# Image compression

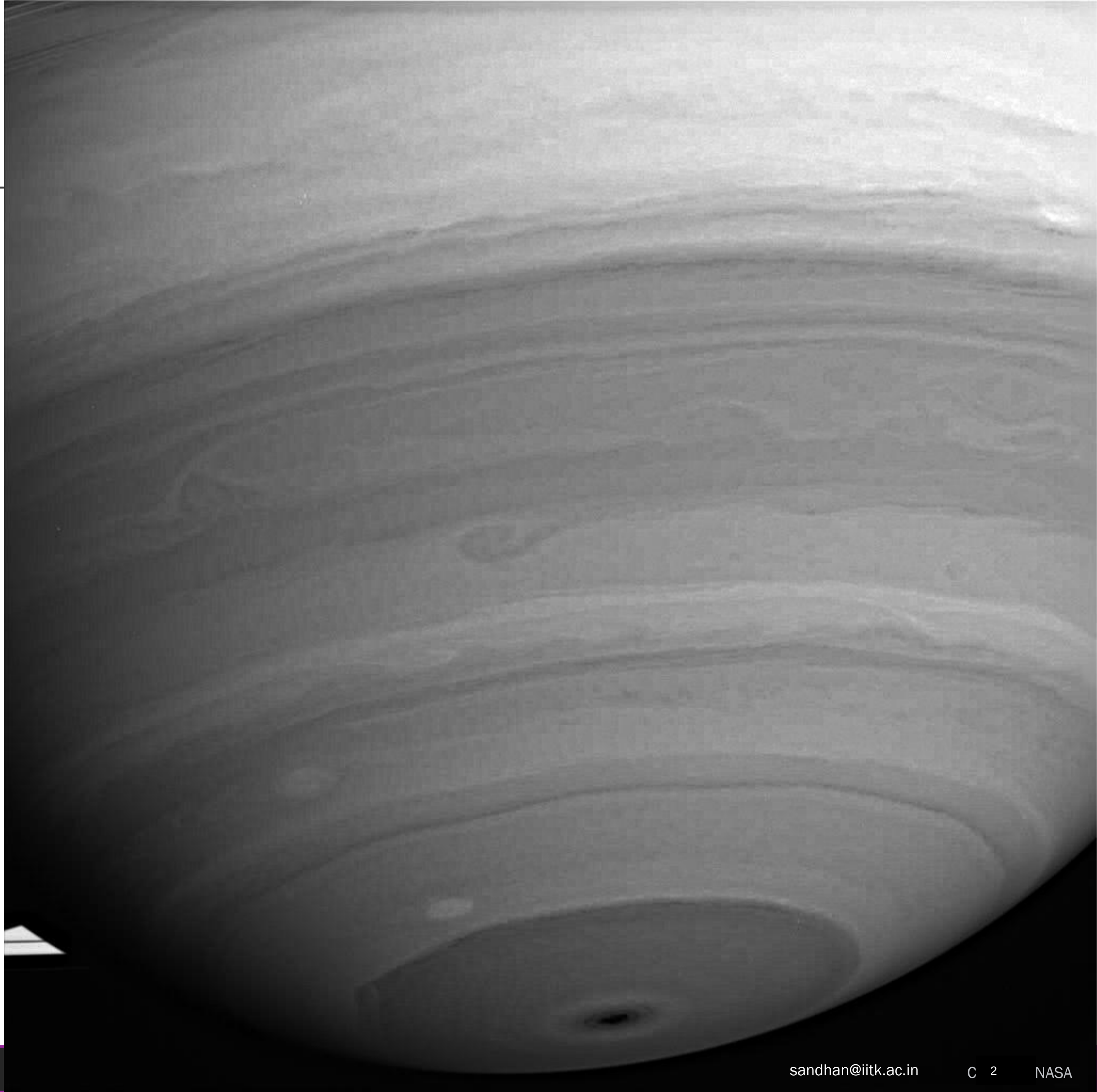
Dr. Tushar Sandhan

# Introduction

- Storage
  - memory devices are cheaper
  - compression & decompression add extra computational burden
  - do we really need compression?

# Introduction

- Storage
  - memory devices are cheaper
  - compression & decompression add extra computational burden
  - do we really need compression?

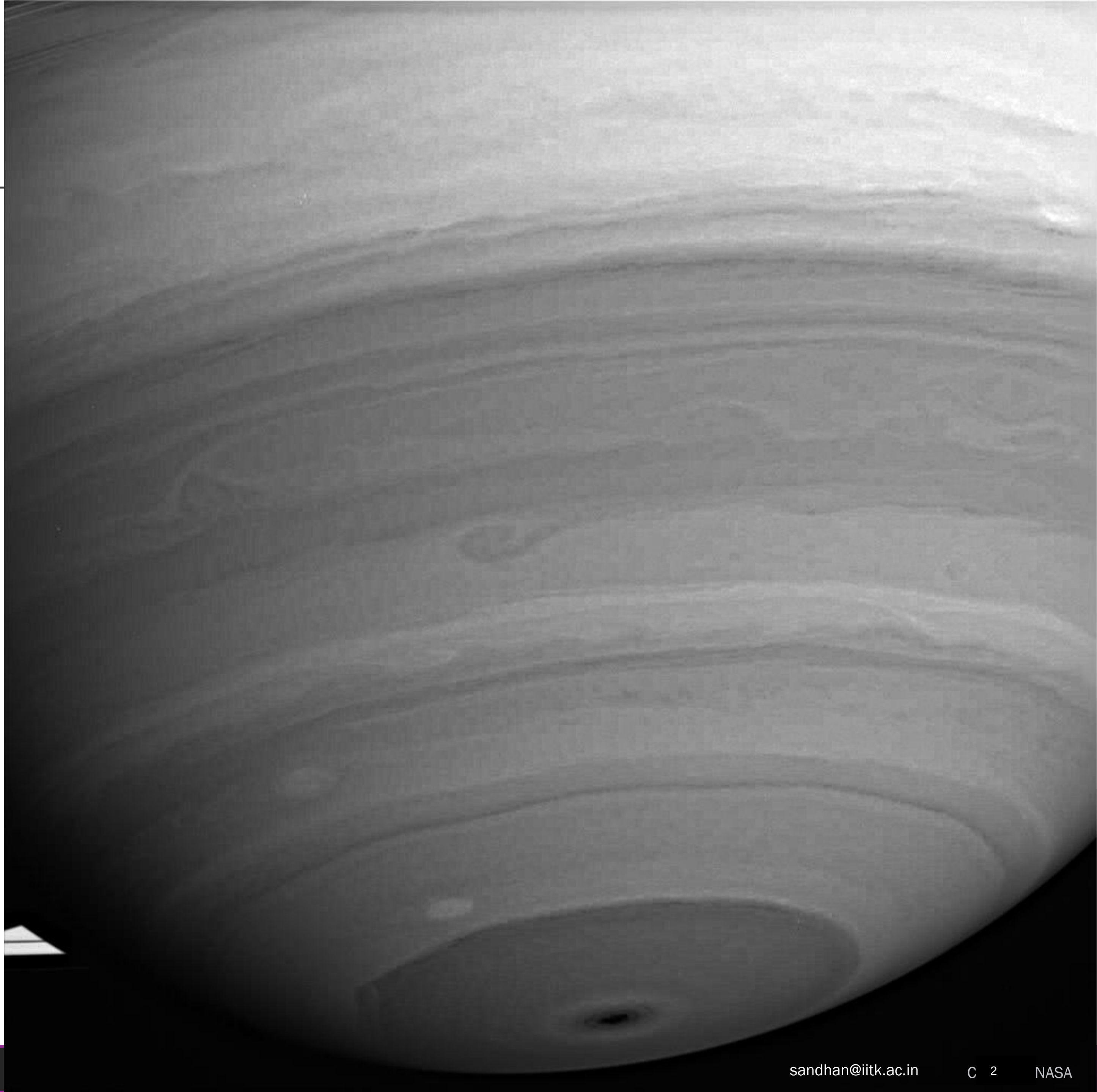


NASA

# Introduction

- Storage
  - memory devices are cheaper
  - compression & decompression add extra computational burden
  - do we really need compression?

- Saturn
  - infrared view of southern hemisphere
  - by Casssini spacecraft
  - taken from 1.3 million Km
  - obtained ground resolution 77 Km



NASA

# Compression need

- Storage & transmission
  - raw data occupies huge space
  - 1920 x 1080 image at 24 bits per pixels has a size of about 6.2 MB uncompressed
  - JPEG makes it 200KB
  - your 1hr long video lec recordings ~100MB

# Compression need

- Storage & transmission
  - raw data occupies huge space
  - 1920 x 1080 image at 24 bits per pixels has a size of about 6.2 MB uncompressed
  - JPEG makes it 200KB
  - your 1hr long video lec recordings ~100MB

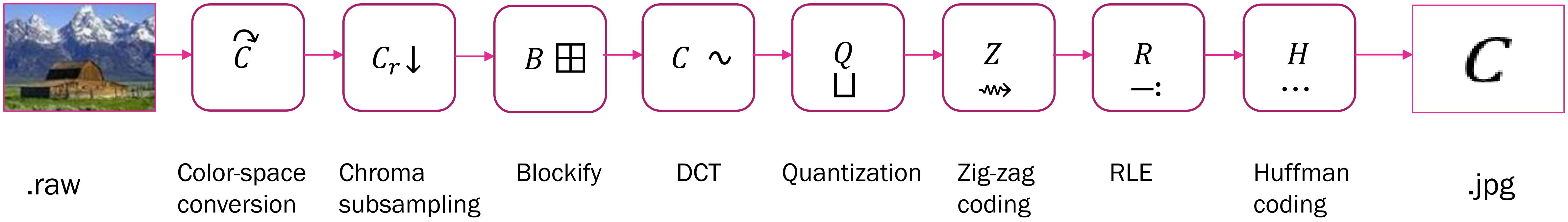
- We will see
  - JPEG
  - JPEG2000

# JPEG

- Joint Photographic Experts Group
  - JPEG: a lossy compression algorithm
  - it’s not a file format
  - standardized in 1992
  - a lossy compression

