JPEG and H.26x Standards

- Video Data Size and Bit Rate
- DCT Transform and Quantization
- JPEG Standard for Still Image
- Intra-frame and Inter-frame Compression
- Block-based Motion Compensation
- H.261 Standard for Video Compression
- H.263, H.263+, H.263++, H.26L, H.264

Video Bit Rate Calculation

width ~ pixels (160, 320, 640, 720, 1280, 1920, …)

height ~ pixels (120, 240, 480, 485, 720, 1080, …)

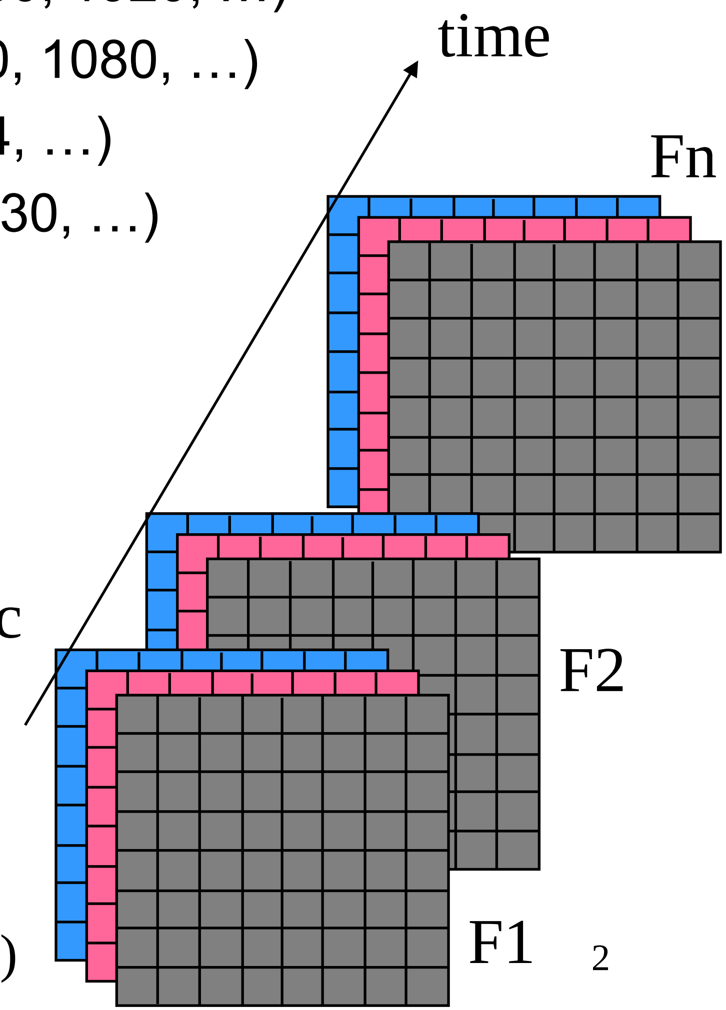
depth ~ bits per pixel (1, 4, 8, 15, 16, 24, …)

fps ~ frames per second (5, 15, 20, 24, 30, …)

compression factor (1 ~ 100 ~)

$$\left[\frac{\text{width * height * depth * fps}}{\text{compression factor}} \right] = \text{bits/sec}$$

bps



One Frame =
3 pictures (YCrCb)

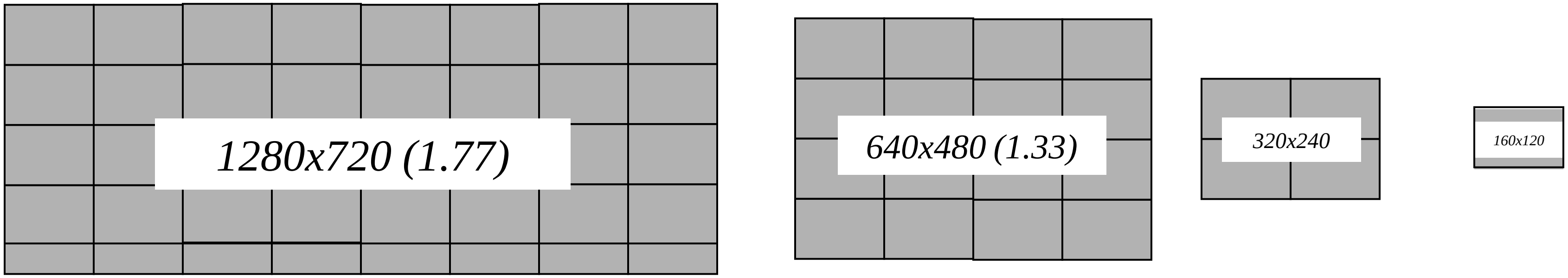
Uncompressed Video Data Size

compression factor = 1

Size of uncompressed video in gigabytes

	1920x1080	*1280x720*	*640x480*	*320x240*	*160x120*
1 sec	0.19	0.08	0.03	0.01	0.00
1 min	11.20	4.98	1.66	0.41	0.10
1 hour	671.85	298.60	99.53	24.88	6.22
1000 hours	671,846.40	298,598.40	99,532.80	24,883.20	6,220.80

Image size of video



Effects of Compression

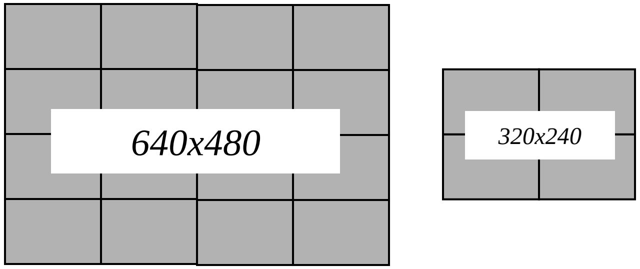
storage for 1 hour of compressed video in megabytes

