LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Lecture Coding Theory

Source Coding

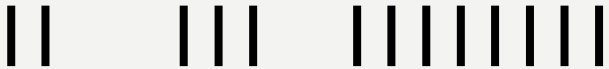
Image and Video Compression

Images: Wikipedia

- Entropy Coding:
  - Unary Coding → Golomb Coding
  - Static Huffman Coding
  - Adaptive Huffman Coding
  - Arithmetic Coding
  - Run Length Encoding (RLE) ➔ e.g. BMP
  - Differential Encoding
- Context Sensitive Coding
  - Lempel-Ziv-*
    - LZ77 ➔ Gameboy
    - LZW (Welch) ➔ GIF, TIFF
    - Deflate ➔ PNG, zlib, zip, etc.
- Lossy Image Coding ➔ JPEG
  - The human eye
  - Color Space Conversion, RGB→YUV YCrCb, Chroma Subsampling
  - Blocks 8x8, Discrete Cosine Transformation (H.264: wavelet transform)
  - Ordering of Coefficients, Differentical Coding of DC color
  - Thresholding
  - Entropy Encoding
  - Effects: low-contrast edges, deblocking filter
- Video Coding: MPEG-2, MPEG-4, H.264
  - Analog video (also for BMP): CRT scan order
  - Prediction: spatial, temporal
  - I-Frames 1e6 bit, P-Frames 3e5 bit, B-Frames 1e5 bit
  - Motion Compensation, Optical Flow, Lukas Canade

Numeral Systems

unary systems



additive systems: (roman: **I, II, III, IV, V, X, C, M, MMXI**)

positional systems

$$a_n a_{n-1} \ldots a_1 a_0 = \sum_i b^i a_i$$

common base: 2, 10, 16

10	2	16
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Example: Numeral Systems

Example: 2017_{10}

$$\underbrace{2 \cdot 10^3}_{2000} + 0 \cdot 10^2 + \underbrace{1 \cdot 10^1}_{10} + 7 \cdot \underbrace{10^0}_{1}$$

2017-1024 = 993-512 = 481-256 = 225-128 = 97-64 = 33

33-32 = 1-1 = 0

binary: **$111\ 1101\ 1110_2$**

$$\underbrace{111}_{7}\ \underbrace{1110}_{14=E}\ \underbrace{0001}_{1}$$

hexadecimal: $7E1_{16}$

2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024
2^{11}	2048
2^{12}	4096

Run Length Encoding

Compresses Repetition of Symbols

e.g. A A A B B B B B ➔ 3x A, 5x B

Special case: binary data {0,1}

0 0 0 1 1 1 1 1 ➔ 3, 5

Disadvantage: compressed data may be longer than original

Soltution: Escape symbol with original data

A A A L M U B B B B B ➔ 3x A, ESC LMU, 5x B

Examples: BMP image format

Windows Bitmap Format (BMP)

Offset	Type	Name	Description
0	uint16_t	bfType	„BM“ = 0x42 0x4D
2	uint32_t	bfSize	Total size in bytes
6	uint32_t	bfReserved	0
10	uint32_t	bfOffBits	Offset to image data (54 if no color table)
14	uint32_t	biSize	40 (sizeof(BITMAPINFOHEADER))
18	uint32_t	biWidth	Width of image
22	uint32_t	biHeight	Height of image (negative: top-down, positive: bottom-up)
26	uint16_t	biPlanes	1
28	uint16_t	biBitCount	Bits per pixel (1,4,8,16,24,32)

Windows Bitmap Format (BMP) – Part 2

Offset	Type	Name	Description
30	uint32_t	biCompression	0: BI_RGB 1: BI_RLE4 2: BI_RLE8 3: BI_BITFIELDS
34	uint32_t	biSizeImage	Size of image data in bytes
38	uint32_t	biXPelsPerMeter	Bits per pixel in X direction (or 0)
42	uint32_t	biYPelsPerMeter	Bits per pixel in Y direction (or 0)
46	uint32_t	biClrUsed	Number of entries in color table (max $2^{biBitCount}$)
50	uint32_t	biClrImportant	Same as biClrUsed

Windows Bitmap Format (BMP) – Part 3

Color Table: biClrUsed entries of form:

Blue, green, red, „0“ as bytes

Image data: 24 bpp ➔ blue, green, red as bytes

1,4,8 bpp ➔ index into color table

Windows Bitmap Format (BMP) – RLE

BI_RLE4, BL_RLE8 in biCompression (offset 30)