# JPEG

- Joint Photographic Experts Group
  - JPEG: a lossy compression algorithm
  - it’s not a file format
  - standardized in 1992
  - a lossy compression



.raw

Color-space conversion

Chroma subsampling

Blockify

DCT

Quantization

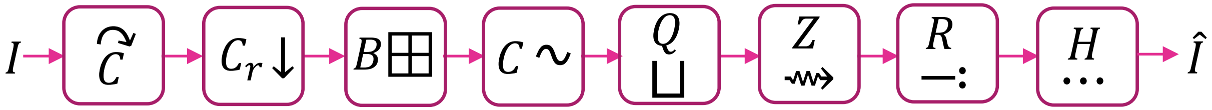
Zig-zag coding

RLE

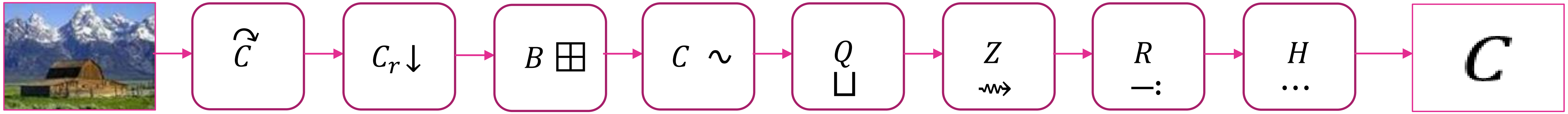
Huffman coding

.jpg

# JPEG



- Joint Photographic Experts Group
  - JPEG: a lossy compression algorithm
  - it’s not a file format
  - standardized in 1992
  - a lossy compression



.raw

Color-space conversion

Chroma subsampling

Blockify

DCT

Quantization

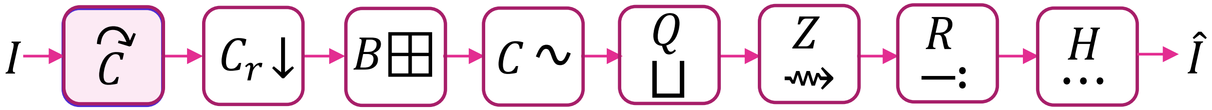
Zig-zag coding

RLE

Huffman coding

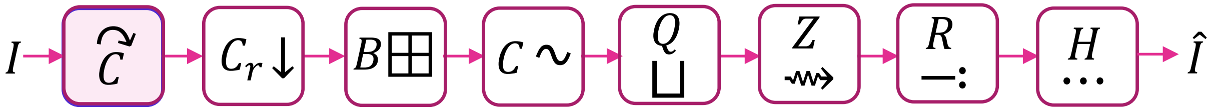
.jpg

# Color conversion



- RGB → YCbCr
  - Y – Luma
  - Cb – blue chroma
  - Cr – red chroma
  - used in TV, videos, JPEG, MPEG
  - allocate high bandwidth for Y & low for chromas
  - other variations:
    - YUV=PAL, YIQ=NTSE

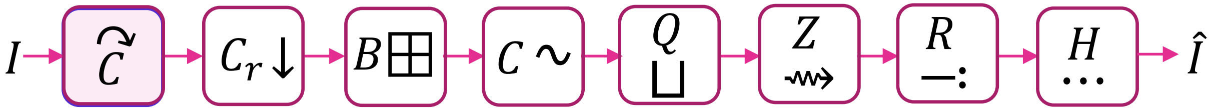
# Color conversion



- RGB → YCbCr
  - Y – Luma
  - Cb – blue chroma
  - Cr – red chroma
  - used in TV, videos, JPEG, MPEG
  - allocate high bandwidth for Y & low for chromas
  - other variations:
    - YUV=PAL, YIQ=NTSE

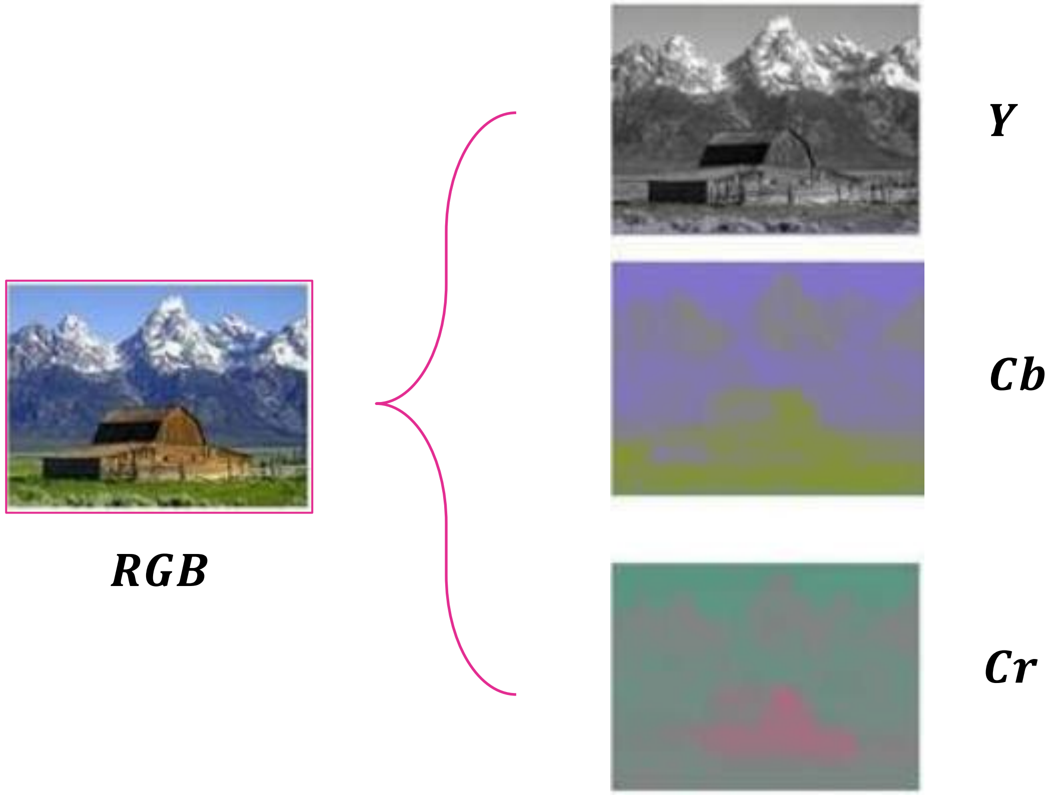
$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256}\begin{bmatrix} 65.73 & 129.05 & 25.06 \\ -37.94 & -74.49 & 112.43 \\ 112.43 & -94.15 & -18.28 \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

# Color conversion

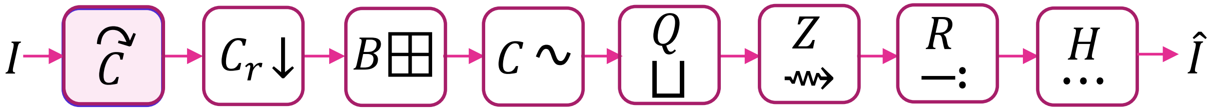


- RGB → YCbCr
  - Y – Luma
  - Cb – blue chroma
  - Cr – red chroma
  - used in TV, videos, JPEG, MPEG
  - allocate high bandwidth for Y & low for chromas
  - other variations:
    - YUV=PAL, YIQ=NTSE

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256}\begin{bmatrix} 65.73 & 129.05 & 25.06 \\ -37.94 & -74.49 & 112.43 \\ 112.43 & -94.15 & -18.28 \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



# Color conversion

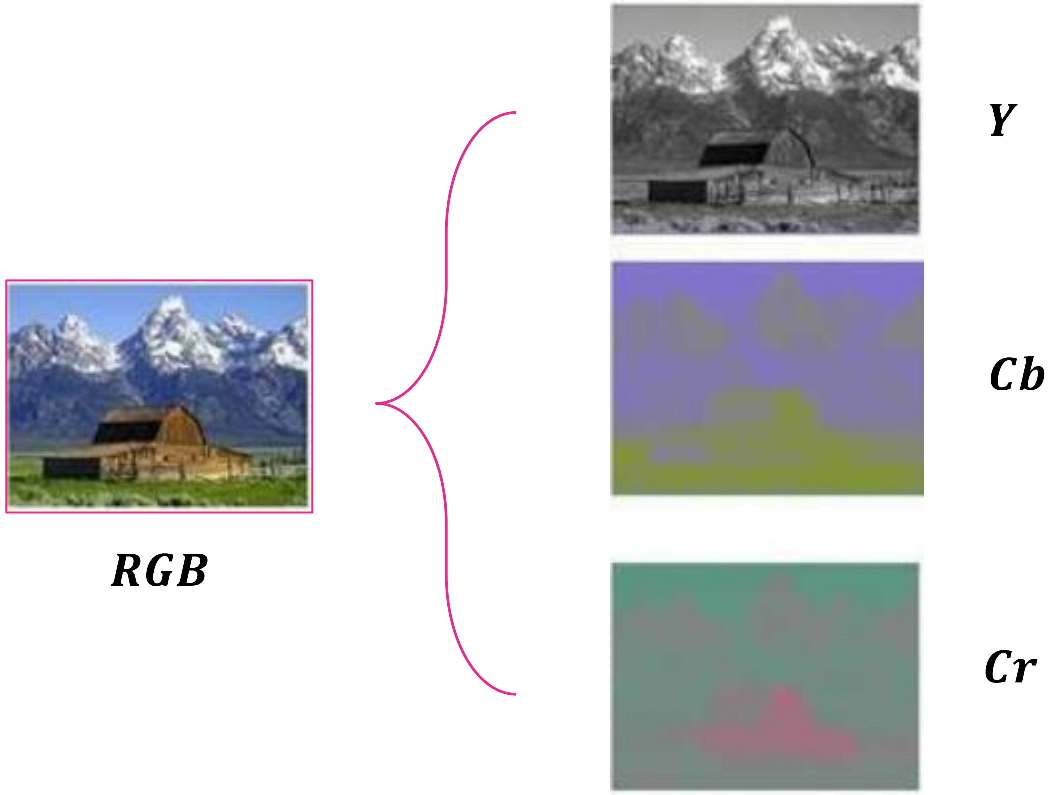


- RGB → YCbCr
  - Y – Luma
  - Cb – blue chroma
  - Cr – red chroma
  - used in TV, videos, JPEG, MPEG
  - allocate high bandwidth for Y & low for chromas
  - other variations:
    - YUV=PAL, YIQ=NTSE