Compression ration

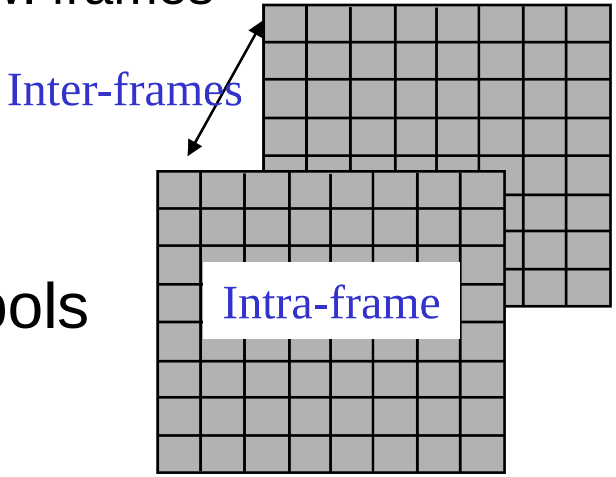
	1920x1080	*1280x720*	*640x480*	*320x240*	*160x120*
1:1	671,846	298,598	99,533	24,883	6,221
3:1	223,949	99,533	33,178	8,294	2,074
6:1	111,974	49,766	16,589	4,147	1,037
25:1	26,874	11,944	3,981	995	249
100:1	6,718	2,986	995	249	62

3 bytes/pixel, 30 frames/sec

Coding Overview

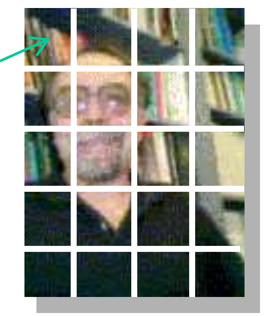


- Digitize
  - Subsample to reduce data
- Compression algorithms exploit:
  - Spatial redundancy - correlation between neighboring pixels
    - **Intra-frame** compression
    - remove redundancy within frame
  - Temporal redundancy - correlation betw. frames
    - **Inter-frame** compression
    - Remove redundancy between frames
- Symbol Coding
  - Efficient coding of sequence of symbols
    - RLC (Run Length Coding)
    - Huffman coding

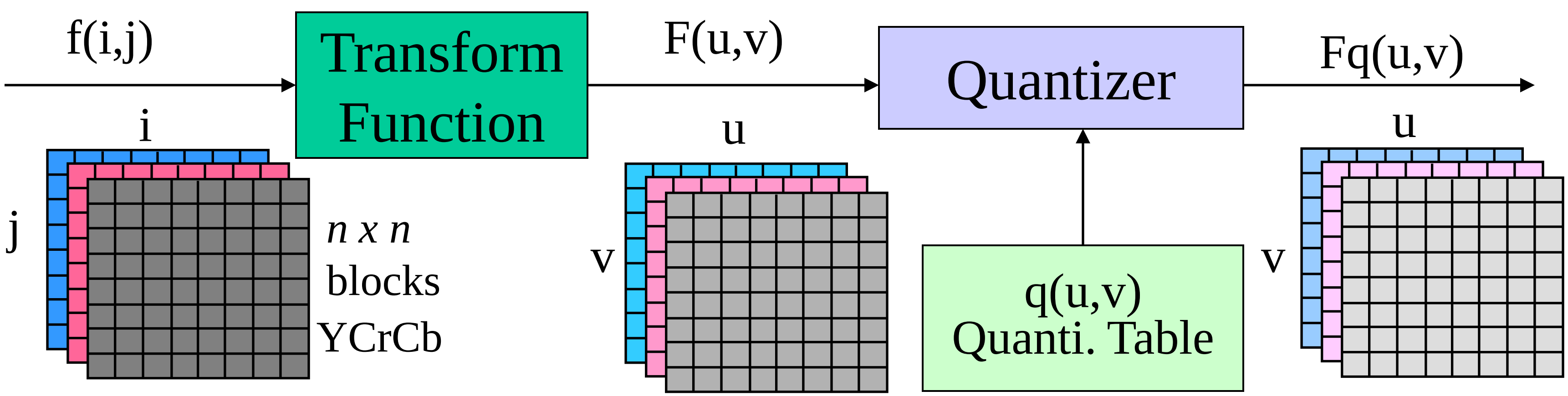


Transform Coding

N x M image



- An image conversion process that transforms an image from the spatial domain to the frequency domain.
- Subdivide an individual *N x M* image into small *n x n* **blocks**
- Each *n x n* block undergoes a reversible transformation
- Basic approach:
  - De-correlate the original block - radiant energy is redistributed amongst only a small number of transform coefficients
  - Discard many of the low energy coefficients (through quantization)



DCT – *nxn* Discrete Cosine Transform

$$\boldsymbol{F} = \boldsymbol{D} \ \boldsymbol{x} \ \boldsymbol{f} \qquad \textit{F, D, f are n-by-n matrixes}$$

$$F[u,v] = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right]$$

where

$$C(w) = \begin{cases} \dfrac{1}{\sqrt{2}} & \text{for } w=0 \\ 1 & \text{for } w=1,2,\ldots,n-1 \end{cases}$$

- IDCT is very similar
- 8x8 DCT coefficients

0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
1	0.8	0.6	0.2	-0.2	-0.6	0.8	-1
0.9	0.4	-0.4	-0.9	-0.9	-0.4	0.4	0.9
0.8	-0.2	-1	-0.6	0.6	1	0.2	-0.8
0.7	-0.7	-0.7	0.7	0.7	-0.7	-0.7	0.7
0.6	-1	0.2	0.8	-0.8	-0.2	1	-0.6
0.4	-0.9	0.9	-0.4	-0.4	0.9	-0.9	0.4
0.2	-0.6	0.8	-1	1	-0.8	0.6	-0.2

Quantization

- Purpose of quantization
  - Achieve high compression by representing DCT coefficients with no greater precision than necessary
  - Discard information which is not visually significant
- After output from the FDCT, each of the 64 DCT coefficients is quantized
  - Many-to-one-mapping => fundamentally lossy process
  - $F_q[u,v]$ = Round (F[u,v] / q[u,v])
  - Example: F[u,v] =101101 = 45 (6 bits).
    If q[u,v] = 4, truncate to 4 bits, $F_q[u,v]$ =1011

Example:
2x2 block

$$F[u,v] = \begin{bmatrix} 45 & 12 \\ 8 & 3 \end{bmatrix} \qquad Q[u,v] = \begin{bmatrix} 4 & 6 \\ 6 & 8 \end{bmatrix} \qquad F_q[u,v] = \begin{bmatrix} 11 & 2 \\ 1 & 0 \end{bmatrix}$$

- Quantization is the principal source of lossiness in DCT-based encoders
- Uniform quantization: each F[u,v] is divided by the same constant *N*
- Non-uniform quantization: use quantization tables from psycovisual experiments to exploit the limit of human visual system

