Introduction

Compressed Domain Processing (CDP)

UG-4, Program Elective

(L-T-P-C: 2-1-0-3)

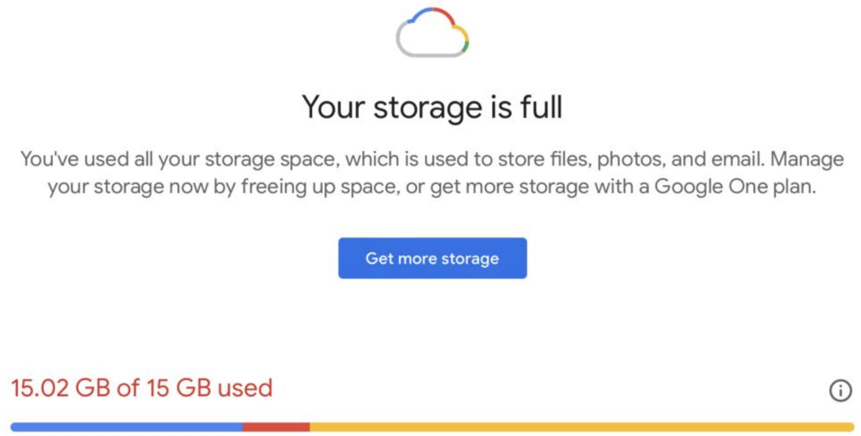
2026S

By

Dr. Bulla Rajesh

Assistant Professor, IIIT - Sri City

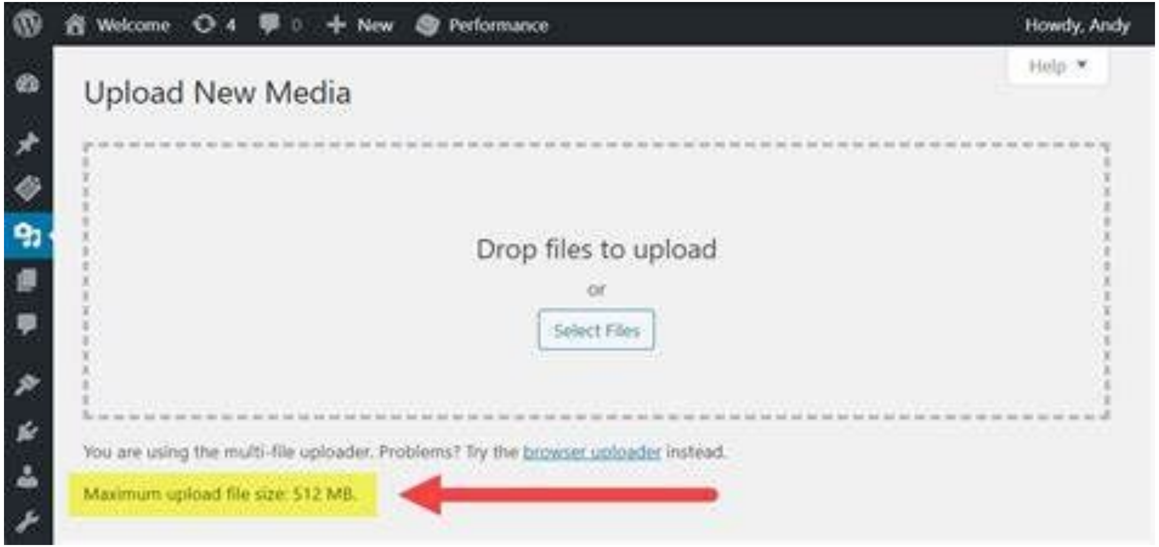
- Gmail Storage
- Google Photos
- Application Uploading
- Online Computations and transmission



Your storage is full

You've used all your storage space, which is used to store files, photos, and email. Manage your storage now by freeing up space, or get more storage with a Google One plan.

15.02 GB of 15 GB used



Upload New Media

Drop files to upload

or

You are using the multi-file uploader. Problems? Try the browser uploader instead.

Maximum upload file size: 512 MB.

Increase the Luggage Volume by 70%
Before
Organize your luggage and carry it easily
After

Proper Organization or removal insignificant leads to **Compression**

Major Data forms

Text

Image

video

Uncompressed File
File Size: 65KB
Compressed File
File Size: 13KB

Compressed Domain Processing ???   What??

Processing and analyzing on compressed representation of data

What is compression ??

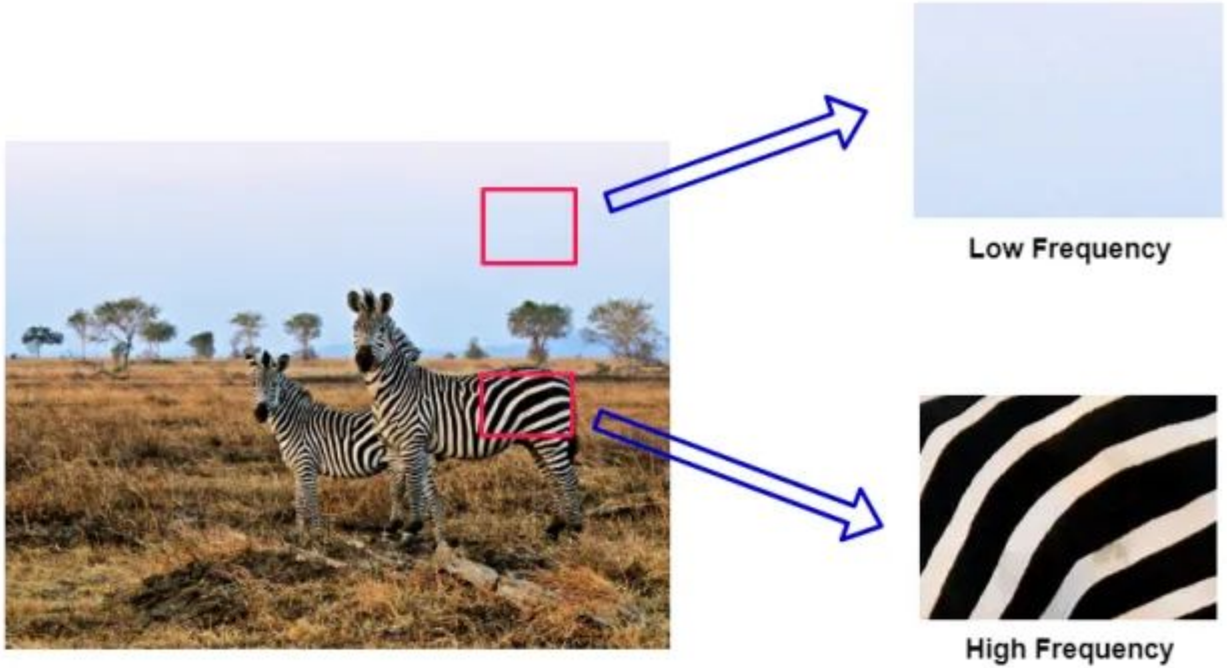
- ❖ In image processing or in general, compression refers to the process of keeping data in possible compact representation or reducing the size of an image file without significantly compromising its quality.

- ❖ The main goal of compression are to
  - ➢ Save storage space or reduce the time required for transmission, such as in web pages or over networks, by making the image file smaller.

How?

- ❖ Compression techniques typically involve
  - ➢ Removing redundancy in the image data, such as repeating patterns or colors, using algorithms like Huffman coding, Run-Length Encoding (RLE), or more advanced ones like Discrete Cosine Transform (DCT) for JPEG images etc,.

high-frequency, low-perceptibility details.



There are two main types of image compression:

Lossy Compression:

- Some amount of the image/text data is discarded to achieve higher compression ratios.
- The original image/text quality is lost, and it may not be fully recoverable.
- The file size can be significantly reduced.
- A well-known example is JPEG format, RLE, details that are less perceptible to the human eye (like small color variations) are removed to reduce the file size.

Lossless Compression:

- No data is lost.
- Original image can be perfectly reconstructed from the compressed file.
- The compression ratios tend to be lower than those of lossy compression, but the image quality remains intact.
- Common lossless formats include PNG and TIFF

- **Definition of Data:**
  - Data is a collection of facts, such as numbers, text, or images, which can be processed and analyzed.
- **Raw Data:**
  - Unprocessed, unmodified data in its original format.
- **Compressed Data:**
  - Data that has been encoded or processed to reduce its size for storage or transmission.

Encryption ≠ Encoding

Some Image (data) types such as TIFF are good for printing while JPG or PNG, are best for the web.

- **Raw Data Characteristics:**
    - Original, unaltered data.
    - No compression applied.
    - Usually requires more storage space.
    -
- Examples:
    - Uncompressed image files (e.g., BMP, RAW formats).
    - Text files, sensor data, or unprocessed video/audio files.