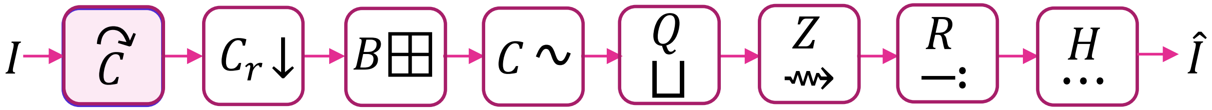
$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256}\begin{bmatrix} 65.73 & 129.05 & 25.06 \\ -37.94 & -74.49 & 112.43 \\ 112.43 & -94.15 & -18.28 \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Y ∈ [16/255, 235/255]

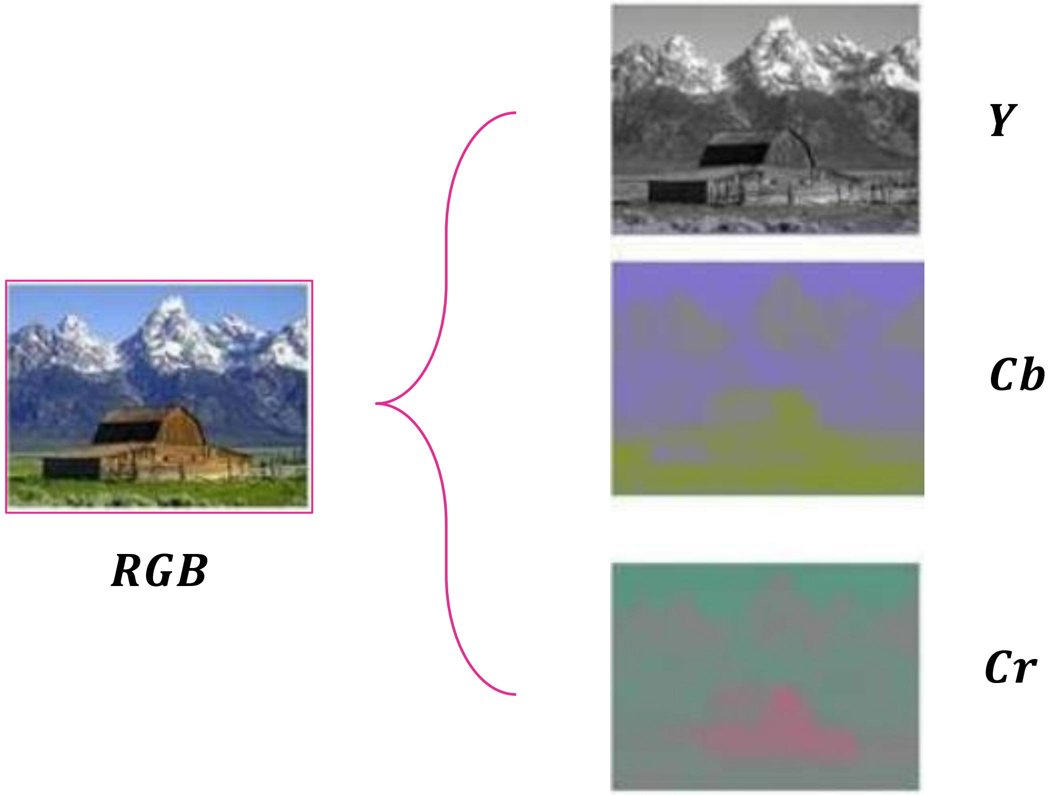
$C_b, C_r$ ∈ [16/255, 240/255]



# Color conversion



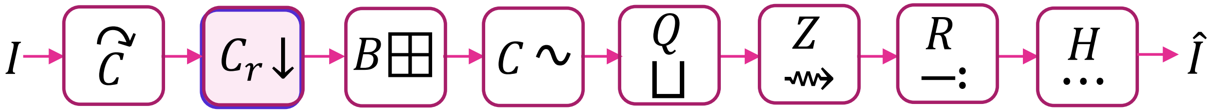
- RGB → YCbCr
  - Y – Luma
  - Cb – blue chroma
  - Cr – red chroma
  - used in TV, videos, JPEG, MPEG
  - allocate high bandwidth for Y & low for chromas
  - other variations:
    - YUV=PAL, YIQ=NTSE



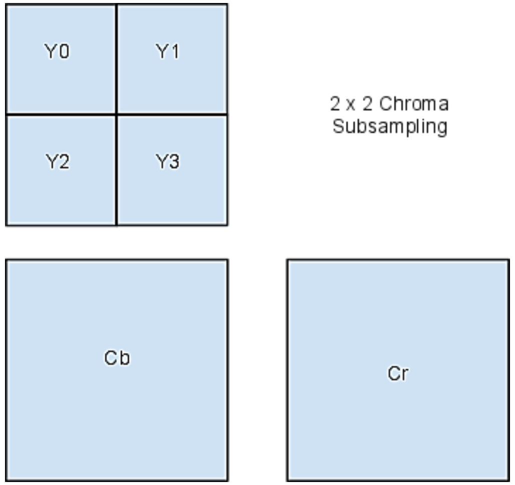
$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256}\begin{bmatrix} 65.73 & 129.05 & 25.06 \\ -37.94 & -74.49 & 112.43 \\ 112.43 & -94.15 & -18.28 \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\left.\begin{array}{l} Y \in [16/255, 235/255] \\ C_b, C_r \in [16/255, 240/255] \end{array}\right\} \text{Additional non-image info can be added}$$

# Chroma subsampling

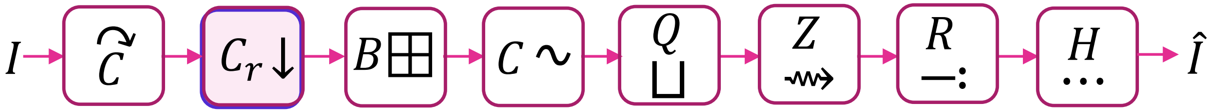


- Decimating only chroma
  - ○ subsampling is done only in chroma
  - ○ 4:2:0 JPG, video H.264 codec
  - ○ Y is stored at full resolution

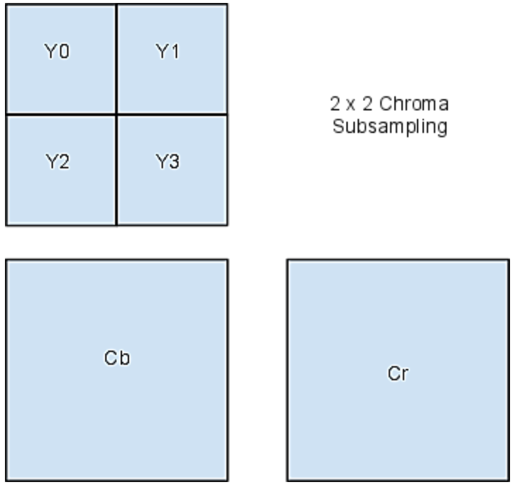


| | | | |
|---|---|---|---|
| $Y'_{00}$<br>$Cb_{00}$<br>$Cr_{00}$ | $Y'_{10}$ | $Y'_{20}$<br>$Cb_{20}$<br>$Cr_{20}$ | $Y'_{30}$ |
| $Y'_{01}$ | $Y'_{11}$ | $Y'_{21}$ | $Y'_{31}$ |

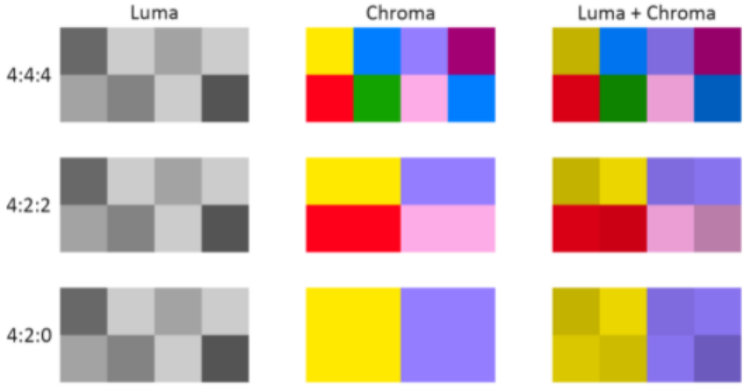
# Chroma subsampling



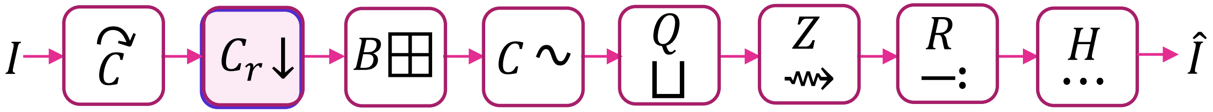
- Decimating only chroma
  - subsampling is done only in chroma
  - 4:2:0 JPG, video H.264 codec
  - Y is stored at full resolution



| | | | |
|---|---|---|---|
| $Y'_{00}$<br>$Cb_{00}$<br>$Cr_{00}$ | $Y'_{10}$ | $Y'_{20}$<br>$Cb_{20}$<br>$Cr_{20}$ | $Y'_{30}$ |
| $Y'_{01}$ | $Y'_{11}$ | $Y'_{21}$ | $Y'_{31}$ |