DCT and Quantization Example

DC component, others called AC

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(a) source image samples **f**

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

(b) forward DCT coefficients **F**

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

(c) quantization table **Q**

15	0	-1	0	0	0	0	0
-2	-1	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
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(d) normalized quantized coefficients **Fq**

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	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

(e) denormalized quantized coefficients $\mathbf{F}^{-1}$

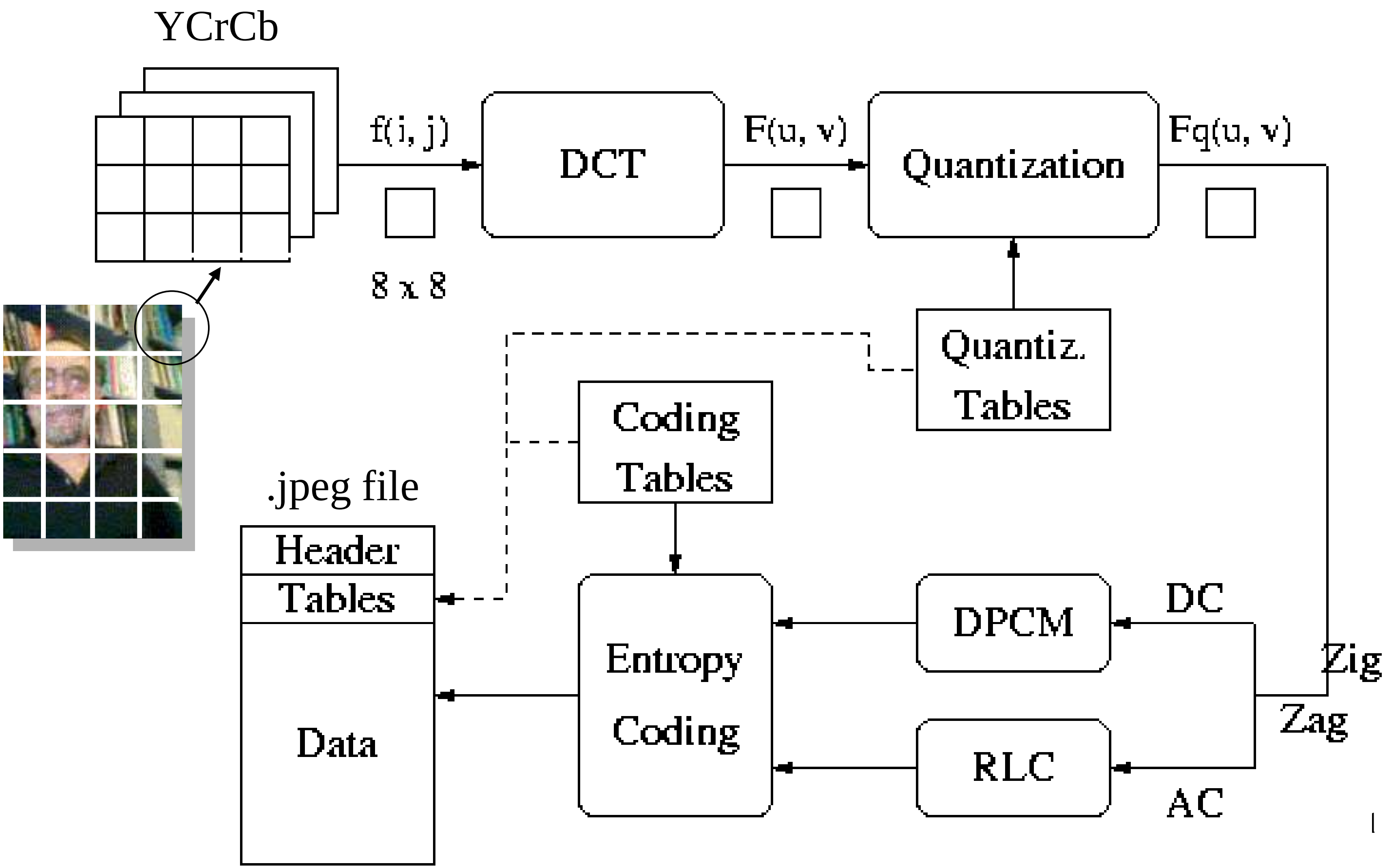
144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