Compressed Data Characteristics:

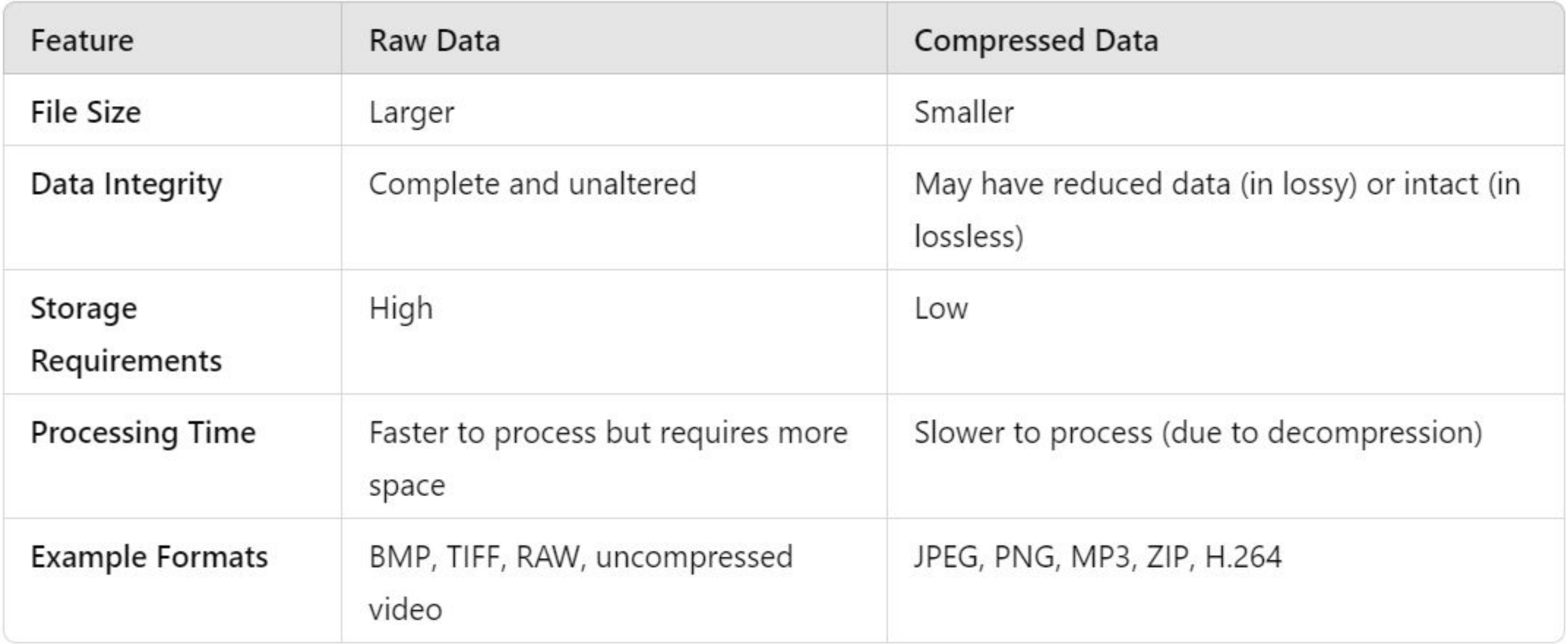
- Data that has been encoded or compressed using compression algorithms.
- Occupies less storage space.
- Can be either lossless or lossy compression.

Examples:

JPEG images (lossy compression).

PNG images (lossless compression).

ZIP files, MP3, MP4.



Feature	Raw Data	Compressed Data
File Size	Larger	Smaller
Data Integrity	Complete and unaltered	May have reduced data (in lossy) or intact (in lossless)
Storage Requirements	High	Low
Processing Time	Faster to process but requires more space	Slower to process (due to decompression)
Example Formats	BMP, TIFF, RAW, uncompressed video	JPEG, PNG, MP3, ZIP, H.264

When to Use Raw Data:

If data integrity and accuracy are critical.

Not willing to lose any information, ex: remote sensing image, medical images

For professional work requiring editing flexibility.

Data Analysis: When high precision and accuracy are essential (e.g., scientific data, medical imaging).

Editing: For applications where high-quality manipulation is needed (e.g., raw image formats in photography, video editing).

Archiving: Preserving original quality for archival purposes.

When to Use Compressed Data:

When file size is a concern, and slight loss of quality is acceptable.

For storage and efficient sharing or transmission.

Web Content: Compressing images and videos for faster loading on websites.

Streaming: Reducing file sizes for quicker streaming (e.g., MP3 for music, H.264 for video).

File Sharing: Compressing files into formats like ZIP for easier sharing and smaller transfer sizes.

Theoretical Assumptions

- Every compression requires decompression to see original data back
- There are computational costs for both compression and decompression steps
- Computational costs are more expensive than size
- Computational operations demands more space during compression and decompression steps. More than data.
- Depending on the need we select compression technique
- In our case we are not bothering about compression and decompression computation costs

Early Traditional compression algorithms

RLE: Useful when there are long sequence of repeating values.

Long sequence/repeating values take less bits

Uncompressed Data: AAAAAAABBBBBCCCCCCCC

Size: 7*65 + 7*66 + 8*67=

7*7bits + 7*7bits + 8*7bits = 154 bits = 19.25Bytes

Compressed data: 7A7B8C

Size: 3bits+7bits + 3bits+7bits + 4+7bits = 31 bits

Drawbacks: Not useful when there are no long sequence.

XYZ = 1X1Y1Z

Run-Length Coding classified as a **lossless compression technique**?

What types of data are best suited for Run-Length Coding? Why?

Ex:

Apply Run-Length Coding to the following and find Compression Ratio :

Q1. 5 5 5 5 3 3 8 8 8 1

Q2. 000011110000111

Q3

1 1 1 0 0

1 0 0 0 0

1 1 1 1 1

Q4 Decode the following RLC stream and reconstruct the original data:

(A,4), (B,2), (C,5)

Uncompressed

aaaaabbbbbbbbbbbbccccddddddddddeeeeeeeeee

Compressed

5a12b4c9d10e

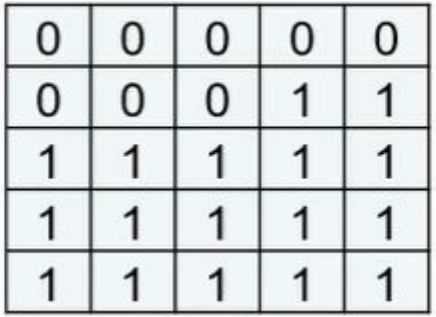
Uncompressed

aabccdeefghijjjklmnopqrrstttuvvwwxyyyz

Compressed

2a1b2c1d2e1f1g1i3j1k1l1m1n1o1p1q2r1s3t1u2v2w1x3y1z

Ex: Given a sample Binary image, apply run length encoding?



0	0	0	0	0
0	0	0	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Horizontal RLC:

Run length Vectors:

Row 1: (0,5)

Row2: (0,3) (1,2)

Row3: (0,5)

Row4: (0,5)

Row5: (0,5)

No of bits per Pixel: 1 bit

No of Vectors: 6

Max Length: 5 (3bits)

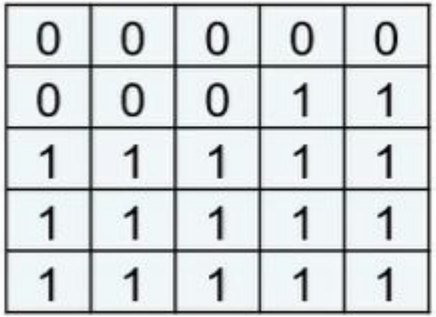
Total No of Pixels: 6*(3+1) = 24

Total No of Pixels in original Image = 5*5 = 25

Compression Ratio: 25/24 = 1.042:1

Positive, Compression achieved

Ex: Given a sample Binary image, apply run length encoding?