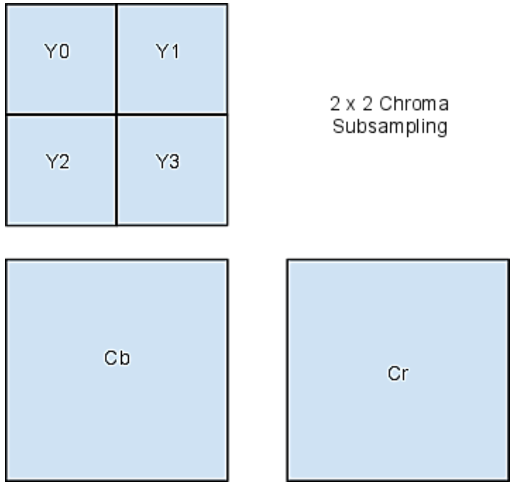


# Chroma subsampling

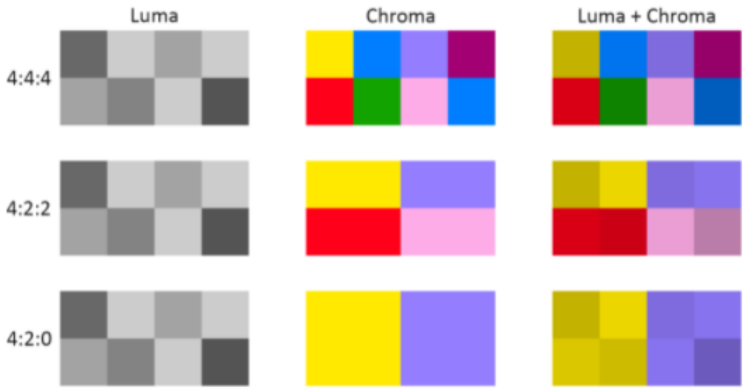


- Decimating only chroma
  - subsampling is done only in chroma
  - 4:2:0 JPG, video H.264 codec
  - Y is stored at full resolution

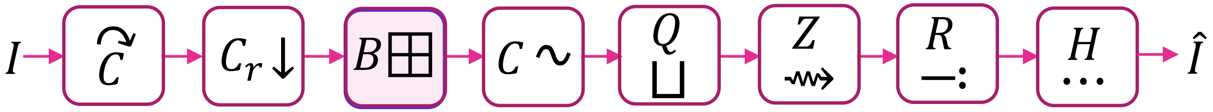
| | **Subsampling** |
|---|---|
| **PC** | 4:4:4 |
| **Movies** | 4:2:0 |
| **Video Games** | 4:4:4 |
| **Sports** | 4:2:0 |
| **TV Shows** | 4:2:0 |



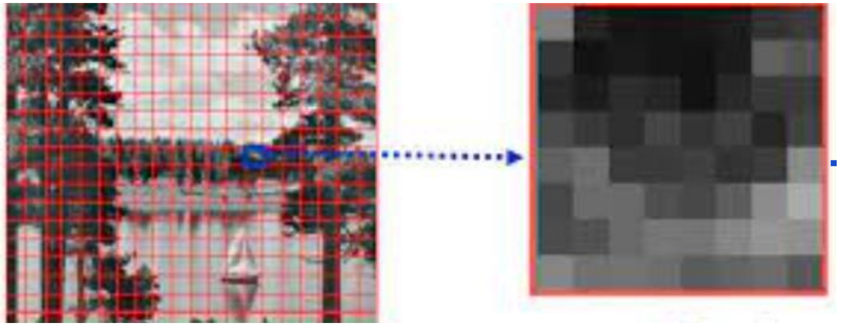
| | | | |
|---|---|---|---|
| $Y'_{00}$<br>$Cb_{00}$<br>$Cr_{00}$ | $Y'_{10}$ | $Y'_{20}$<br>$Cb_{20}$<br>$Cr_{20}$ | $Y'_{30}$ |
| $Y'_{01}$ | $Y'_{11}$ | $Y'_{21}$ | $Y'_{31}$ |



# Blocking

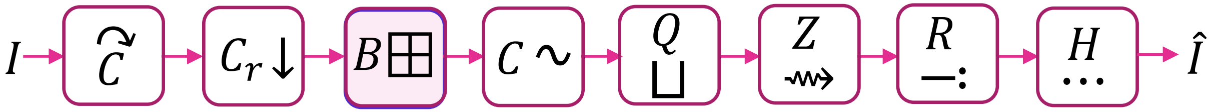


- Analyze the image into smaller blocks
  - small blocks called subimage
  - subimage sizes 8x8, 16x16 etc.
  - Jpeg uses 8x8

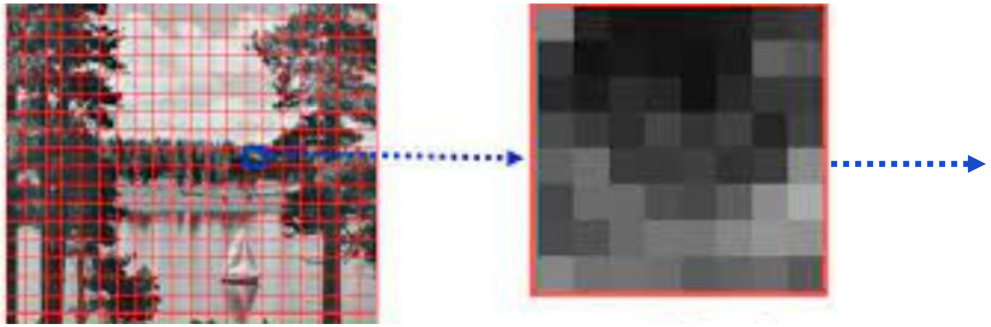


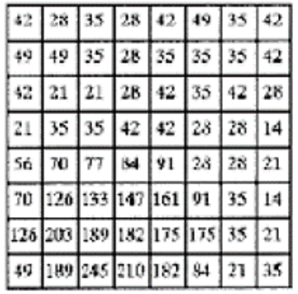
|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 42 | 28 | 35 | 28 | 42 | 49 | 35 | 42 |
| 49 | 49 | 35 | 28 | 35 | 35 | 35 | 42 |
| 42 | 21 | 21 | 28 | 42 | 35 | 42 | 28 |
| 21 | 35 | 35 | 42 | 42 | 28 | 28 | 14 |
| 56 | 70 | 77 | 84 | 91 | 28 | 28 | 21 |
| 70 | 126 | 133 | 147 | 161 | 91 | 35 | 14 |
| 126 | 203 | 189 | 182 | 175 | 175 | 35 | 21 |
| 49 | 189 | 245 | 210 | 182 | 84 | 21 | 35 |

# Blocking



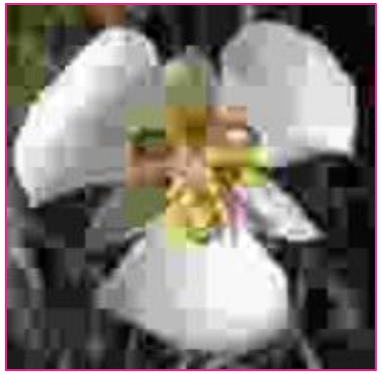
- Analyze the image into smaller blocks
  - small blocks called subimage
  - subimage sizes 8x8, 16x16 etc.
  - Jpeg uses 8x8



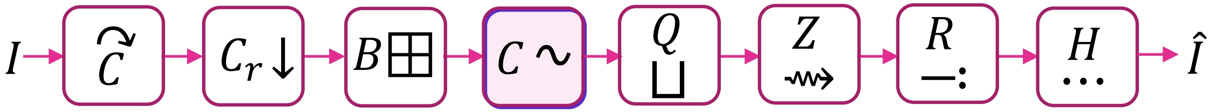


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 42 | 28 | 35 | 28 | 42 | 49 | 35 | 42 |
| 49 | 49 | 35 | 28 | 35 | 35 | 35 | 42 |
| 42 | 21 | 21 | 28 | 42 | 35 | 42 | 28 |
| 21 | 35 | 35 | 42 | 42 | 28 | 28 | 14 |
| 56 | 70 | 77 | 84 | 91 | 28 | 28 | 21 |
| 70 | 126 | 133 | 147 | 161 | 91 | 35 | 14 |
| 126 | 203 | 189 | 182 | 175 | 175 | 35 | 21 |
| 49 | 189 | 245 | 210 | 182 | 84 | 21 | 35 |

Blocking artifacts

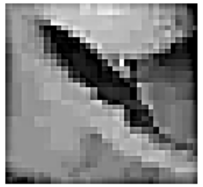


# DCT

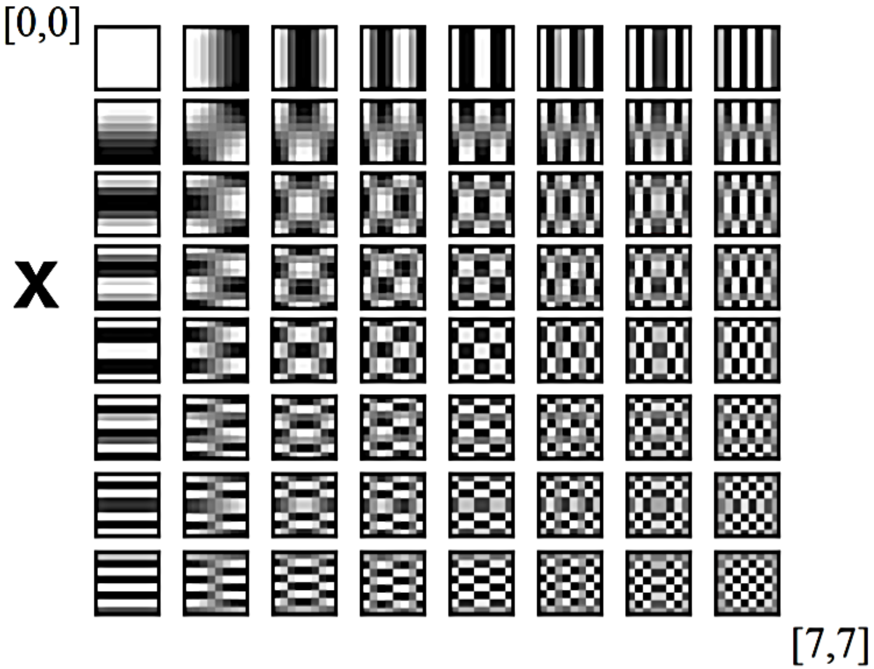


- Discrete cosine transform

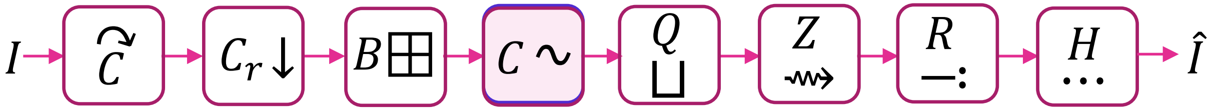
$$\text{basis}[i,j] = \cos\left[\pi\frac{i}{N}\left(x+\frac{1}{2}\right)\right] \times \cos\left[\pi\frac{j}{N}\left(y+\frac{1}{2}\right)\right]$$