Two bytes (a, b) ➔ a times „b“

e.g. A A A B B B B B ➔ 0x03 0x41 0x05 0x42

If a=0:

Value of b	Description
0	End of image line
1	End of image
2	(a‘,b‘): Skip a‘ columns and b‘ rows
3-255	ESC for b bytes (align to 16 bits)

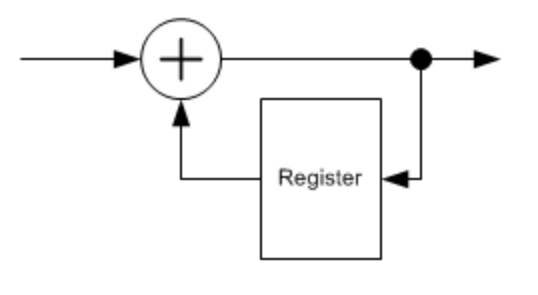
Differential Encoding

Difference between neighboring values

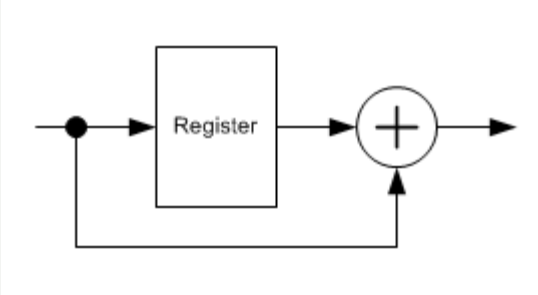
$y_i = y_{i-1} \ominus x_i$

e.g. 1 2 3 4 5 4 3 2 1 ➔ +1 +1 +1 +1 +1 -1 -1 -1 -1

Encoder



Decoder

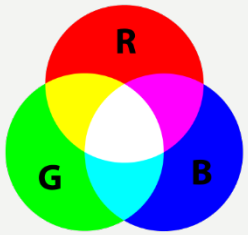


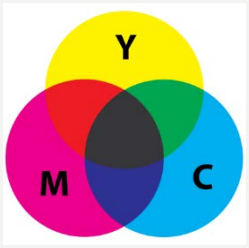
JPEG Encoding

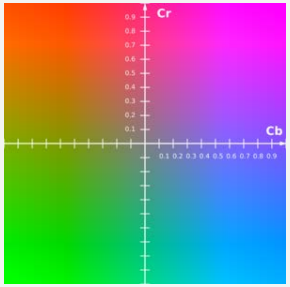
- Conversion RGB → Y′ $C_B C_R$
- Reduction of color resolution
- Discrete Cosine Transformation of 8x8 pixel blocks
- Quantization (lossy)
- Encoding, e.g. Huffman encoding

JPG: Color Spaces

- Color Spaces: e.g. additive, subtractive
- Typical on screen: RGB
- Human eye perceives intensity and color differently