0	0	0	0	0
0	0	0	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Veritical RLC:

Run length Vectors:

column 1: (0,2) (1,3)

Column 2: (0,2) (1,3)

Column 3: (0,2) (1,3)

column 4: (0,1) (1,4)

Column 5: (0,1) (1,4)

No of bits per Pixel: 1 bit

No of Vectors: 10

Max Length: 4 (3bits)

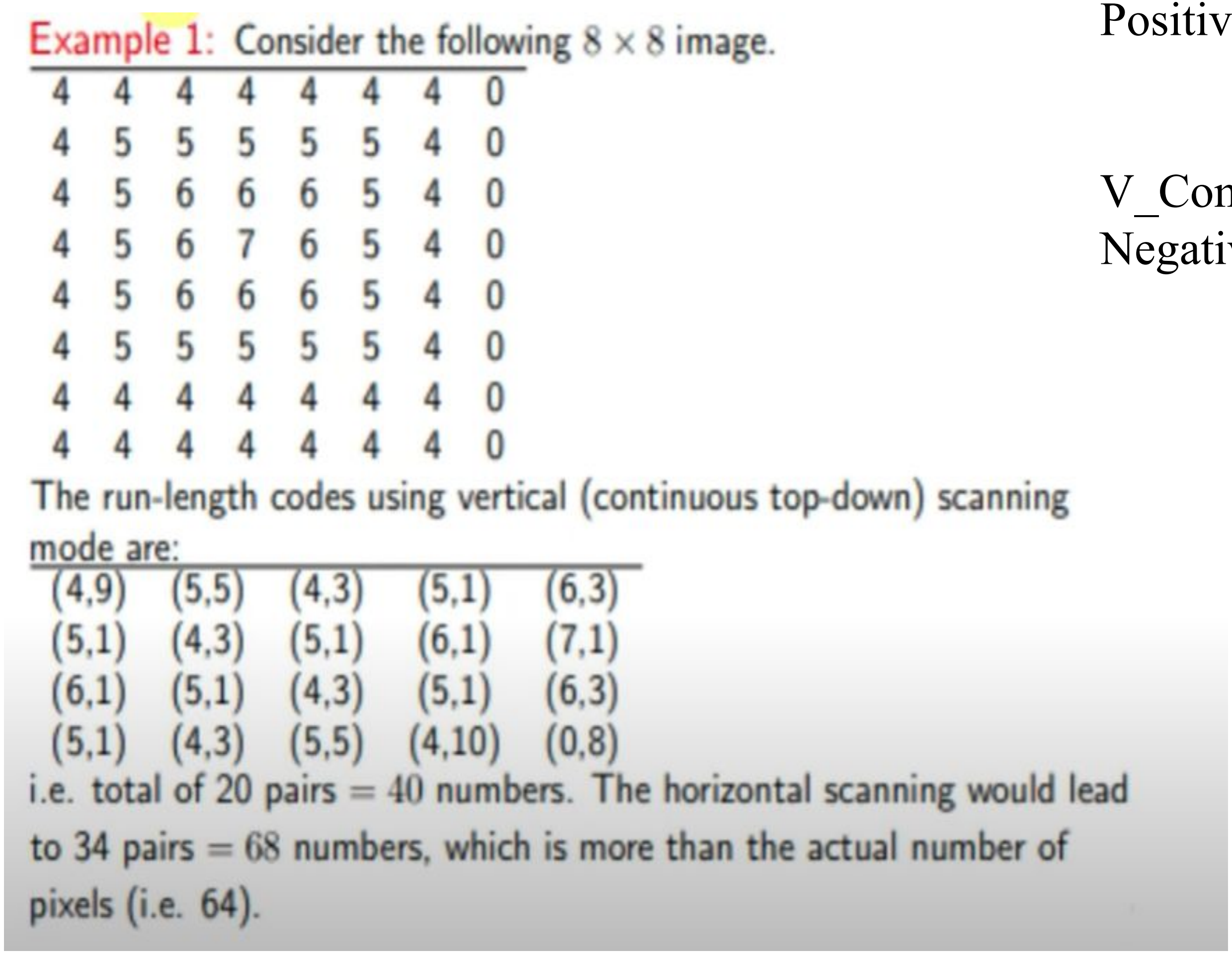
Total No of Pixels: 10*(3+1) = 40

Total No of Pixels in original Image = 5*5 = 25

Compression Ratio: 25/40 = 0.625:1

Negative, NO COMPRESSION ACHIEVED

Analyze



Example 1: Consider the following 8 × 8 image.

4	4	4	4	4	4	4	0
4	5	5	5	5	5	4	0
4	5	6	6	6	5	4	0
4	5	6	7	6	5	4	0
4	5	6	6	6	5	4	0
4	5	5	5	5	5	4	0
4	4	4	4	4	4	4	0
4	4	4	4	4	4	4	0

The run-length codes using vertical (continuous top-down) scanning mode are:

(4,9)	(5,5)	(4,3)	(5,1)	(6,3)
(5,1)	(4,3)	(5,1)	(6,1)	(7,1)
(6,1)	(5,1)	(4,3)	(5,1)	(6,3)
(5,1)	(4,3)	(5,5)	(4,10)	(0,8)

i.e. total of 20 pairs = 40 numbers. The horizontal scanning would lead to 34 pairs = 68 numbers, which is more than the actual number of pixels (i.e. 64).

H_Compression Ratio = 64/40=1.6:1
Positive Compression

V_Compression Ratio = 64/68=0.941:1
Negative Compression

Huffman Encoding and Decoding

It is a lossless data compression algorithm.

Purpose: To reduce the size of data by using variable-length codes for characters based on their frequencies.

Key Idea: More frequent characters are assigned shorter codes, and less frequent characters are assigned longer codes.

File Compression

- Used to compress text files
- Frequently occurring characters → shorter codes
- ZIP files (with LZ + Huffman)
- Huffman coding is used in the final stage of JPEG
- MP3
- AAC
- MPEG audio standards
- Compresses frequency-domain coefficients efficiently
- MPEG-2 (DVD)
- H.264 / AVC
- H.265 / HEVC (partially)

Practical Assumptions

- Without redundancy, compression is not possible (Shannon limit)
- Computational Resources Are Available
- Encoding happens once, decoding happens many times.
- Encoder may be computationally complex and Decoder is usually simpler
- Compression gain > overhead cost
- Encoding algorithms store intermediate results in: RAM, stack, temporary structures
- Both encoder and decoder must agree on: Codebook, Tree structure, Tables or models. Otherwise decoding is impossible
- Input symbols/data come from a finite set and symbols are well-defined

Example of Huffman Encoding

How Huffman Encoding Works

Calculate Frequency: Count the frequency of each character in the data.

Generate Codes: Traverse the binary tree to assign binary codes to each character.

Example of Huffman Encoding **Fixed size code**

Input String: "*BCCABBDDAECCBBAEDDCC* "   20*8 = 160

Frequency of Characters:

Symbol	**count**	**code**
A	3	000
B	5	001
C	6	010
D	4	011
E	2	100

Total 5 Symbols, at max 3 Bits Required

20 * 3 = 60 Bits for message

5 * **8** = 40   Bits for symbols

5 * 3 = 15   Bits of code

A ascii code is 65, requires **8** Bits

Total Bits = 60 + 55 = 115 Bits

Compression Ratio = 160/115=1.391:1   Positive

Example of Huffman Encoding **variable size code**

How Huffman Encoding Works

Calculate Frequency: Count the frequency of each character in the data.

Optimal merge pattern: Insert all characters into a priority queue (or min-heap), ordered by frequency.

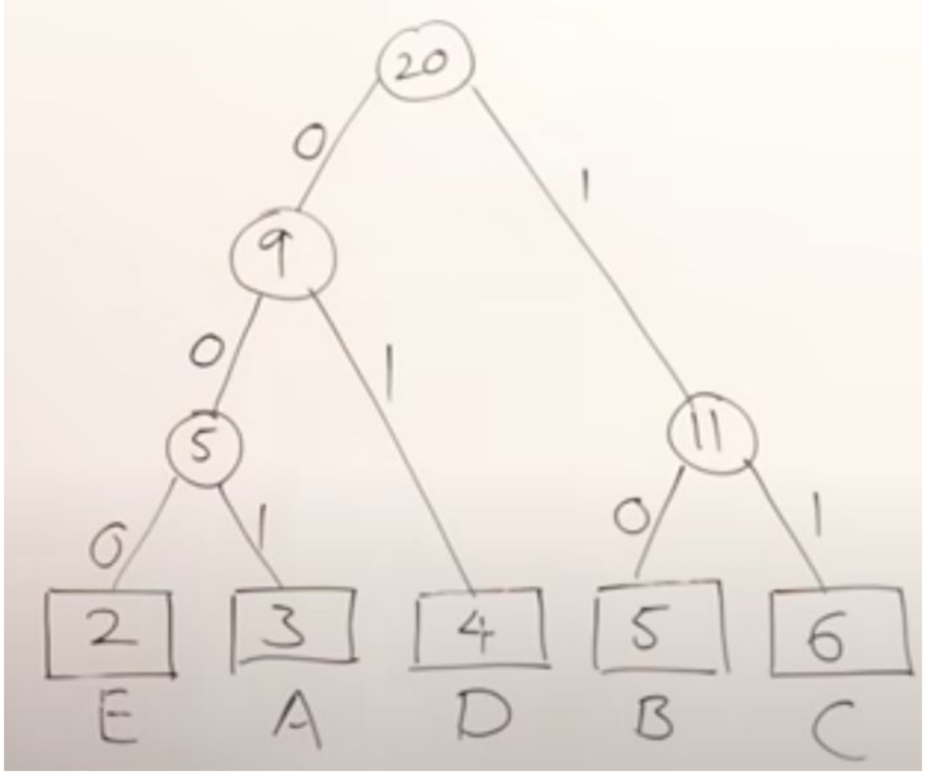
Create Binary Tree: Build a binary tree by repeatedly combining the two least frequent characters or nodes.

Generate Codes: Traverse the binary tree to assign binary codes to each character.

