Cb, and one Cr 8 x 8 blocks.
- For each 8 x 8 block a DCT transform is applied, the DCT coefficients then go through quantization zigzag scan and entropy coding.

Inter-frame (P-frame) Predictive Coding

- Figure 10.6 shows the H.261 P-frame coding scheme based on motion compensation:



- For each macroblock in the Target frame, a motion vector is allocated by one of the search methods discussed earlier.

- After the prediction, a *difference macroblock* is derived to measure the *prediction error*.

- Each of these 8 x 8 blocks go through DCT, quantization, zigzag scan and entropy coding procedures.

- The P-frame coding encodes the difference macroblock (not the Target macroblock itself).

- Sometimes, a good match cannot be found, i.e., the prediction error exceeds a certain acceptable level.
  - The MB itself is then encoded (treated as an Intra MB) and in this case it is termed a *non-motion compensated MB*.

- For a motion vector, the difference **MVD** is sent for entropy coding:

$$\mathbf{MVD} = \mathbf{MV}_{\mathbf{Preceding}} - \mathbf{MV}_{\mathbf{Current}} \qquad (10.3)$$

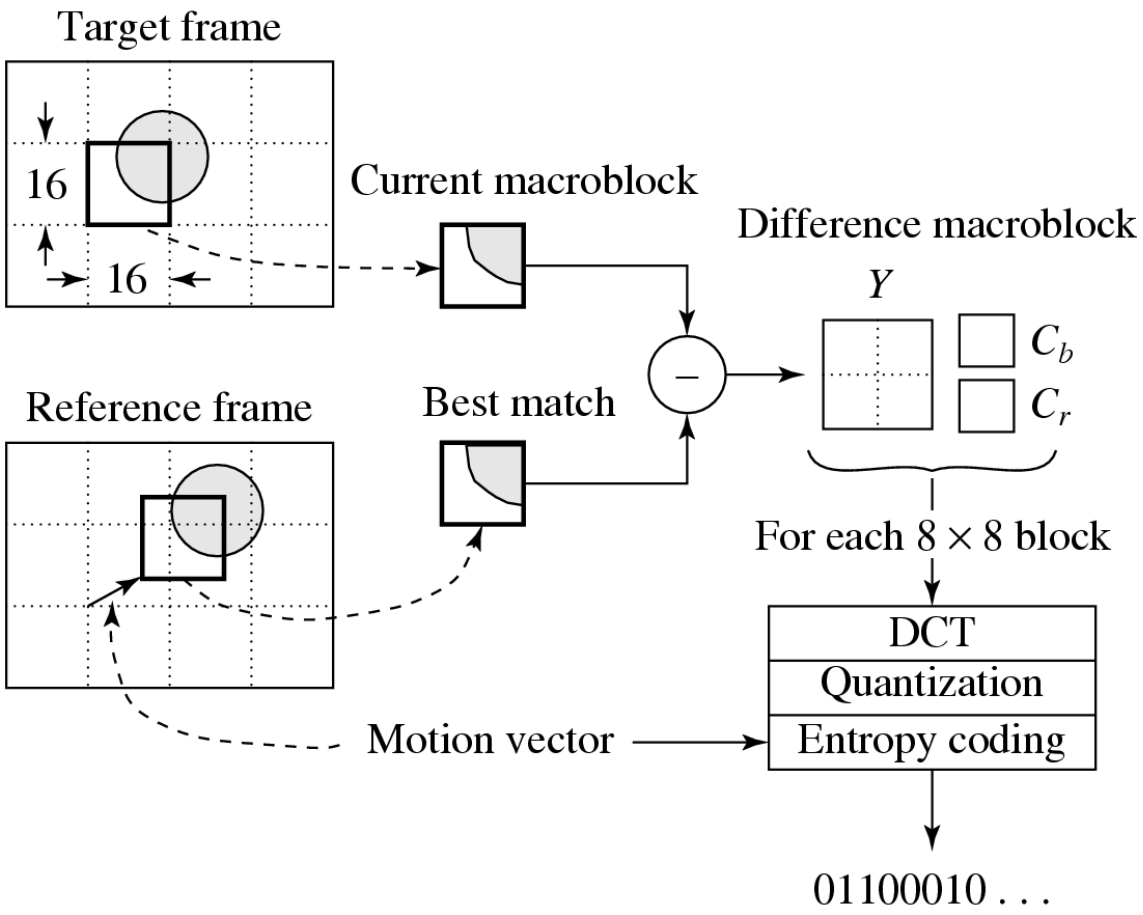


Fig. 10.6: H.261 P-frame Coding Based on Motion Compensation.

Quantization in H.261

- The quantization in H.261 uses a constant *step_size*, for all DCT coefficients within a macroblock (with the DC different)
- If we use *DCT* and *QDCT* to denote the DCT coefficients before and after the quantization, then for DC coefficients:

$$QDCT = round\left(\frac{DCT}{step\_size}\right) = round\left(\frac{DCT}{8}\right) \qquad (10.4)$$

for all other coefficients (AC coefficients):

$$QDCT = \left\lfloor \frac{DCT}{step\_size} \right\rfloor = \left\lfloor \frac{DCT}{2 * scale} \right\rfloor \qquad (10.5)$$

scale — an integer in the range of [1, 31] and is adaptive to control the bit rate and the quality.

H.261 Encoder and Decoder

- Fig. 10.7 shows a relatively complete picture of how the H.261 encoder and decoder work.

  A scenario is used where frames I, P_1, and P_2 are encoded and then decoded.

- Note: decoded frames (not the original frames) are used as reference frames in motion estimation.
- The

- The data that goes through the observation points indicated by the circled numbers are summarized in Tables 10.3 and 10.4.