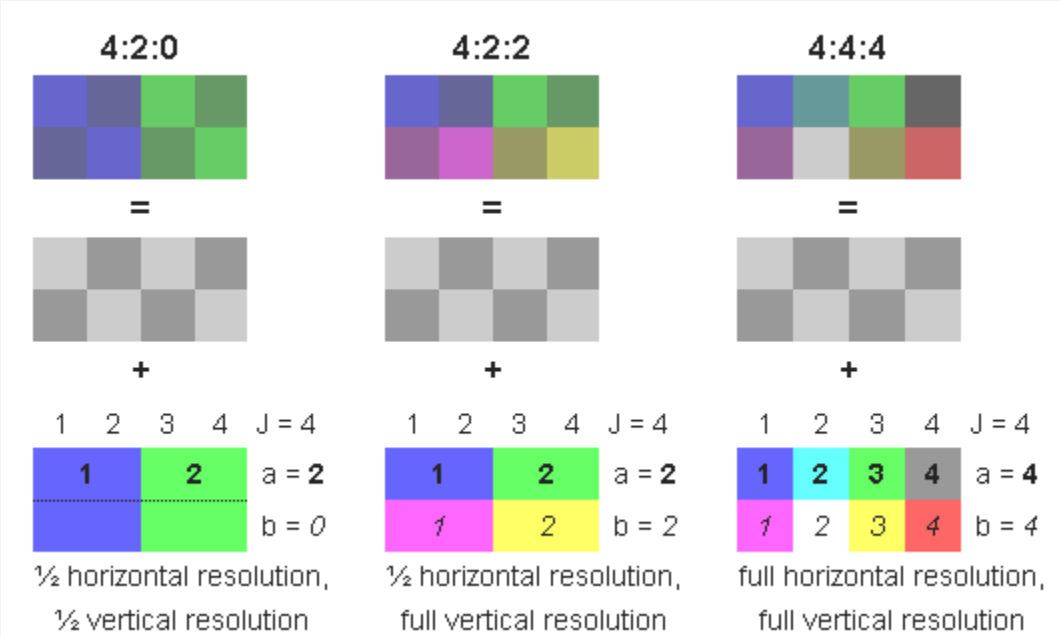




$$
\begin{aligned}
Y' &= 0 + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D) \\
C_B &= 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \cdot B'_D) \\
C_R &= 128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)
\end{aligned}
$$

JPG: Reduction of color resolution

The human eye is much more sensitive about intensity then about color



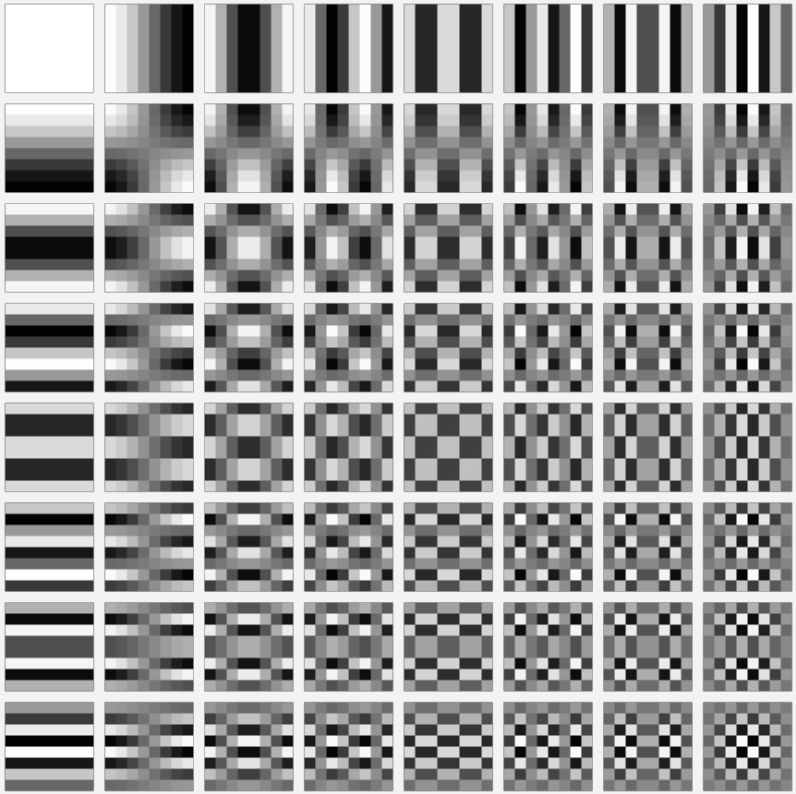
JPG: Block Splitting

The image is split into 8x8 pixel blocks.

4 blocks (2x2) are a macro-block

Discrete Cosine Transform

$$
\begin{aligned}
X_{k_1,k_2} &= \sum_{n_1=0}^{N_1-1}\left(\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right]\right)\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right] \\
&= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]\cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right].
\end{aligned}
$$



JPEG: Quantization

- Lossy step
- Controlled by 8x8 quantization matrix
- Oriented at human perception