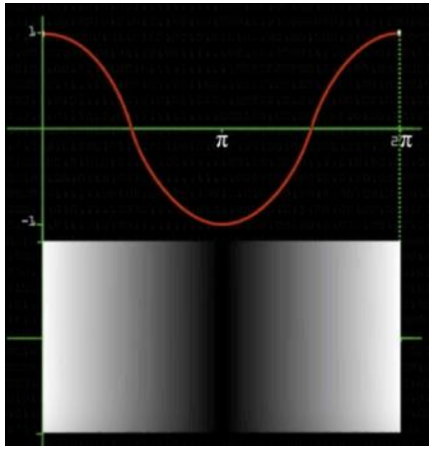


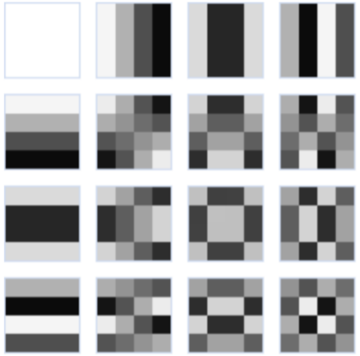
$$= \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \times$$



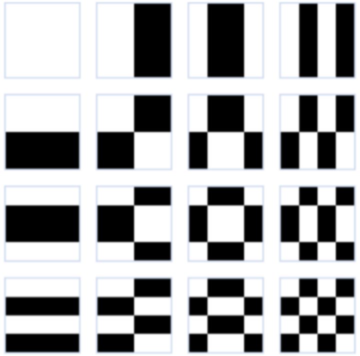
# DCT



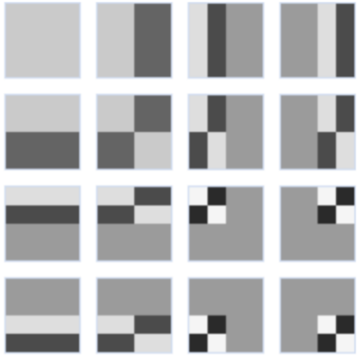




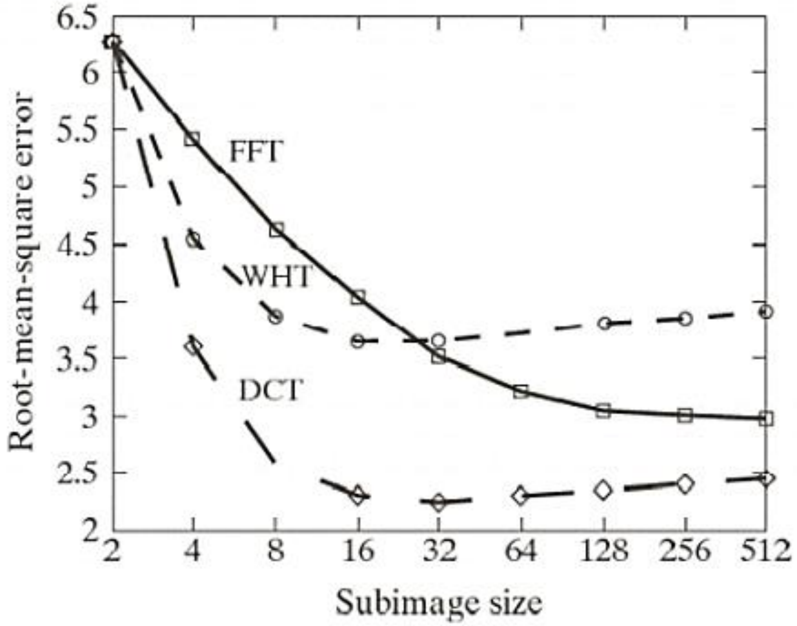
DCT



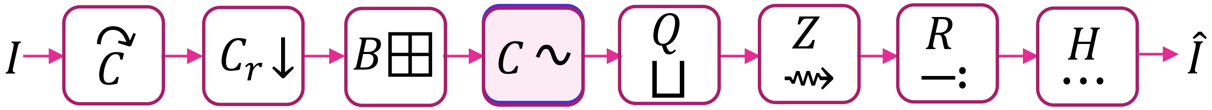
Walsh Hadamard

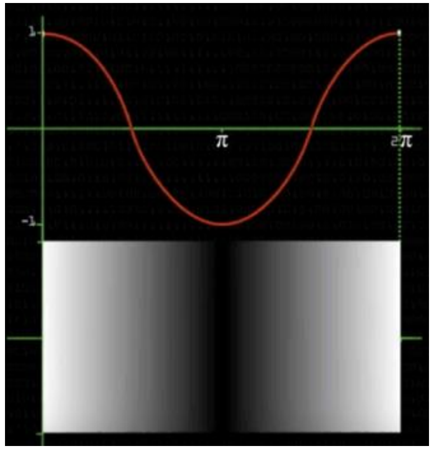


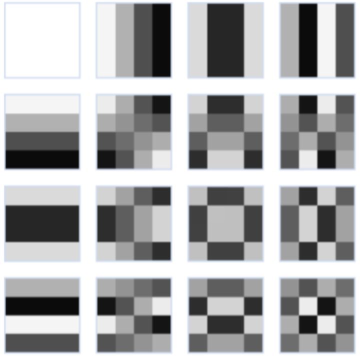
Haar wavelet



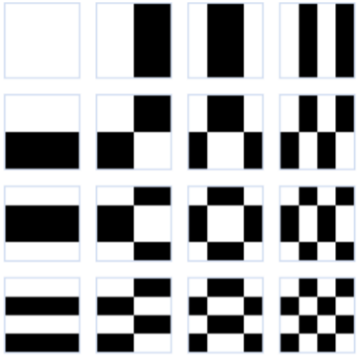
# DCT



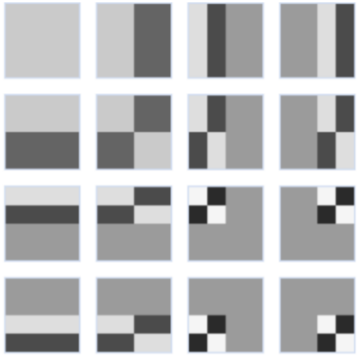




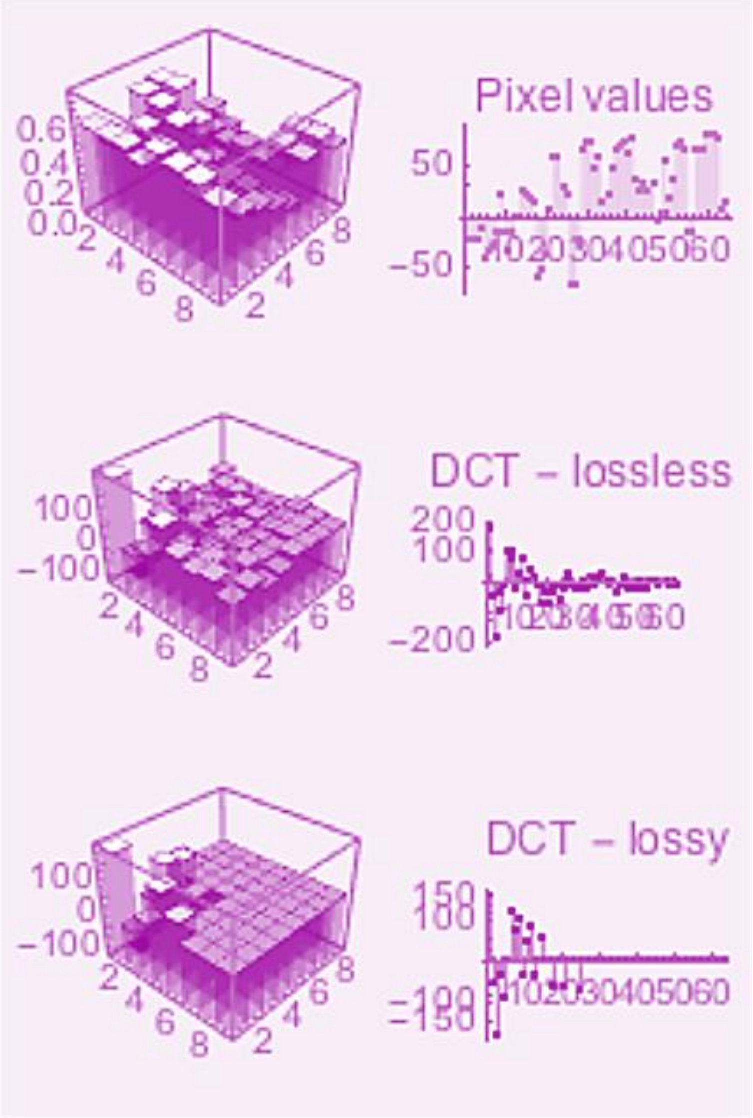
DCT



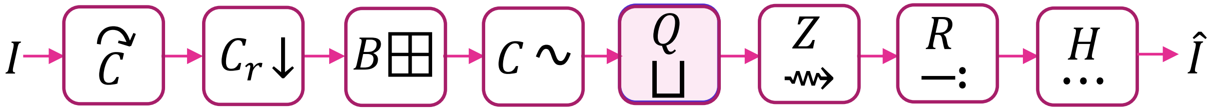
Walsh Hadamard



Haar wavelet



# Quantization



- Psycovisually-tunned quantization tables
  - experiments on human subjects to find quantization values