Example of Huffman Encoding **Variable size code**

Input String: "BCCABBDDAECCBBAEDDCC" 20*8 = 160

Symbol	count	Binary code	Count * Code
A	3	001	3 * 3 = 9
B	5	10	5 * 2 = 10
C	6	11	6 * 2 = 12
D	4	01	4 * 2 = 8
E	2	000	2 * 3 = 6
		12 Bits	45 Bits



Example of Huffman Encoding **Variable size code**

Input String: "BCCABBDDAECCBBAEDDCC"   20*8 = 160

Symbol	**count**	**Binary code**	**Count * Code**
A	3	001	3 * 3 = 9
B	5	10	5 * 2 = 10
C	6	11	6 * 2 = 12
D	4	01	4 * 2 = 8
E	2	000	2 * 3 = 6
		12 Bits	45 Bits

45   Bits for message

5 * **8** = 40   Bits for symbols

12 Bits for code

Total Bits = 45 + 40 + 12 = 97 Bits

Compression Ratio = 160/97 =1.649 : 1 Positive

Huffman examples

https://www.youtube.com/watch?v=yue6yqhmfSg

https://www.youtube.com/watch?v=iiGZ947Tcck

Huffman encoding on images

https://www.youtube.com/watch?v=acEaM2W-Mfw&t=51s

https://www.youtube.com/watch?v=IwNjHmOAduw&t=67s

https://www.youtube.com/watch?v=qPIyFx1fexs

Arithmetic Coding

https://www.youtube.com/watch?v=G6u8bt5unZk

JPEG

- invented by the Joint Photographic Expert Group in the late 1980s

- The goal of this committee was to reduce the file size of natural, photographic-like true-color images as much as possible without affecting the quality of the Image as experienced by the human sensory engine

- Development of JPEG-2000: Standard developing was started at 1996. JPEG2000 is the new JPEG standard which was finally approved in August 2000.

JPEG

- It is the most common format for storing and transmitting photographic images
- It is the most common format for storing and transmitting photographic images on the World Wide Web. These format variations are often not distinguished, and are simply called JPEG.

- It is a commonly used method of lossy compression for digital images particularly for those images produced by digital photography.

- The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality.

- JPEG typically achieves 10:1 compression with little perceptible loss in image quality.

Overview

- Based on Discrete Cosine Transform (DCT):

0) Image is divided into block N×N

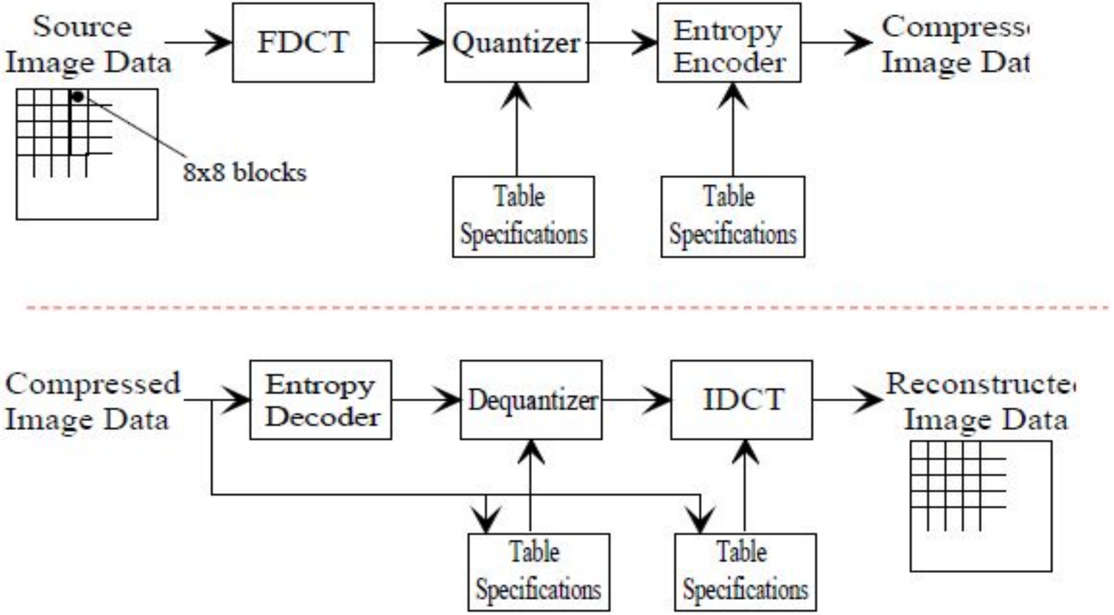
1) The blocks are transformed with 2-D DCT

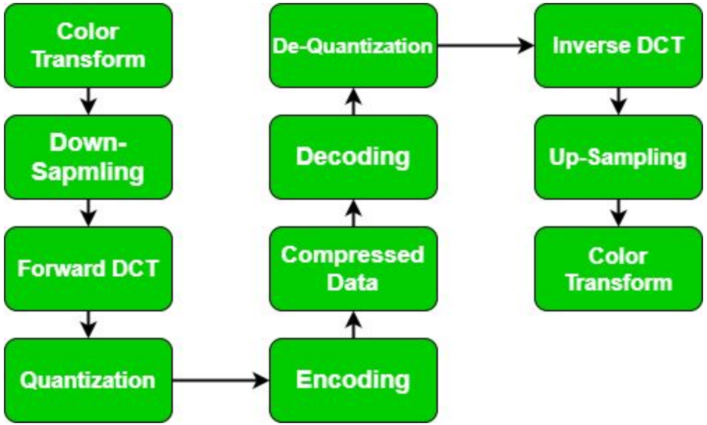
2) DCT coefficients are quantized

3) The quantized coefficients are encoded

G. K. Wallace, “The JPEG Still Picture Compression Standard”, Communications of the ACM, April 1991, vol 34, No. 4, pp 30 - 44

Compression and Decompression steps





Compression steps

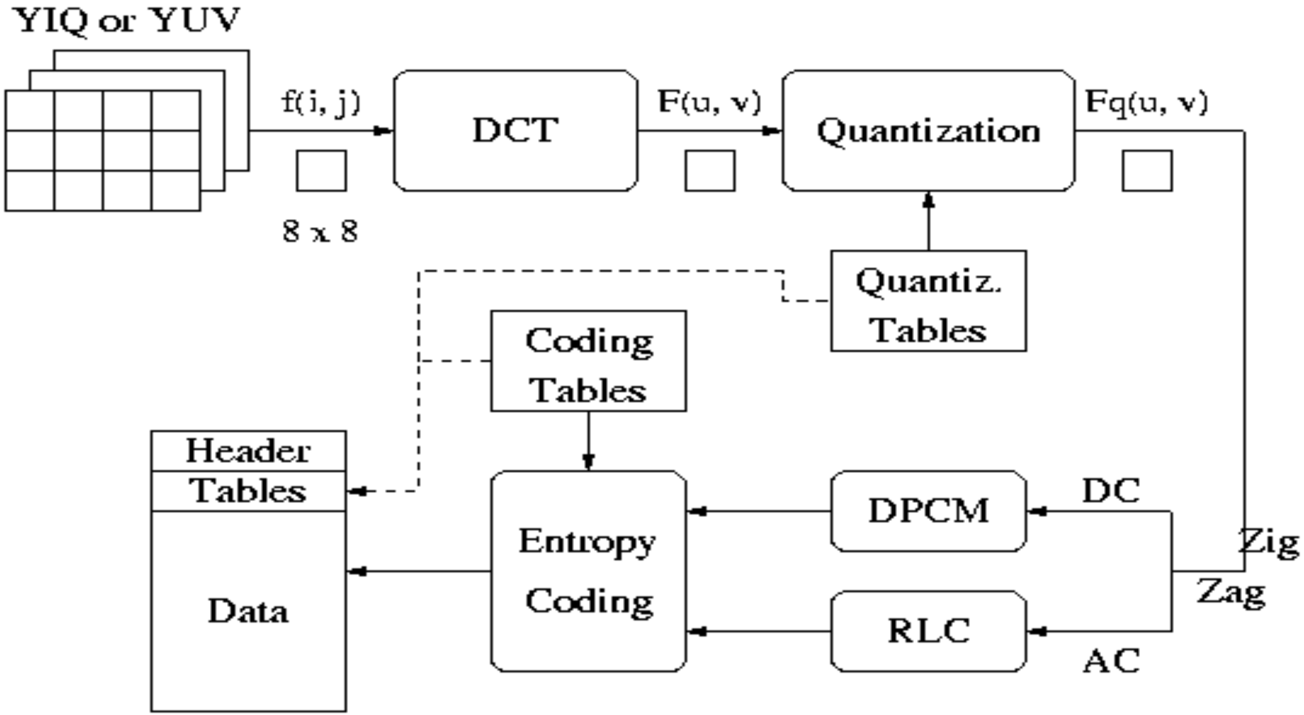


Image pre-processing

- Shift values [0, 2^P-1] to [-2^(P-1), 2^(P-1)-1]
  - e.g. if (P=8), shift [0, 255] to [-127, 127]
- DCT requires range to be centered around 0
- Values in 8x8 pixel blocks are spatial values and there are 64 samples values in each block

Why

DCT works best when input values are centered around 0 because it is designed to efficiently represent variations (AC components), not large constant offsets (DC component).