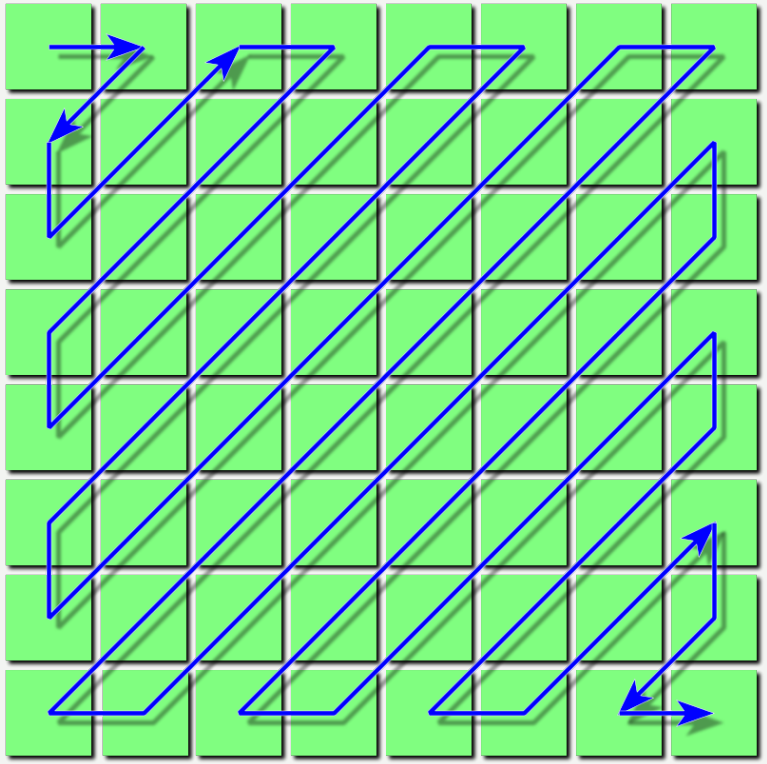
$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

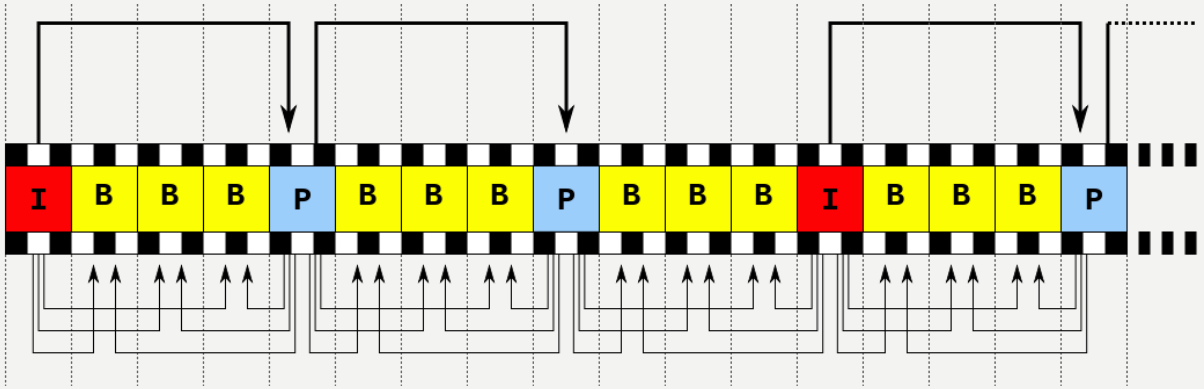
Zig-Zag Sequence

Differential Encoding

Run-Length Encoding



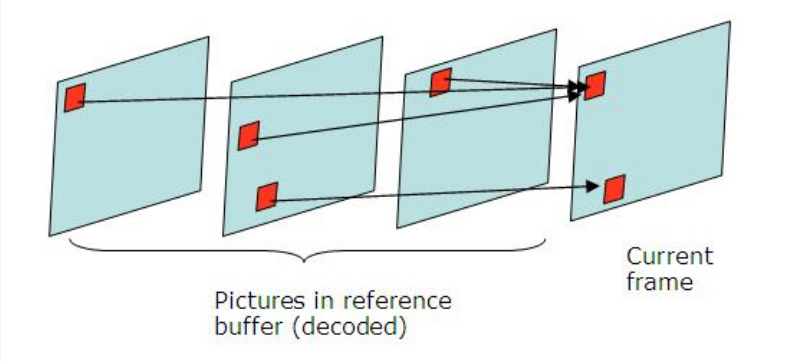
Video Compression:



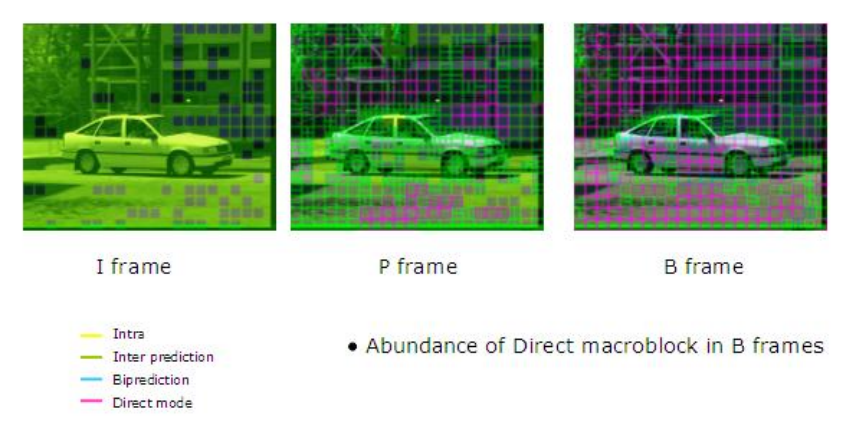
Different Picture Types:

- I-Frame: „inter“ frame, independently encoded image, JPEG style compression, approx. 9×10^5 bit
- P-Frame: „predicted“ frame, is composed of prediction of previous images, approx. 3×10^5 bit
- B-Frame: „bidirectionally predicted“ frame, is composed by interpolating between previous and following images, approx. 1×10^5 bit

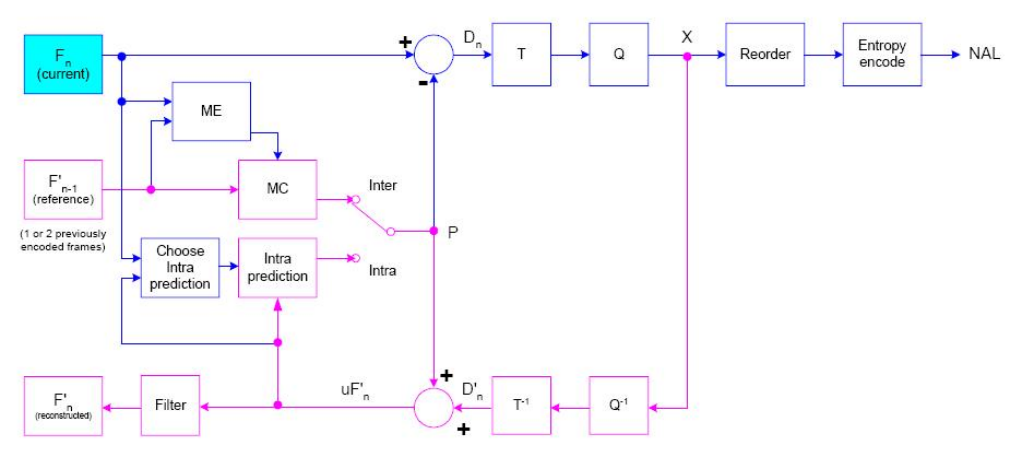
P-Frames: Motion Compensation

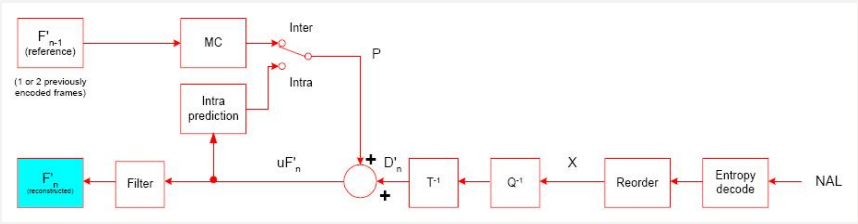


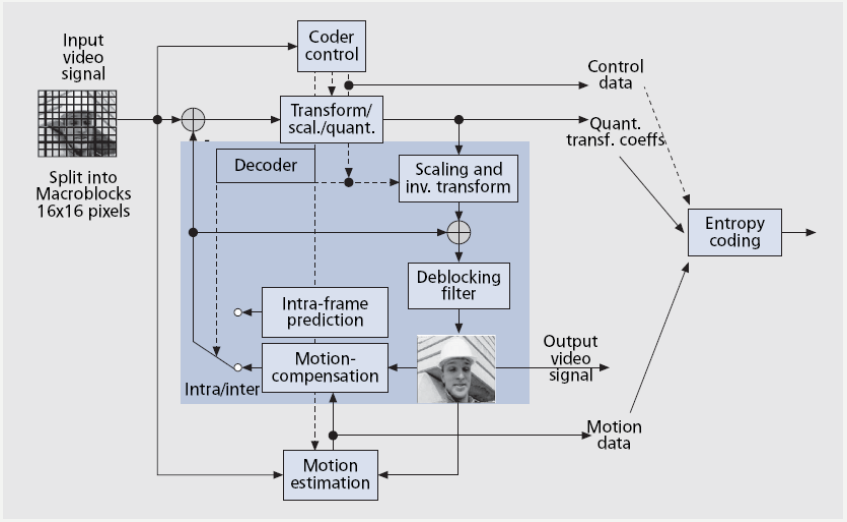
- Image is predicted by copying over parts of previous images
- Encoding of difference to predicted image