(f) reconstructed image samples $\mathbf{f}^{-1}$

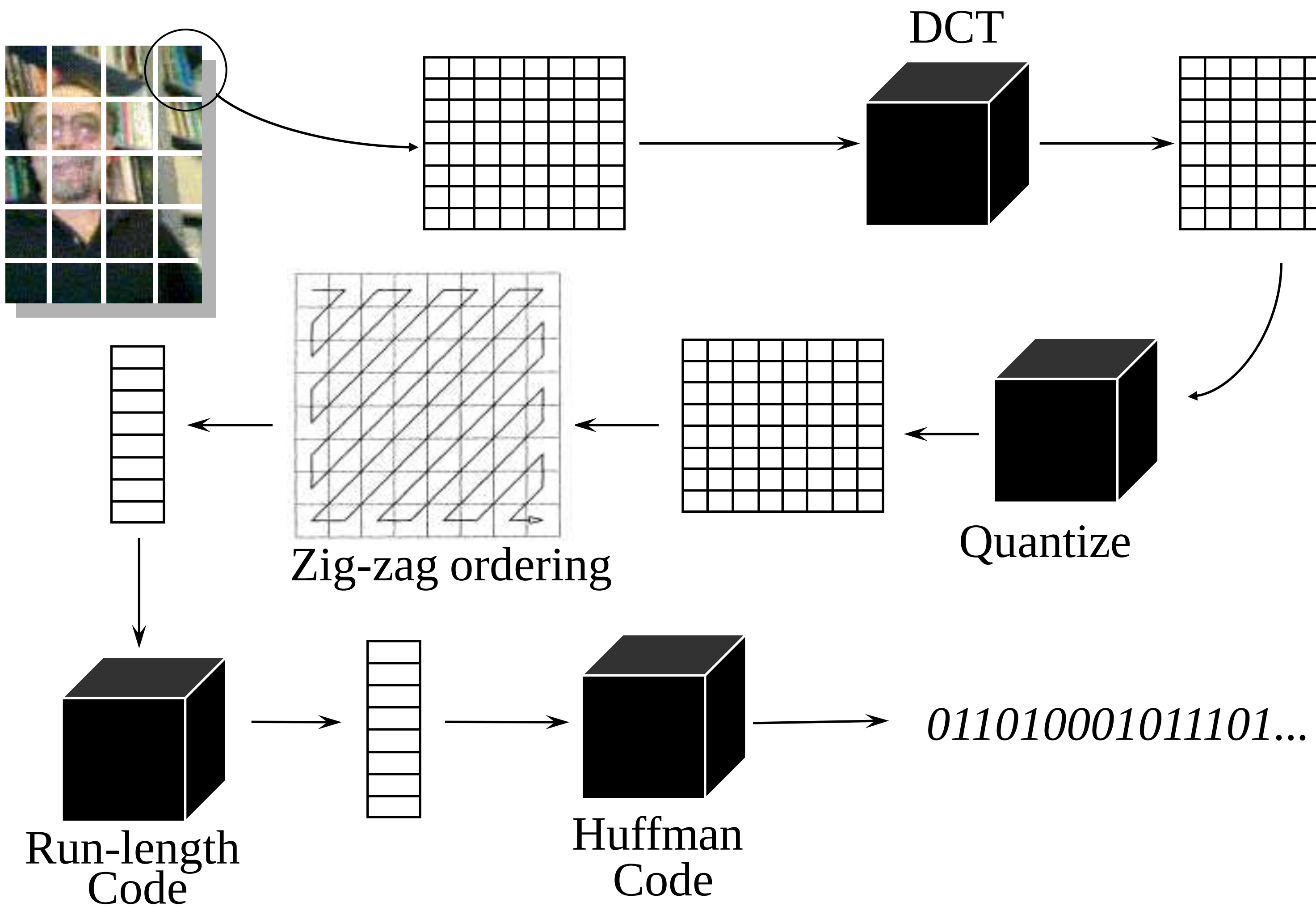
JPEG Image Compression Standard

- Mainly for still image (gray and color)
- Four Modes:
  - Lossless JPEG
  - Sequential (Baseline) JPEG
  - Progressive JPEG
  - Hierarchical JPEG
- Hybrid Coding Techniques:
  - DCT Coding
  - Run Length Encoding(RLE)
  - Huffman Coding
  - Linear Prediction (only in lossless mode)
- New Standard: JPEG2000
- Motion JPEG for video

Overview of Baseline JPEG



Block Transform Encoding



Example of Block Encoding

Quantize Table

$$\begin{bmatrix} 15 & 10 & 10 & 16 \\ 10 & 16 & 12 & 16 \\ 10 & 8 & 16 & 16 \\ 16 & 12 & 16 & 32 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 139 & 144 & 149 & 153 \\ 144 & 151 & 153 & 156 \\ 150 & 155 & 160 & 163 \\ 159 & 161 & 162 & 160 \end{bmatrix}}_{\text{original image}} \xrightarrow{\text{DCT}} \begin{bmatrix} 1260 & -1 & -12 & -5 \\ -23 & -17 & -6 & -3 \\ -11 & -9 & -2 & 2 \\ -7 & -2 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Quantize}} \begin{bmatrix} 79 & 0 & -1 & 0 \\ -2 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

DC component

AC components

$$\xrightarrow{\text{zigzag}} \begin{bmatrix} 79 & 0 & -2 & -1 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\xrightarrow{\text{run-length code}} \begin{bmatrix} 0 & 79 \\ 1 & -2 \\ 0 & -1 \\ 0 & -1 \\ 0 & -1 \\ 2 & -1 \\ 0 & 0 \end{bmatrix} \xrightarrow{\text{Huffman code}} 10011011100011\ldots$$

coded bitstream < 10 bits (0.55 bits/pixel)

Result of Coding/Decoding

$$\begin{bmatrix} 139 & 144 & 149 & 153 \\ 144 & 151 & 153 & 156 \\ 150 & 155 & 160 & 163 \\ 159 & 161 & 162 & 160 \end{bmatrix}$$

original block

$$\begin{bmatrix} 144 & 146 & 149 & 152 \\ 156 & 150 & 152 & 154 \\ 155 & 156 & 157 & 158 \\ 160 & 161 & 161 & 162 \end{bmatrix}$$

reconstructed block

$$\begin{bmatrix} -5 & -2 & 0 & 1 \\ -4 & 1 & 1 & 2 \\ -5 & -1 & 3 & 5 \\ -1 & 0 & 1 & -2 \end{bmatrix} \Rightarrow$$

Small Loss
Neglect-able

errors

Examples

Uncompressed
(262 KB)

8 bits/pixel

Compressed (50)
(22 KB, 12:1)

0.67 bit/pixel

Compressed (1)
(6 KB, 43:1)

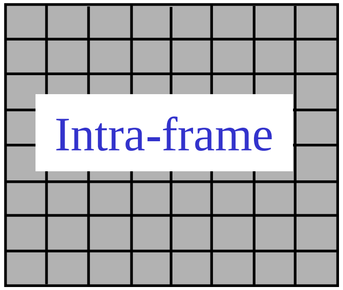
0.17 bit/pixel

JPEG vs. GIF

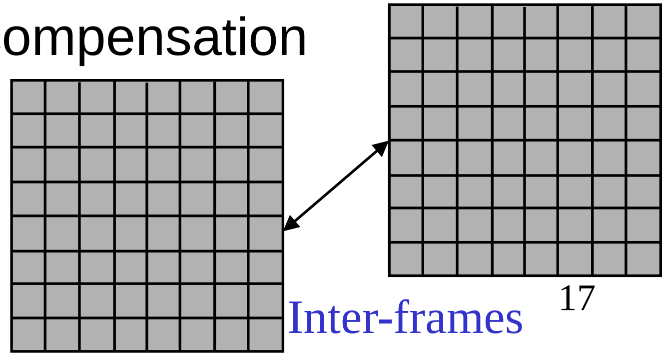
- JPEG Advantages
  - more colors (GIF limited to 256)
  - lossless option
  - best for scanned photographs
  - progressive JPEG downloads rough image before whole image arrives
- GIF Advantages
  - transparent color setting
  - animated GIFs
  - better for flat color fields: clip art, cartoons, etc.
  - interlaced delivery downloads low resolution image before whole image arrives