The Discrete Cosine Transform (DCT) expresses a signal as a sum of cosine waves of different frequencies.

Low-frequency cosines → smooth changes

High-frequency cosines → sharp changes

DC component (0 frequency) → average value of the block

Pixel intensities are in [0, 255], this means the signal has a large positive offset

What happens if we don’t center around 0?

If input is not centered:

DC coefficient becomes very large

AC coefficients become relatively small

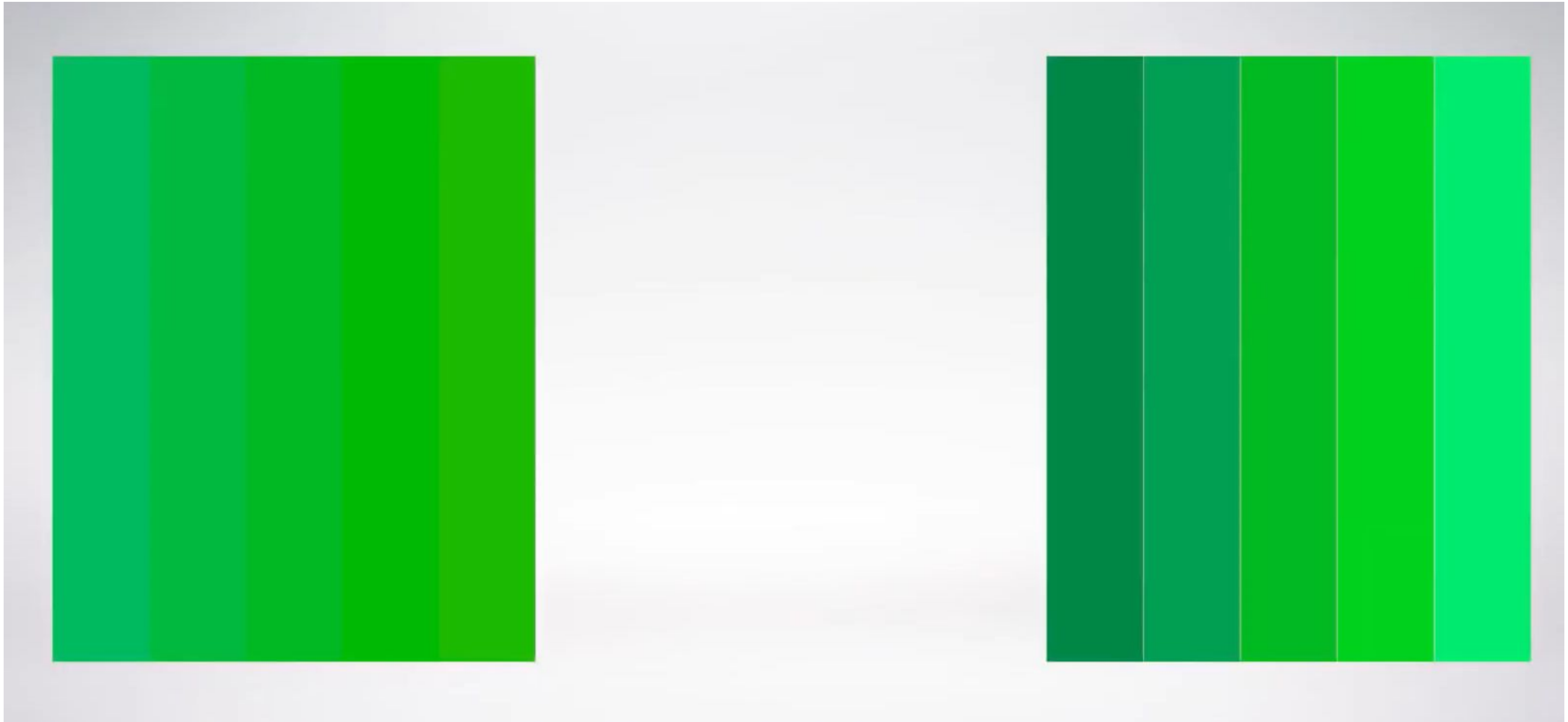
Energy is not well balanced

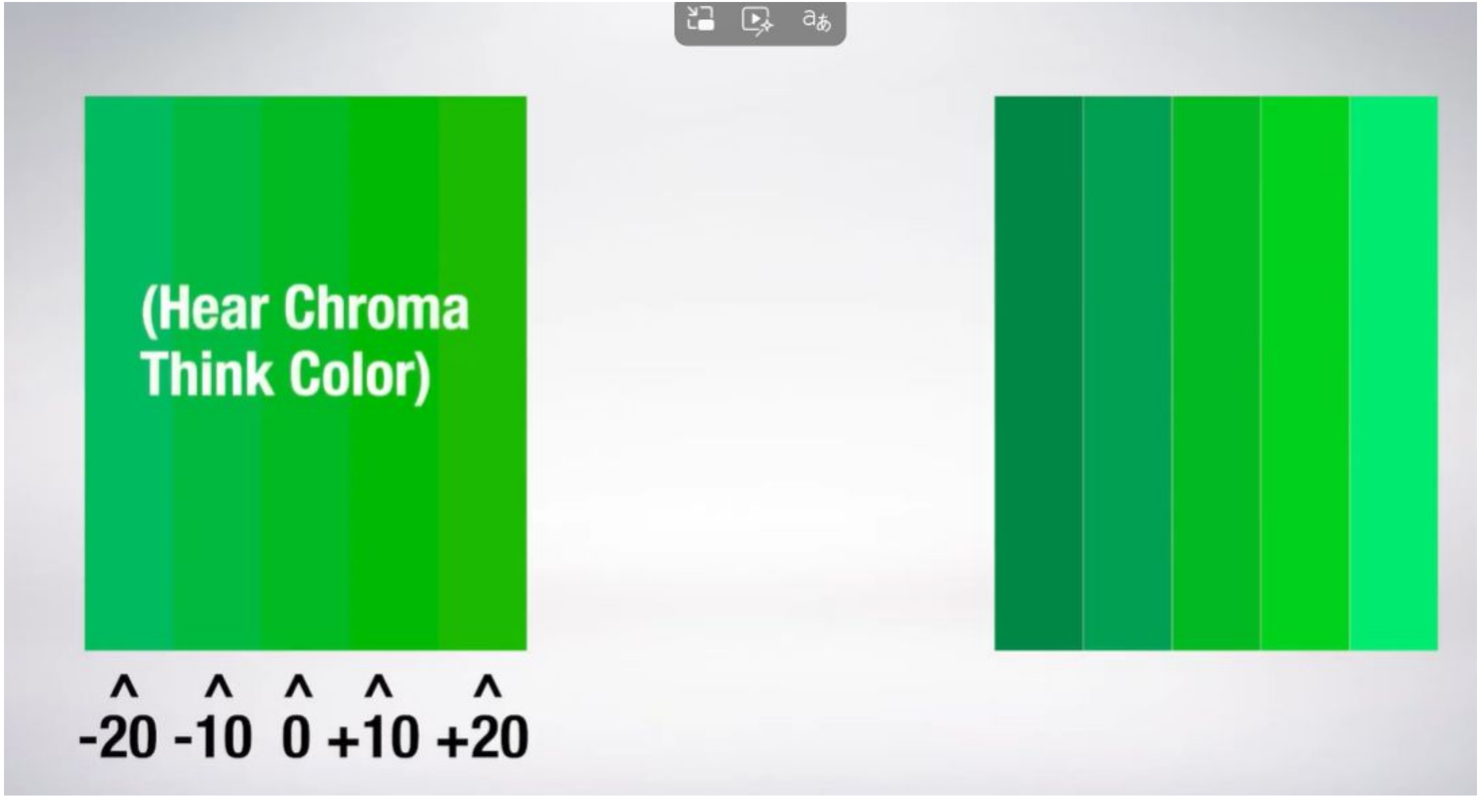
Quantization becomes inefficient

Compression performance degrades

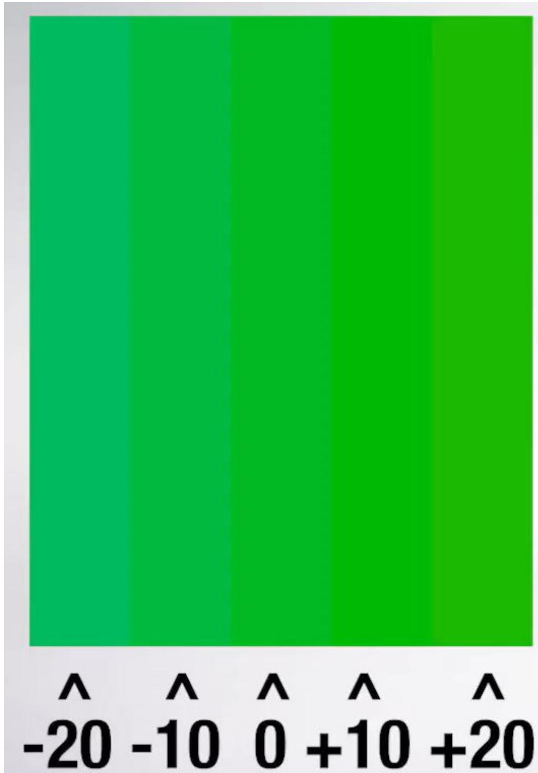
The DCT wastes bits representing a large constant offset.