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

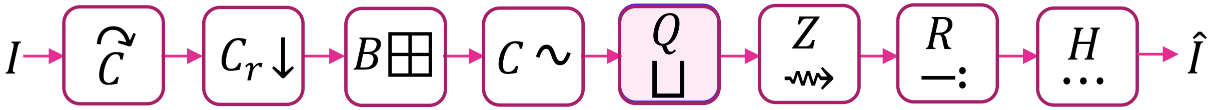
$$\hat{c}[k_1, k_2] = \text{round}(c[k_1, k_2]/Q[k_1, k_2])$$

Format: JPEG

Quality:

File Size: 1.1 MB

# Quantization



- Psycovisually-tunned quantization tables
  - experiments on human subjects to find quantization values

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

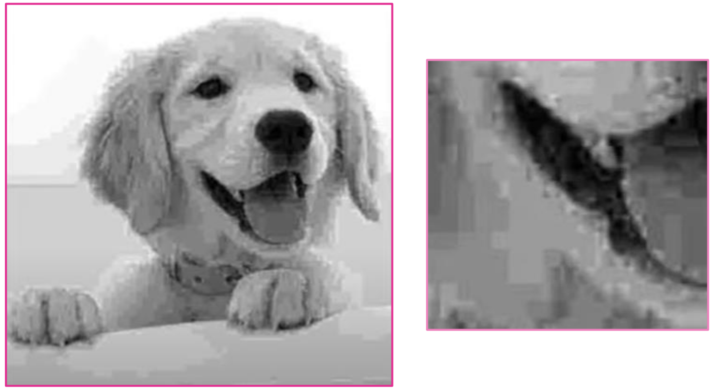
$$\hat{c}[k_1, k_2] = \text{round}(c[k_1, k_2]/Q[k_1, k_2])$$

Format: JPEG

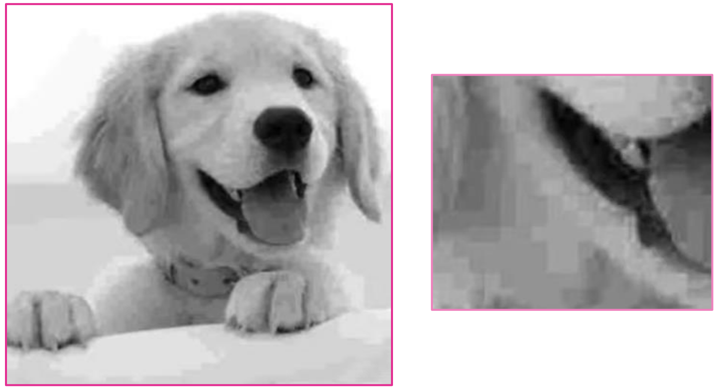
Quality: Least … Best

File Size: 1.1 MB

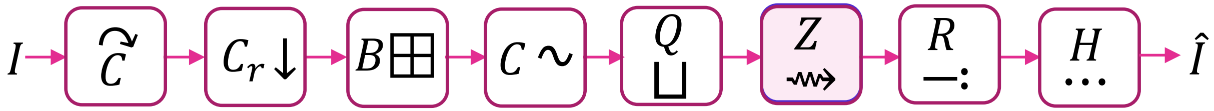
Uniform



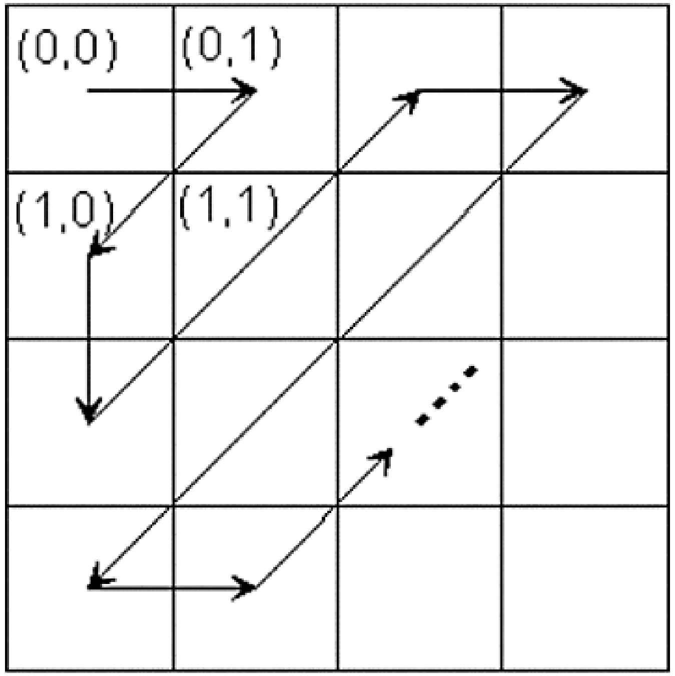
Q



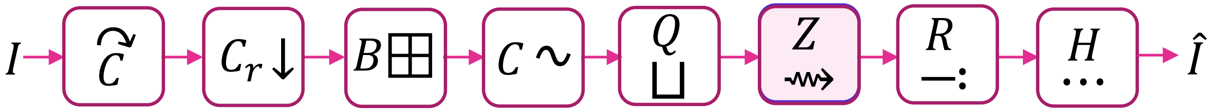
# Coding



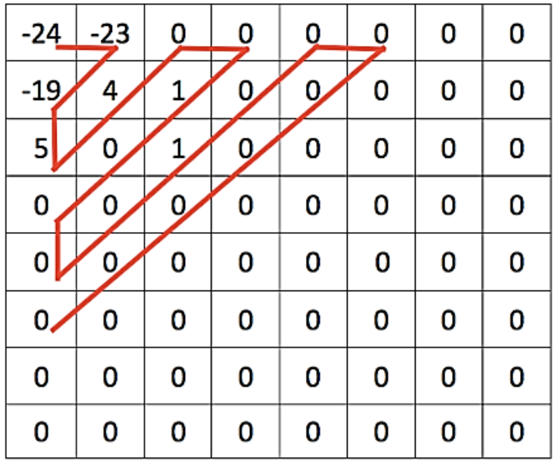
- Zig-zag coding
  - what can we achieve?



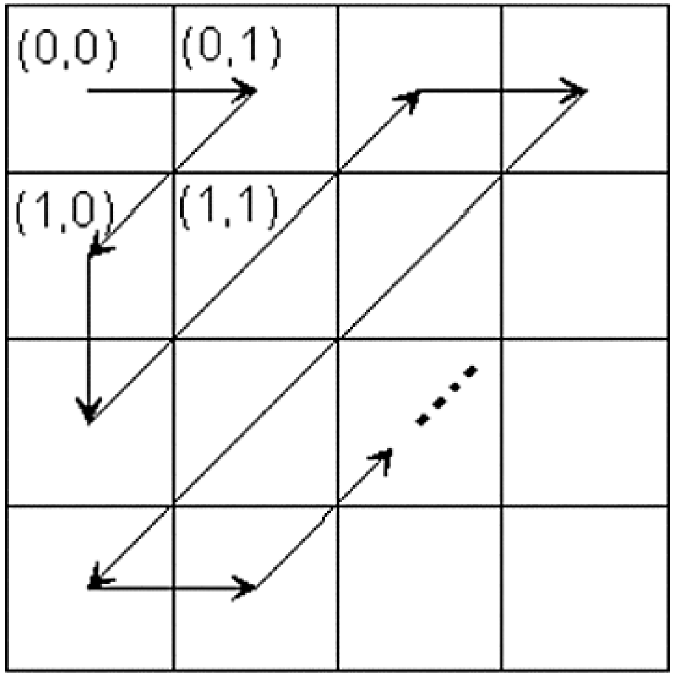
# Coding



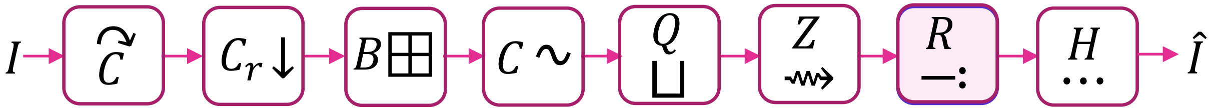
- Zig-zag coding
  - what can we achieve?



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -24 | -23 | 0 | 0 | 0 | 0 | 0 | 0 |
| -19 | 4 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 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 | 0 | 0 | 0 | 0 | 0 | 0 |



# RLE: Runlength encoding



- Quantized: $\hat{C} =$

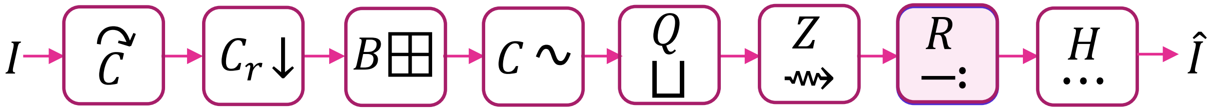
$$\begin{bmatrix} 100 & -60 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Coding:

100, -60, 0, 0, 0, 0, 6, 0, 0, 0, 13, 0, 0, 0, 0, 0, 0, 0, 0, 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

Courtesy: EPFL

# RLE: Runlength encoding



- Quantized: $\hat{C} =$

$$\begin{bmatrix} 100 & -60 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Coding:

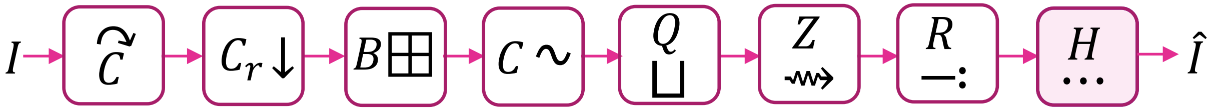
100, -60, 0, 0, 0, 0, 6, 0, 0, 0, 13, 0, 0, 0, 0, 0, 0, 0, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

- $[(r, s), c]$
  - $c$ – current value
  - $r$ – #of 0 before $c$
  - $s$ – #bits needed to encode $c$
  - $0 \leq s \leq 11; \;\; 0 \leq r \leq 15$
  - $(r, s) \leftarrow$ 8 bits uchar