Intra- vs. Inter-frame Compression

- **Intra-frame compression**
  - For still image like JPEG
  - Exploit the redundancy in image (*spatial redundancy*)
  - Can be applied to individual frames in a video sequence
- Techniques
  - Subsampling (small size)
  - Block transform coding
  - Coarse quantization

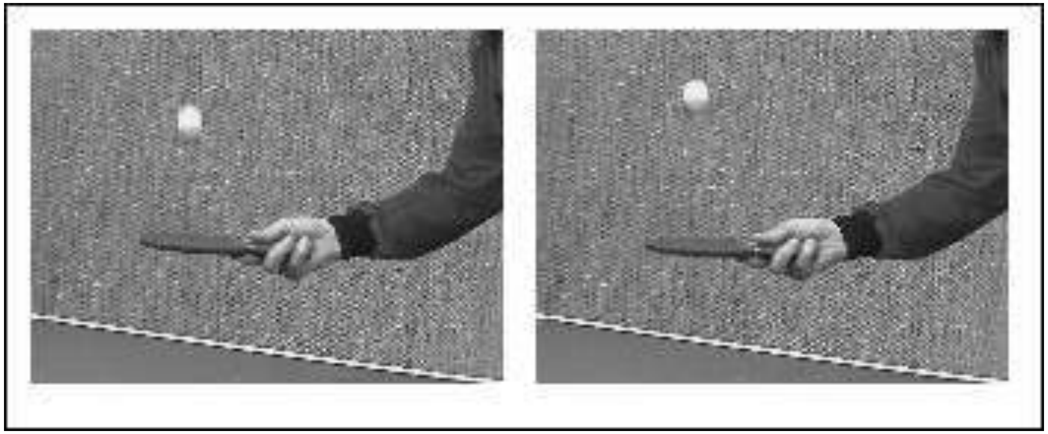


- Intra + **inter-frame compression**
  - For video like H.26x & MPEG
  - Exploit the similarities between successive frames (*temporal redundancy*)
- Techniques
  - Subsampling (small frame rate)
  - Difference coding
  - Block-based difference coding
  - Block-based motion compensation



Difference Coding

- Compare pixels with previous frame
  - Only pixels that have been changed are updated
  - A fraction of the number of pixel values will be recorded



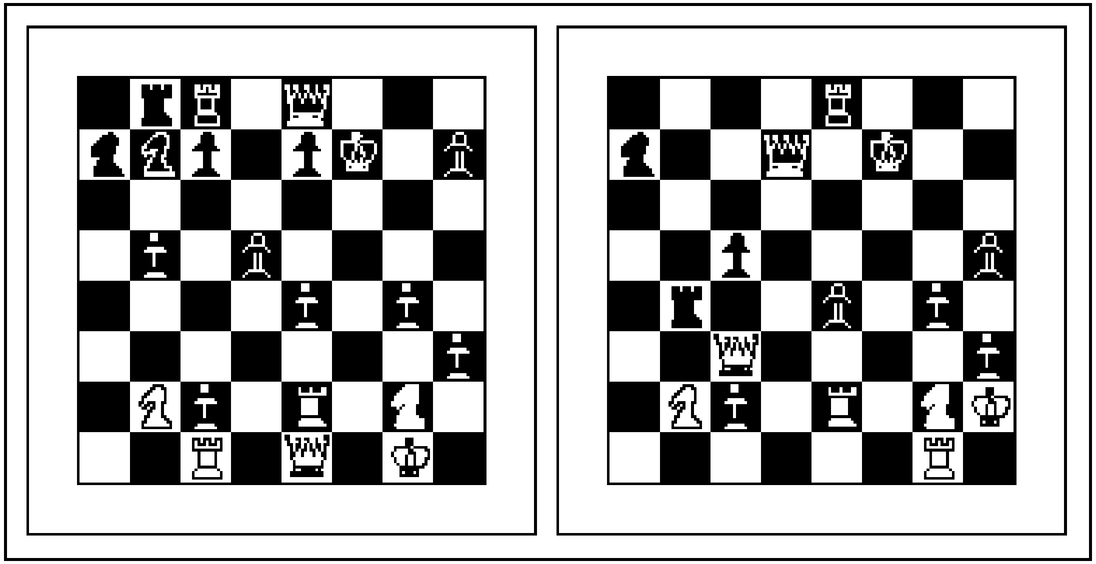
- Overhead associated with which pixels are updated: what if a large number of pixels are changed ?
- Pixels values are slightly different even with no movement of objects: ignore small changes (lossy)

Block-based Difference Coding

- Difference coding *at the block level*
  - Send sequence of blocks rather than frames
  - If previous block similar, skip it or send difference
  - Update a whole block of pixels at once
  - 160 x 120 pixels (19200 pixels) => 8x8 blocks (300 blocks)
  - Possible artifact at the border of blocks
- Limitations of difference coding
  - Useless where there is a lot of motion (few pixels unchanged)
  - What if a camera itself is moving ?
- Need to compensate for object motion