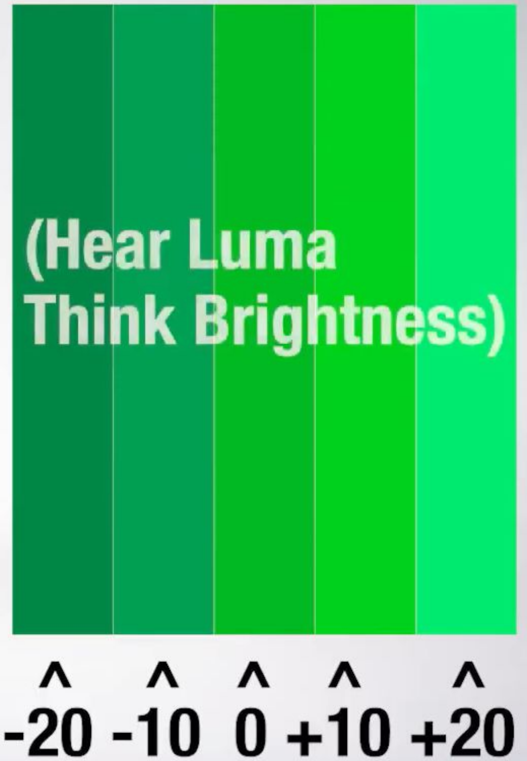
Y vs Cb Cr





(Hear Chroma Think Color)




(Hear Luma
Think Brightness)

RGB vs YCbCr

- Convert RGB into a YCbCr representation:
  - Y is luminance, and Cb, Cr are chrominance
- Downsample the two chrominance components
- 24 bits RGB representation: apply DCT for each component separately

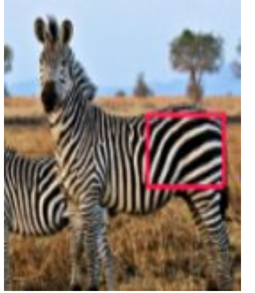
RGB ⇔ YCbCr conversion

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

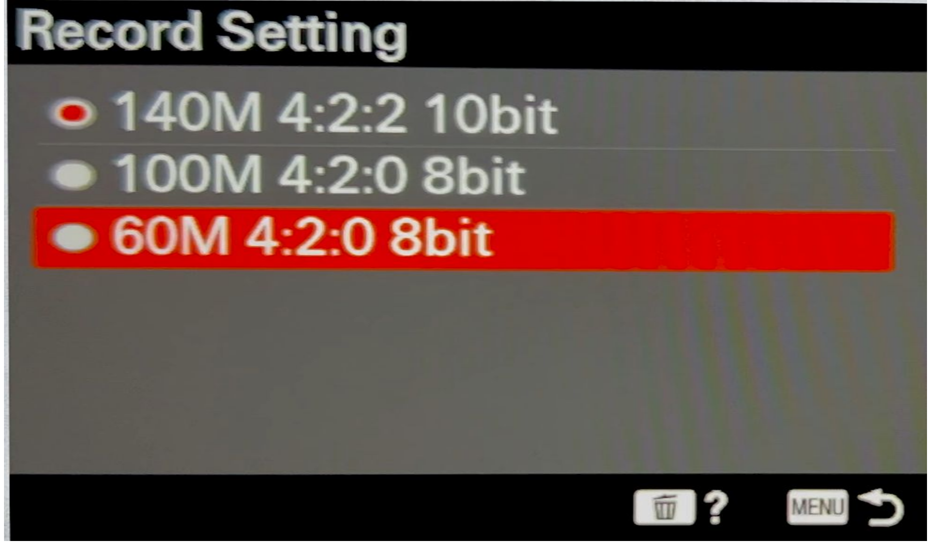
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & -0.001 & 1.402 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.772 & 0.001 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

Luminance Y and two chrominances Cb and Cr

Channel value normalization and chroma Down sampling



- Channel value normalization and sub sampling of the chroma components.
- Normalization can be done by subtracting 128 from every pixel value in image.
- Down sampling means we are not going to get color content for every pixel.
- The algorithm performs down sampling only to the chroma components (Cr and Cb components) and not for the luminance component (Y component).
- This is because the human visual system is more sensitive to luminance than chrominance.
- Therefore, by down sampling, only the chroma components, the algorithm can reduce image size without affecting much of the information it provides to the viewer (picture quality) because a high proportion of that information is contained in only the luminance component (Y component).
- This strategy is called **chroma subsampling** and it is heavily used in other types of image and video processing pipelines also.

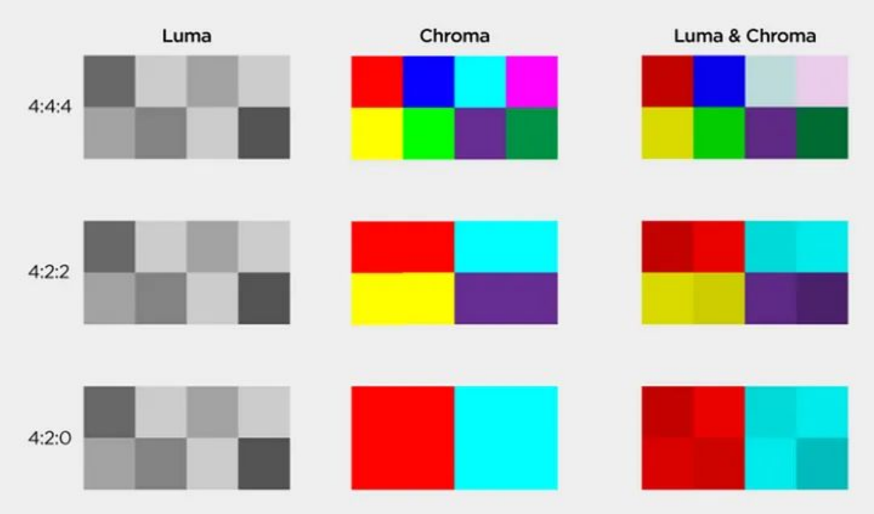


Record Setting

● 140M 4:2:2 10bit

○ 100M 4:2:0 8bit

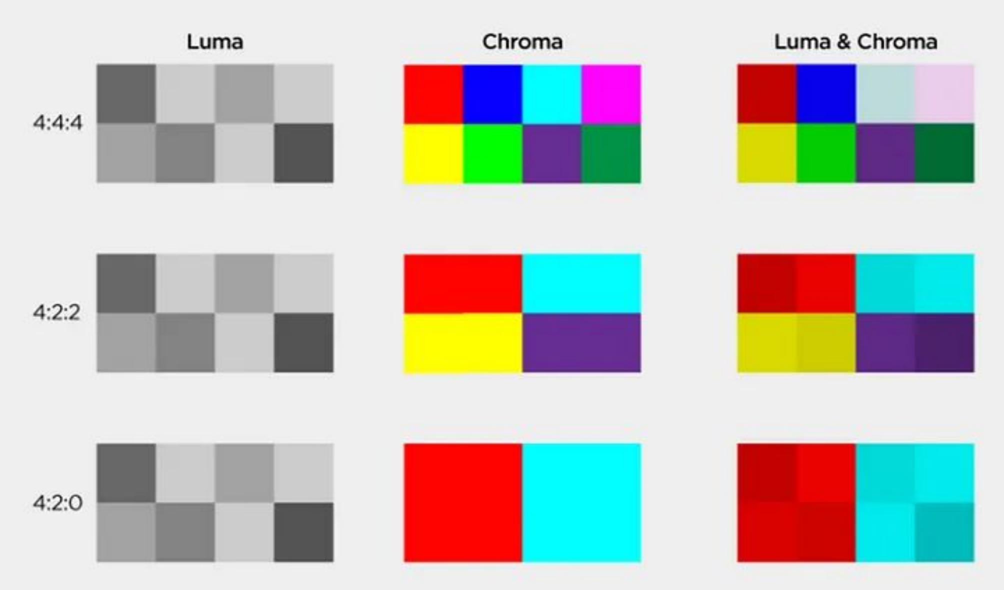
○ 60M 4:2:0 8bit



Format	Samples	Bits/sample	Total bits
4:2:0 8-bit	6	8	48 bits
4:2:0 10-bit	6	10	60 bits
4:2:2 8-bit	8	8	64 bits
4:2:2 10-bit	8	10	80 bits

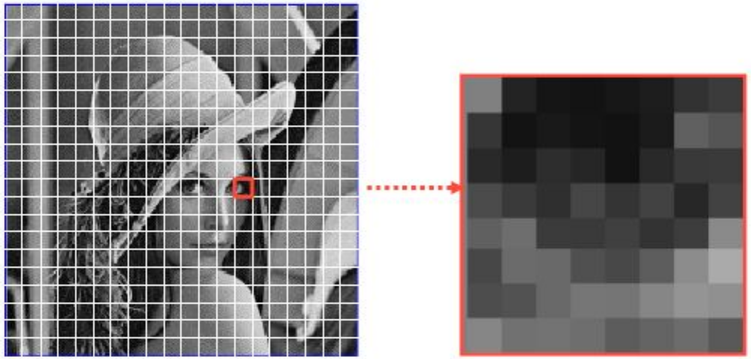
- 4:4:4 — uncompressed image with no chroma subsampling, transports both luminance and color data entirely.