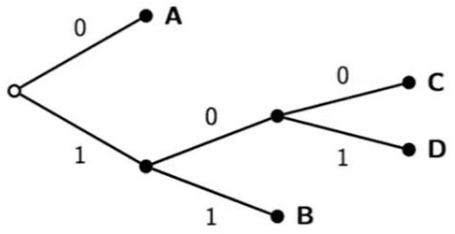
$[(0,7),\ 100],\ [(0,6),\ -60],\ [(4,3),\ 6],\ [(3,4),\ 13],\ [(8,1),\ -1], [(0,0)]$

Courtesy: EPFL

# Hoffman coding

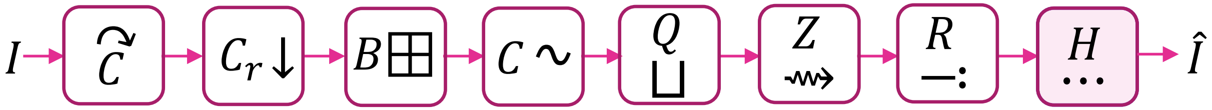


- Lossless coding method
  - RLE output can be coded by any lossless coding methods (e.g. methods from communications, networks etc.)
  - JPEG uses Hoffman coding (1952)
  - it's a variable length code



RLE: AABAABADC → 001100110101100

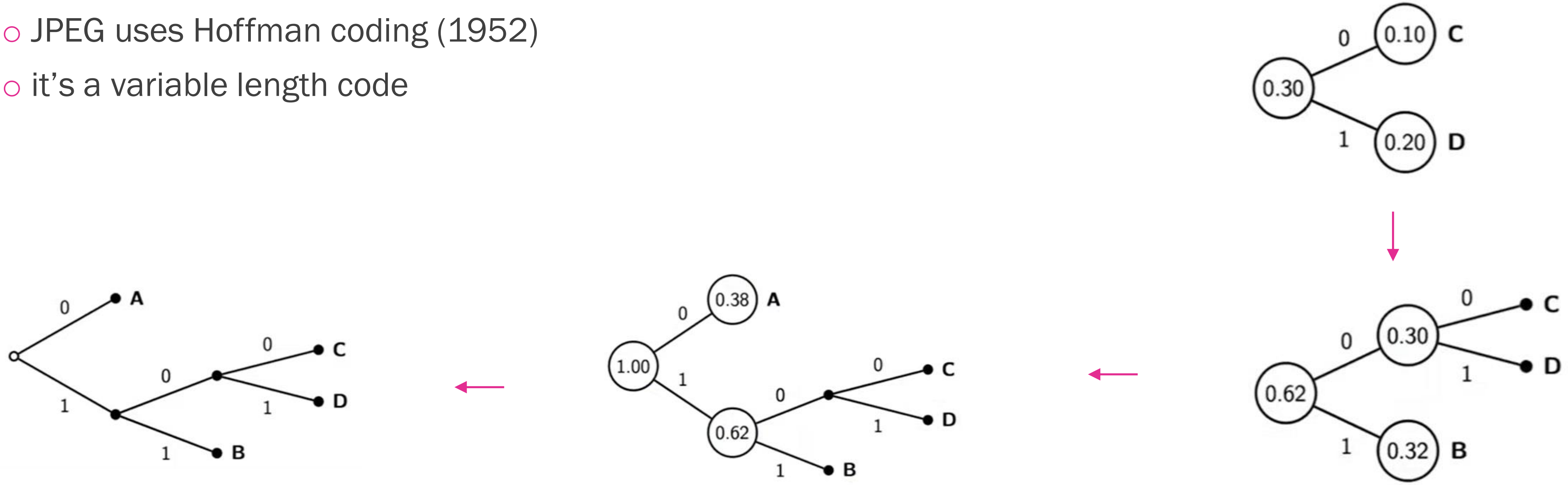
# Hoffman coding



- Lossless coding method
  - RLE output can be coded by any lossless coding methods (e.g. methods from communications, networks etc.)
  - JPEG uses Hoffman coding (1952)
  - it's a variable length code

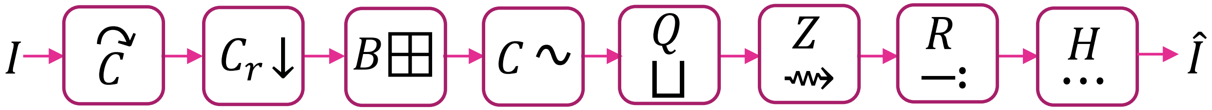
$p(A) = 0.38$ $p(B) = 0.32$

$p(C) = 0.1$ $p(D) = 0.2$



RLE: AABAABADC → 001100110101100

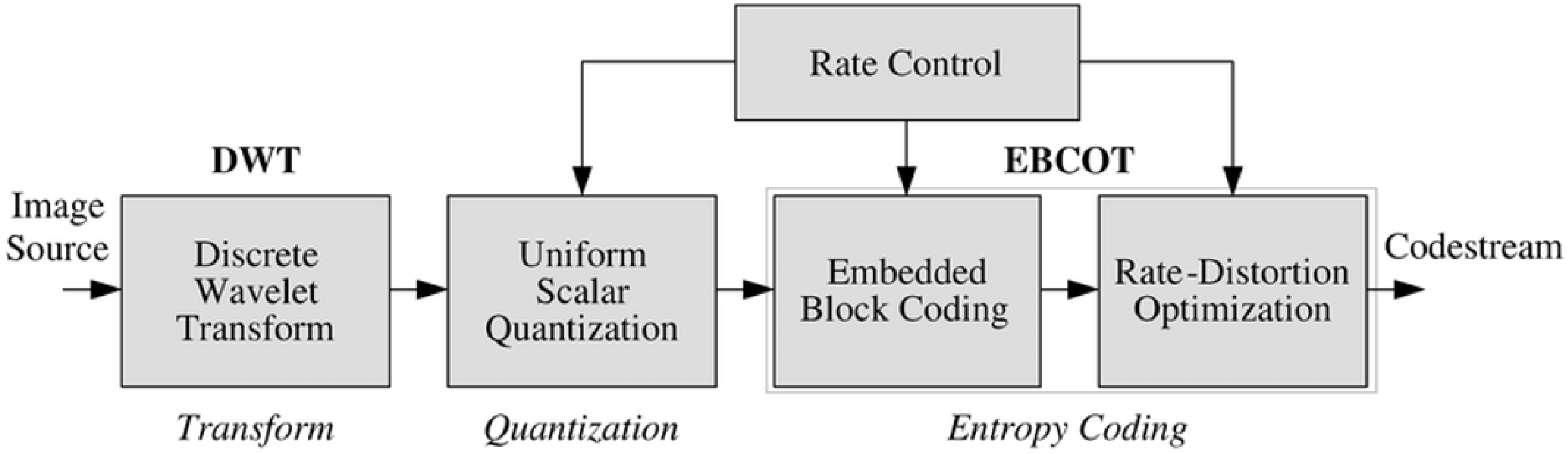
# JPEG

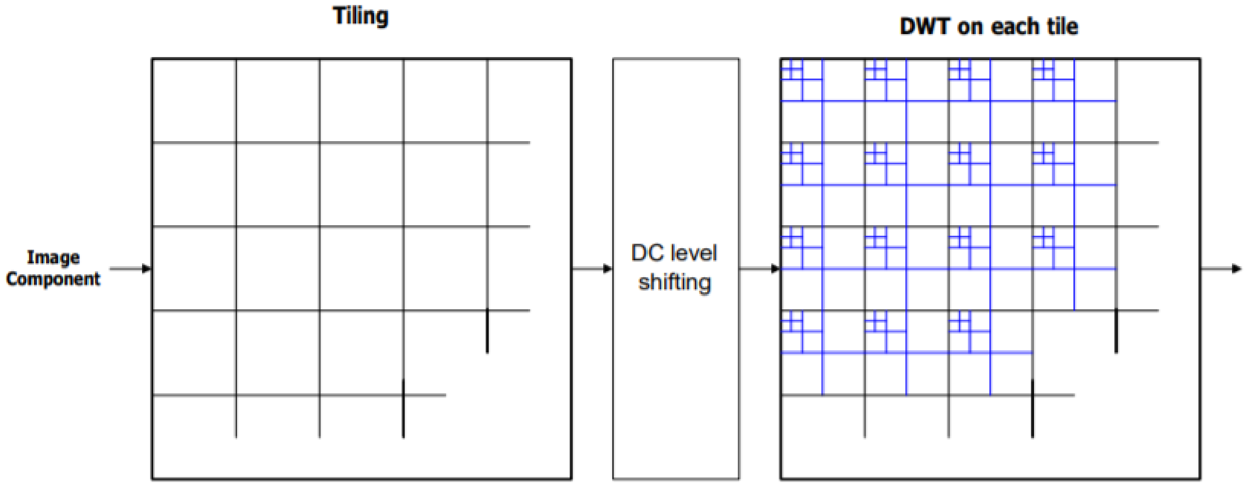


- Pseudocode

```
: $RGB \rightarrow YCbCr$
: $CbCr$ dessimation (4:2:0) to $Cb'Cr'$
: For each channel in $[Y, Cb', Cr']$:
    : blockify into 8x8 subblocks
    : For each subblock
        : get DCT
        : pyscovisually quantize
        : zig-zag coding
        : RLE
        : Huffman
```