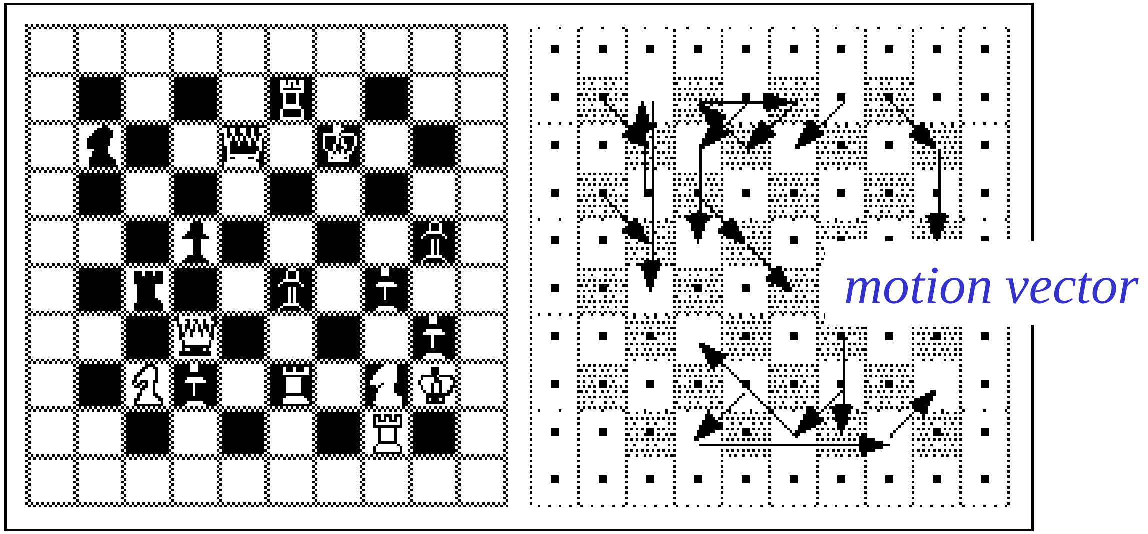
Block-based Motion Compensation

- **Motion compensation** assumes that current frame can be modeled as a translation of a previous frame
- Search around block in previous frame for a better matching block and encode position and error difference

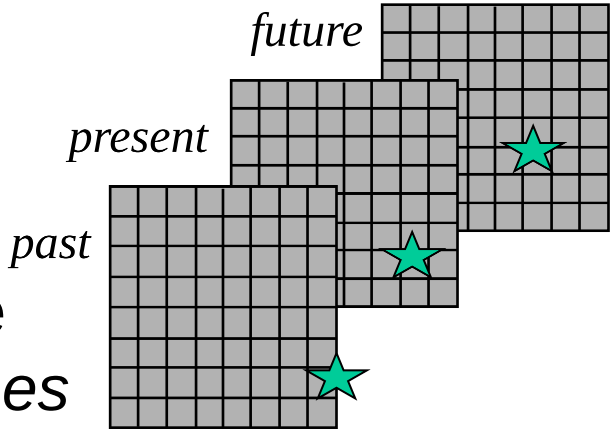


Block-based Motion Compensation

- Current frame is divided into *uniform non-overlapping blocks*
- Each block in the current frame is compared to areas of similar size from the preceding frame in order to find an area that is similar
- The relative difference in locations is known as the *motion vector*
- Because fewer bits are required to code a motion vector than to code actual blocks, compression is achieved.



Bidirectional Motion Compensation



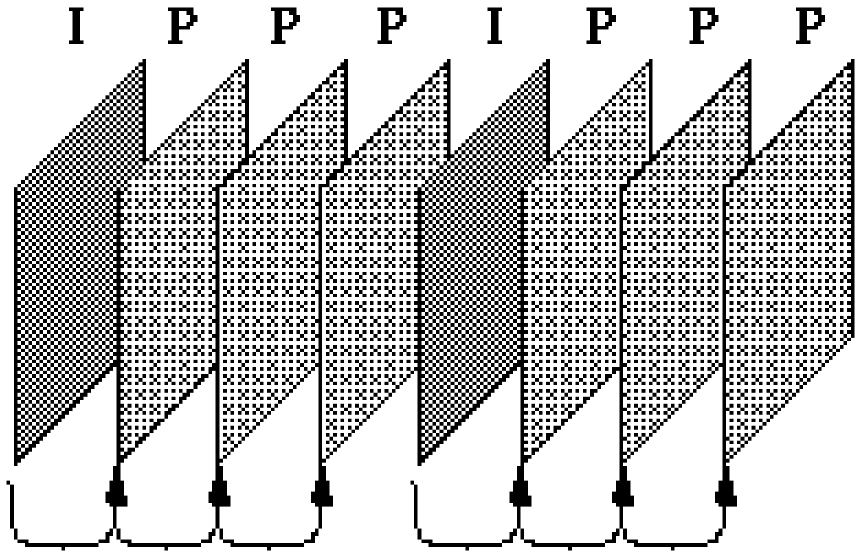
- **Bidirectional motion compensation**
  - Areas just uncovered are not predictable from the past, but can be predicted from the future
  - Search in *both past and future frames*
- Effect of noise and errors can be reduced by averaging between previous and future frames
- Bi-directional interpolation provides a high degree of compression
  - Requires that frames be encoded and transmitted in a different order from which they will be displayed.
- In reality, exact matching is not possible, thus lossy compression

Overview of H.261

- Developed by CCITT (Consultative Committee for International Telephone and Telegraph) in 1988-1990
- Designed for videoconferencing, video-telephone applications over ISDN telephone lines.
  - Bit-rate is p x 64 Kbps, where p ranges from 1 to 30 (2048 kbps)
- Supports CCIR 601 CIF (352 x 288) and QCIF (176 x 144) images with 4:2:0 subsampling.
- Significant influence on H.263, MPEG 1-4, etc.

Frame Sequence of H.261

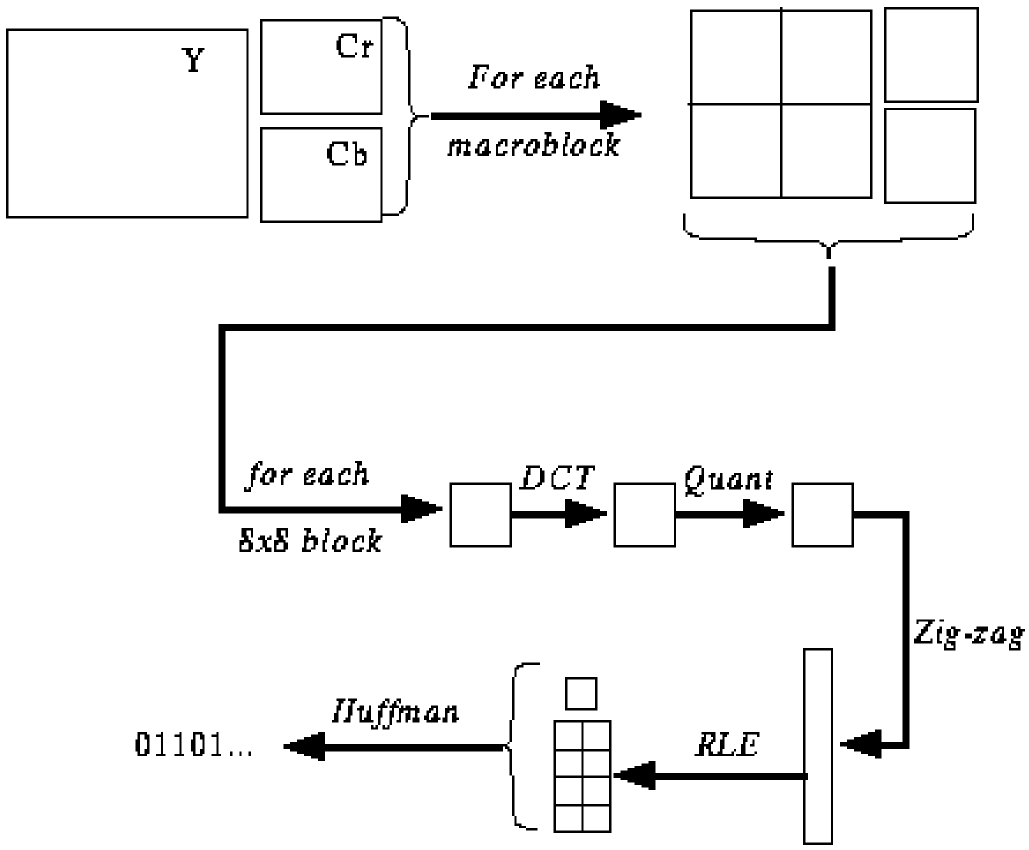
- Two frame types: Intra-frames (***I-frames***) and Inter-frames (***P-frames***): I-frame provides an accessing point, it uses basically JPEG.
- P-frames use **"pseudo-differences"** from previous frame ("predicted"), so frames depend on each other.