- 4:2:2 — has half of the chroma of 4:4:4 and reduces the size of an uncompressed image signal by one-third with little to no visual difference.
- 4:2:0 — has one-quarter of the chroma of 4:4:4 and reduces the bandwidth of an uncompressed video signal by half compared to no chroma subsampling [1].



[1] “Chroma subsampling,” Haivision, 21-Jun-2019. [Online]. Available: https://www.haivision.com/resources/streaming-video-definitions/chroma-subsampling/#:~:text=Chroma%20subsampling%20is%20a%20type,without%20significantly%20affecting%20picture%20quality.
https://www.geeksforgeeks.org/spatial-resolution-down-sampling-and-up-sampling-in-image-processing/

Divide image into N x N blocks



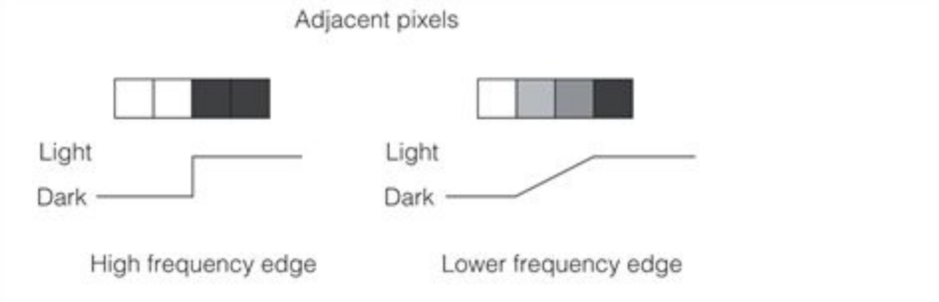
Input image

8x8 block

- Spatial frequency describes the periodic distributions of light and dark in an image.

- High spatial frequencies correspond to features such as **sharp edges and fine details, whereas low spatial frequencies correspond to features such as global shape.**

- frequency-dependent contracts sensitivity characteristic of the human visual system means that **it's easier to miss small objects or finer details in an image as compared to larger ones.**

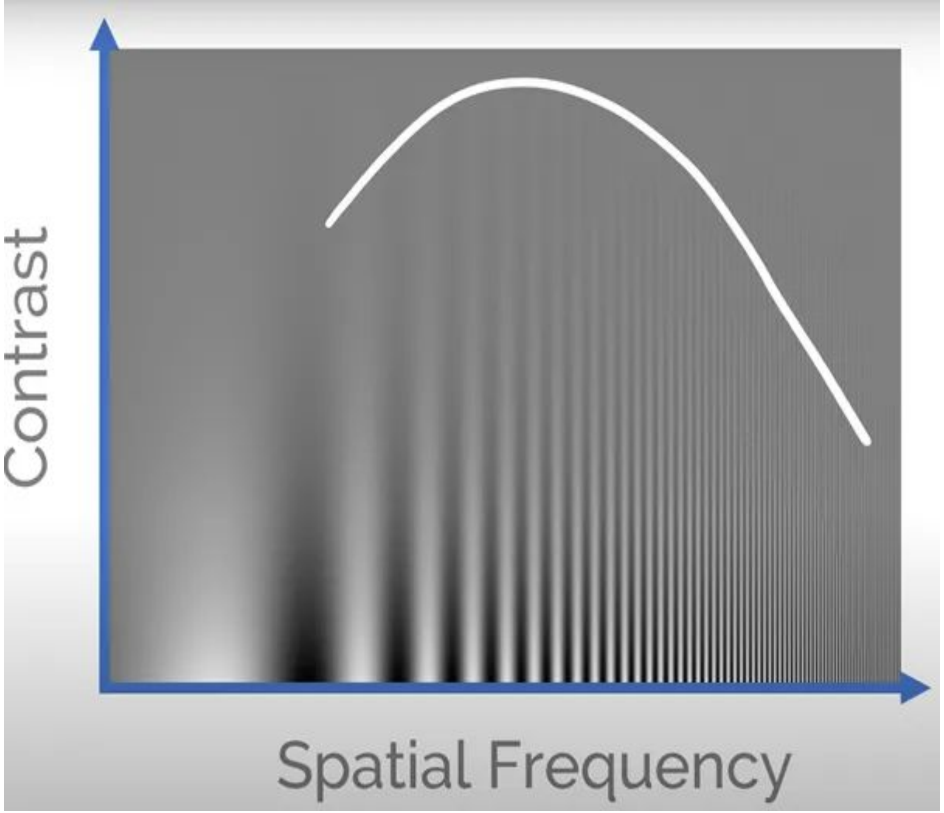


- The bars in figure have increasing spatial frequency from left to right and decreasing contrast from bottom to top.

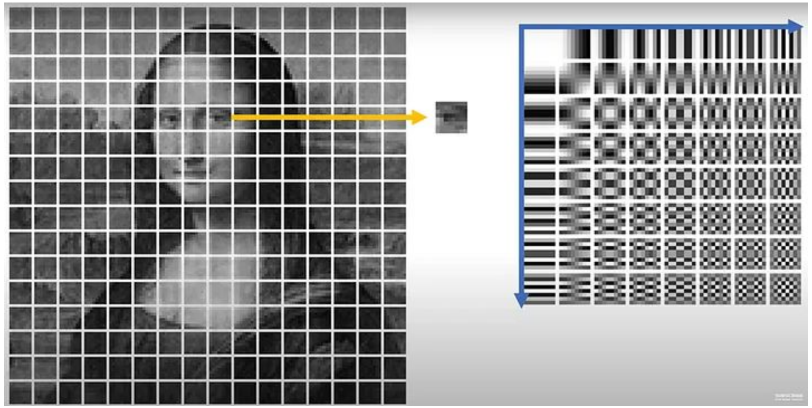
- Due to the 'frequency-dependent contracts sensitivity' phenomena, the bars under the curve (shown in Fig) are more visible than the rest.

- The bars which are located at the low-contrast, high-spatial frequency are barely visible. Therefore, those less visible frequencies give the JPEG algorithm some room for compression.

- The algorithm does that by dividing the image into 8x8 blocks and quantizing them in a frequency domain representation.

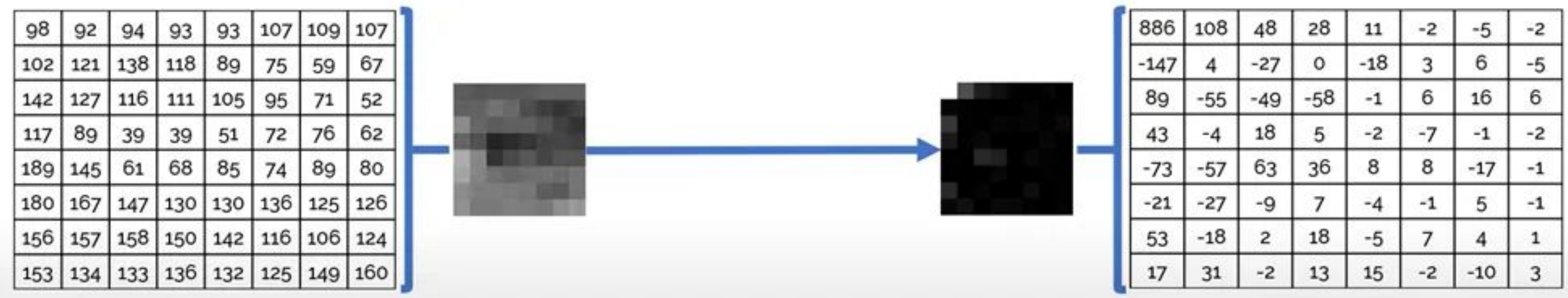


The discrete cosine transformation is then applied to each of the 8x8 blocks.



Each 8x8 pixel is compared with 64 frequency patterns (Fig right-hand side image) where the spatial frequency increases from left to right and top to bottom.