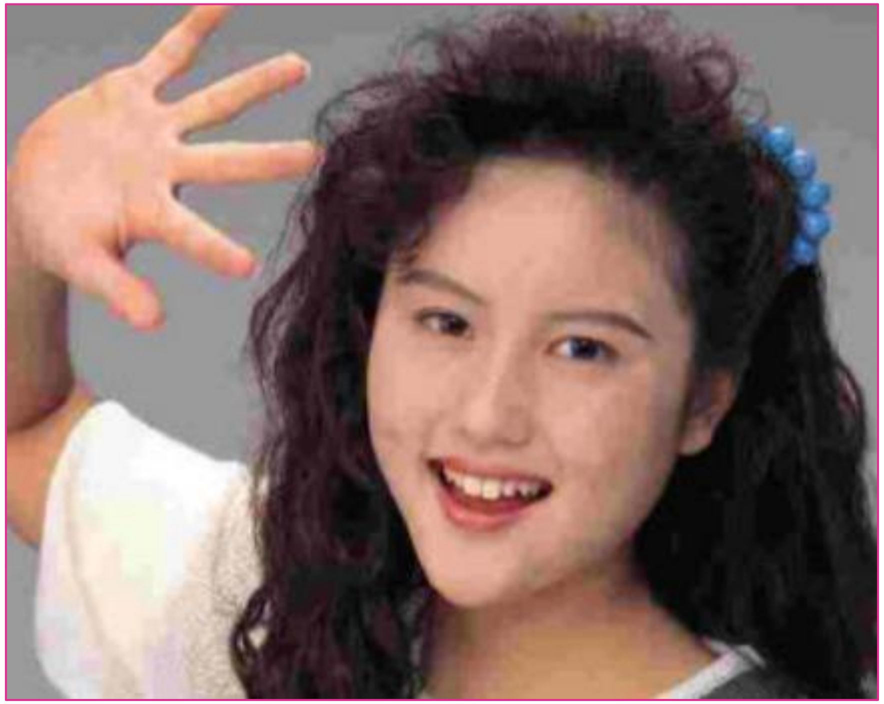
# JPEG 2000


Rate Control
DWT
EBCOT
Image Source
Discrete Wavelet Transform
Uniform Scalar Quantization
Embedded Block Coding
Rate-Distortion Optimization
Codestream
Transform
Quantization
Entropy Coding


Tiling
DWT on each tile
Image Component
DC level shifting

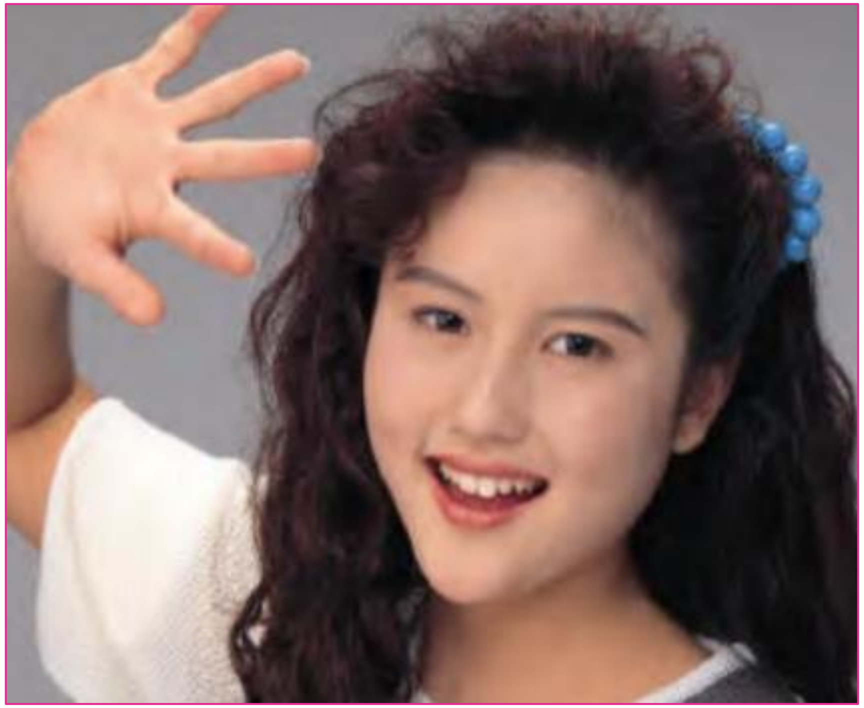
# Comparison

JPEG



0.25bpp

JPEG 2000



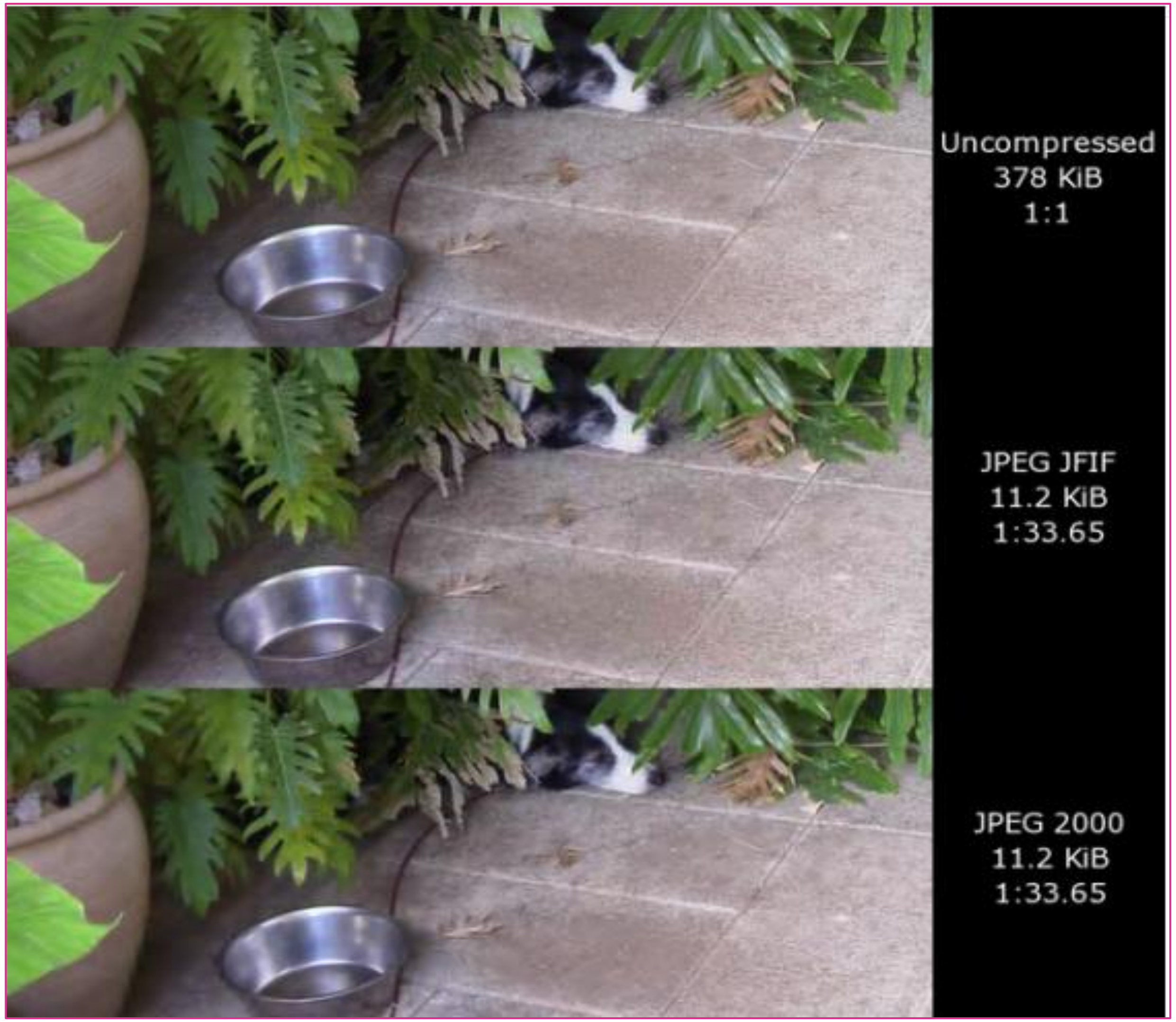
0.25bpp

Courtesy: Christopoulos et al.

# Comparison

| JPEG | JPEG 2000 |
|---|---|
| DCT | DWT |
| Blocks | Tiles |
| Less compression ratio | High compression ratio |
| Less computations | Relatively higher computations |
| Quality low at low bit rate | Better quality at low bit rate |

# Example


Uncompressed
378 KiB
1:1
JPEG JFIF
11.2 KiB
1:33.65
JPEG 2000
11.2 KiB
1:33.65

# References

- Compression

# References

- Compression

❑ Dennis Gabor, ‘STFT: short time Fourier transform’, 1946

❑ N. Ahmed et al. ‘Discrete cosine transform’, IEEE Trans. on computers, 1974

❑ G. Zweig, ‘The first continuous wavelet transform’, 1975

❑ J. Li, ‘Image compression: the mathematics of jpeg 2000’, modern sig proc 2003

❑ HA. Rowley et al., ‘Neural network-based face detection’, IEEE CVPR, 1996

❑ H. Noda et al., ‘Colorization in YCbCr Color Space and Its Application to JPEG Images’, IEEE ICIP, 2007

# References

- Compression

❑ Dennis Gabor, ‘STFT: short time Fourier transform’, 1946

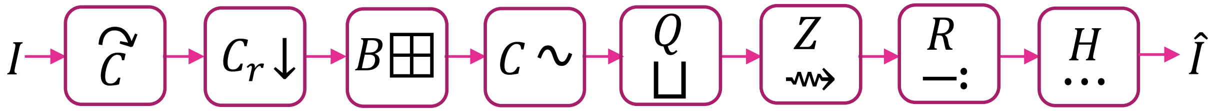
❑ N. Ahmed et al. ‘Discrete cosine transform’, IEEE Trans. on computers, 1974

❑ G. Zweig, ‘The first continuous wavelet transform’, 1975

❑ J. Li, ‘Image compression: the mathematics of jpeg 2000’, modern sig proc 2003

❑ HA. Rowley et al., ‘Neural network-based face detection’, IEEE CVPR, 1996

❑ H. Noda et al., ‘Colorization in YCbCr Color Space and Its Application to JPEG Images’, IEEE ICIP, 2007



$I$
$C$
$C_r\downarrow$
$B$ ⊞
$C\sim$
$Q$ ⊔
$Z$ ⇝
$R$ —:
$H$ …
$\hat{I}$