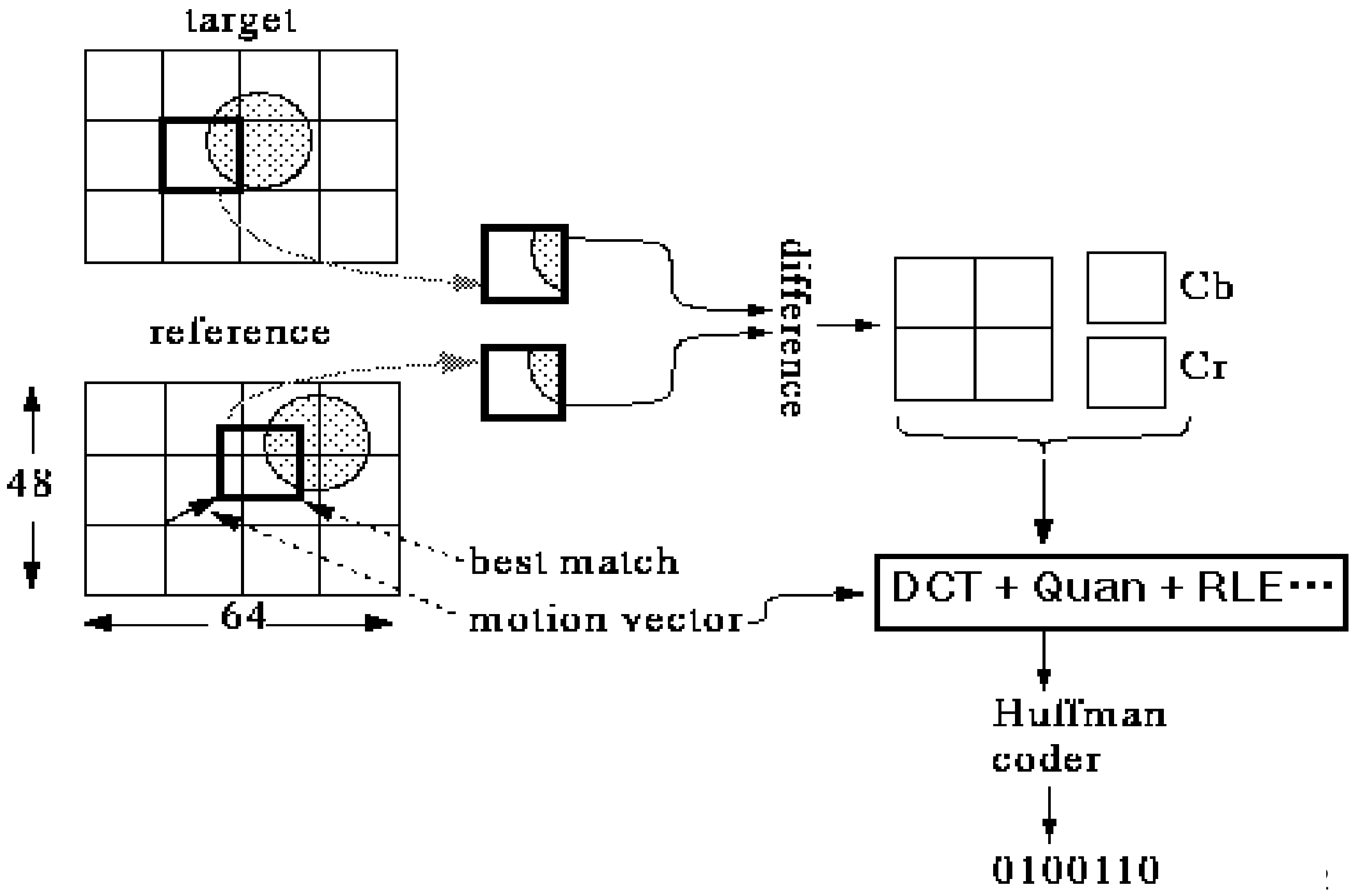
Intra-frame Coding

- Macroblock:
  - 16 x 16 pixel areas on Y plane of original image.
  - Usually consists of 4 Y blocks, 1 Cr block, and 1 Cb block

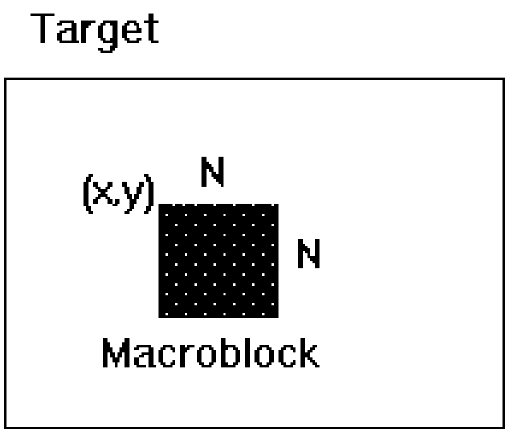
    (4:2:0 or 4:1:1)
- Quantization is by constant value for all DCT coefficients (i.e., no quantization table as in JPEG).