H264 encoder and decoder structure

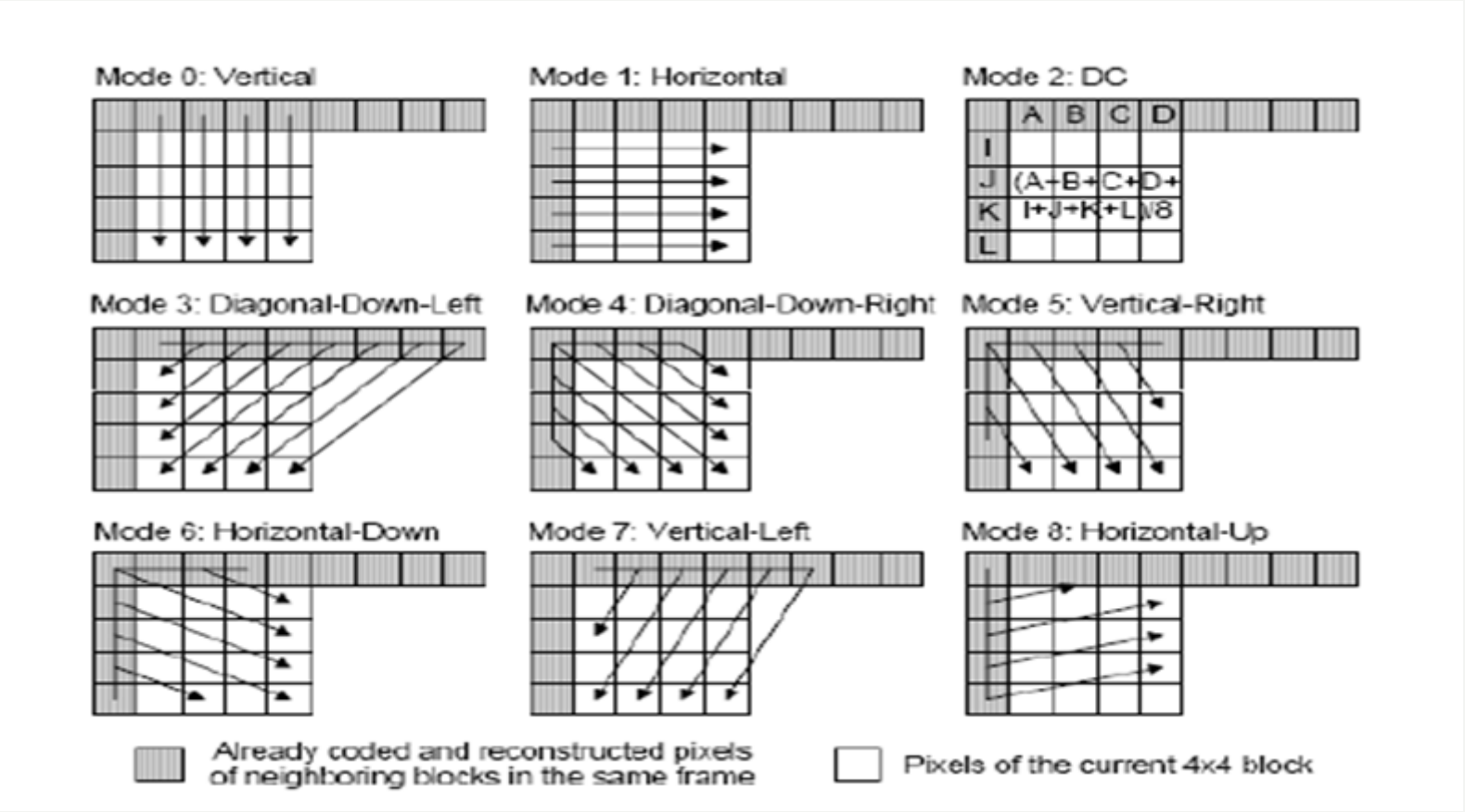






Source: http://mrutyunjayahiremath.blogspot.de/2010/09/h264-video-codec.html

H264 intra-frame prediction



Source: http://stackoverflow.com/questions/25336344/performing-intra-frame-prediction-in-matlab

Reduce artefacts due to 8x8 or 16x16 block structure