Inter-frame Coding

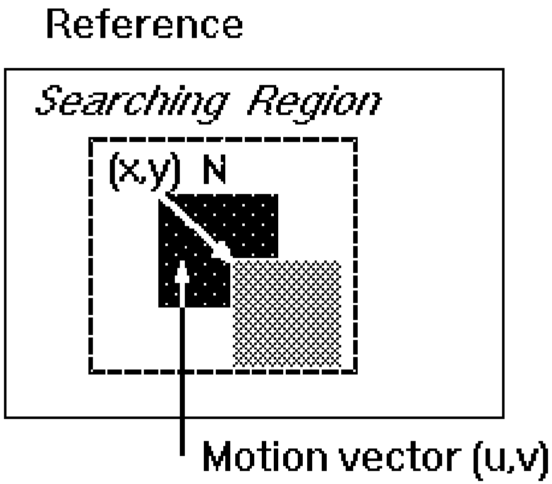


Motion Vector Searches



C(x+k, y+l): macro block pixels in the target
R(x+i+k, y+j+l): macro block pixels in the reference

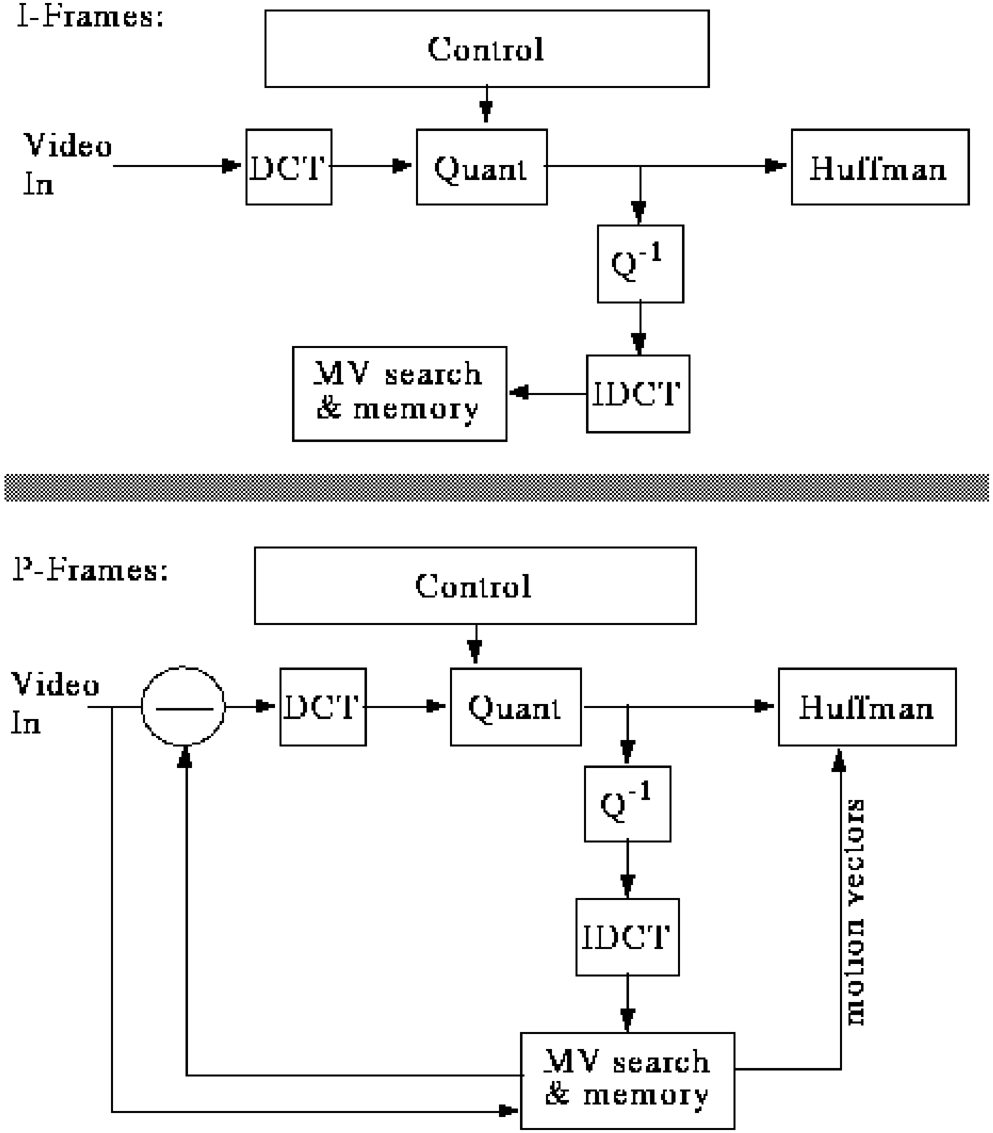
$$MAE(i, j) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\left|C(x+k, y+l) - R(x+i+k, y+j+l)\right|$$



The goal is to find a vector (u, v) such that the mean Absolute Error, *MAE*(u, v) is minimum:

1. Full Search Method
2. Two-dimensional Logarithmic Search
3. Hierarchical Motion Estimation

Encoder



I-Frames:
Control
Video In
DCT
Quant
Huffman
Q-1
IDCT
MV search & memory
P-Frames:
Control
Video In
DCT
Quant
Huffman
Q-1
IDCT
MV search & memory
motion vectors

H.262, H.263 and H.264

- H.262 = MPEG-2 jointly by ITU and ISO/IEC
- ITU-T Rec. H.263 v1 (1995)
  - Current best standard for practical video telecommunication
  - Has overtaken H.261 as videoconferencing codec
  - Superior to H.261 at all bit rates (1/2)
  - Video size: Sub-QCIF (128x96), QCIF (176x144), CIF(352x288), 4CIF(704X576), 16CIF (1408x1152)
  - PB frames mode (bidirectional prediction)
  - 4 motion vector for each block, ½ pixel accuracy
  - Arithmetic coding efficient than Huffman coding in H.261
- H.263 v2 (H.263+, 1997)
- H.263 v3 (H.263++, 2000), H.26L (2002)
- H.264/AVC (now)

Demos of Image GIF and JPEG Coding