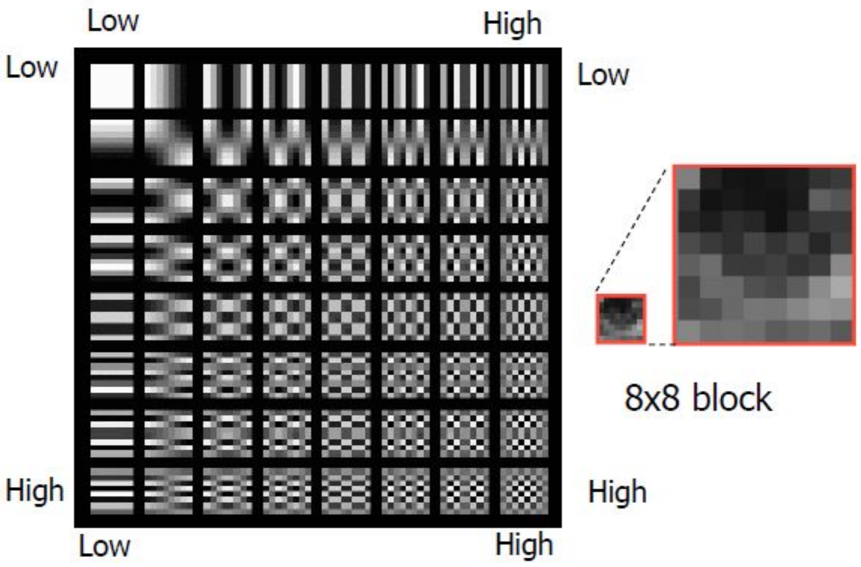
Ultimately each pixel in an 8x8 block which previously represented the brightness of each pixel gets transformed into another 8x8 block in which each pixel represents the particular frequency component (bottom Fig below).



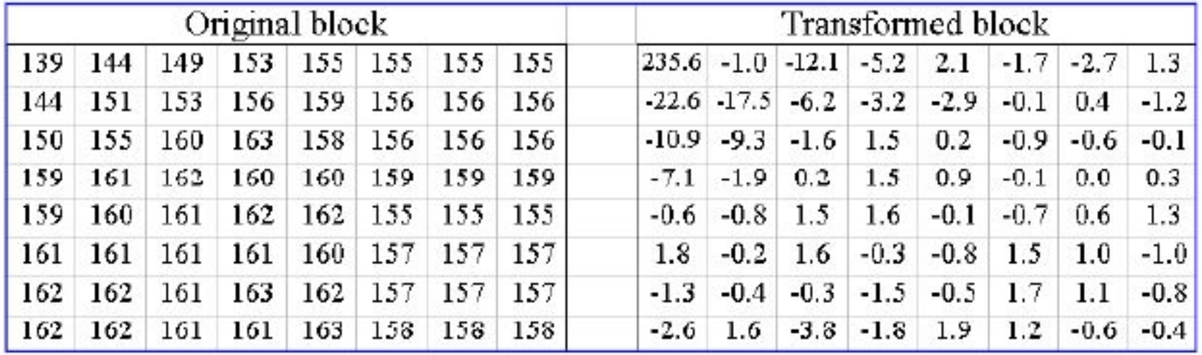
98	92	94	93	93	107	109	107
102	121	138	118	89	75	59	67
142	127	116	111	105	95	71	52
117	89	39	39	51	72	76	62
189	145	61	68	85	74	89	80
180	167	147	130	130	136	125	126
156	157	158	150	142	116	106	124
153	134	133	136	132	125	149	160

886	108	48	28	11	-2	-5	-2
-147	4	-27	0	-18	3	6	-5
89	-55	-49	-58	-1	6	16	6
43	-4	18	5	-2	-7	-1	-2
-73	-57	63	36	8	8	-17	-1
-21	-27	-9	7	-4	-1	5	-1
53	-18	2	18	-5	7	4	1
17	31	-2	13	15	-2	-10	3

2D DCT of image blocks & Basis Image



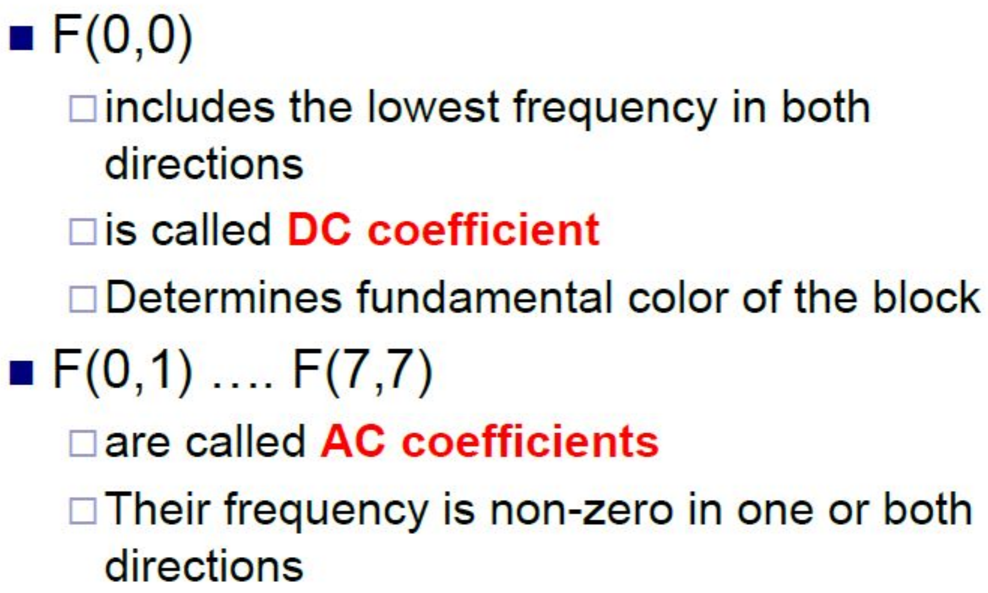
Example of DCT of image block



Original block									Transformed block							
139	144	149	153	155	155	155	155		235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
144	151	153	156	159	156	156	156		-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
150	155	160	163	158	156	156	156		-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
159	161	162	160	160	159	159	159		-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
159	160	161	162	162	155	155	155		-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
161	161	161	161	160	157	157	157		1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
162	162	161	163	162	157	157	157		-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
162	162	161	161	163	158	158	158		-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Matlab: y=dct(x)

Coefficient Differentiation

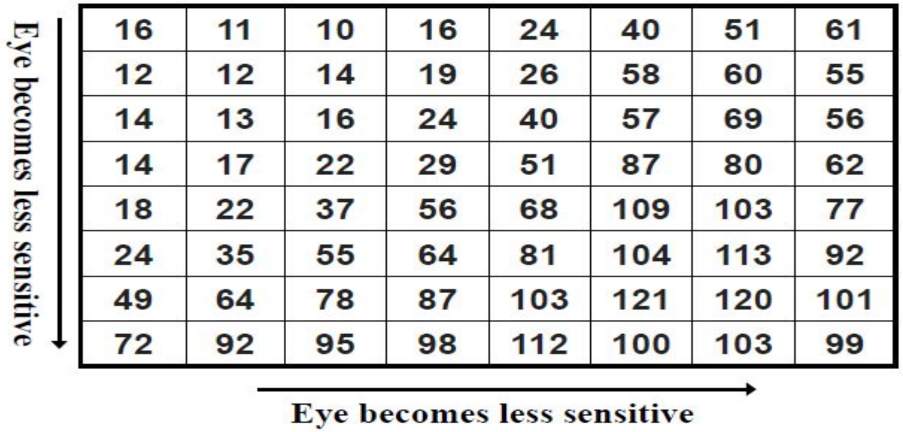


- F(0,0)
  - includes the lowest frequency in both directions
  - is called **DC coefficient**
  - Determines fundamental color of the block
- F(0,1) …. F(7,7)
  - are called **AC coefficients**
  - Their frequency is non-zero in one or both directions

Quantization

- In JPEG: quantization is achieved by dividing DCT coefficients using quantization tables $Fq(u,v) = F(u,v)/Quv$

De facto Quantization Table

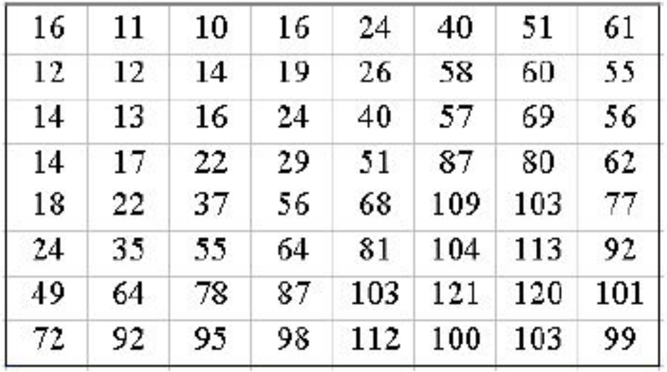


16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Eye becomes less sensitive

Eye becomes less sensitive

Default quantization matrix



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

$$\boxed{y_q(k,l)=\text{round}[y(k,l)/Q(k,l)]}$$

Examples: 236/16 → 15
-22/11 → -2

Matlab: Qy=quant (y)

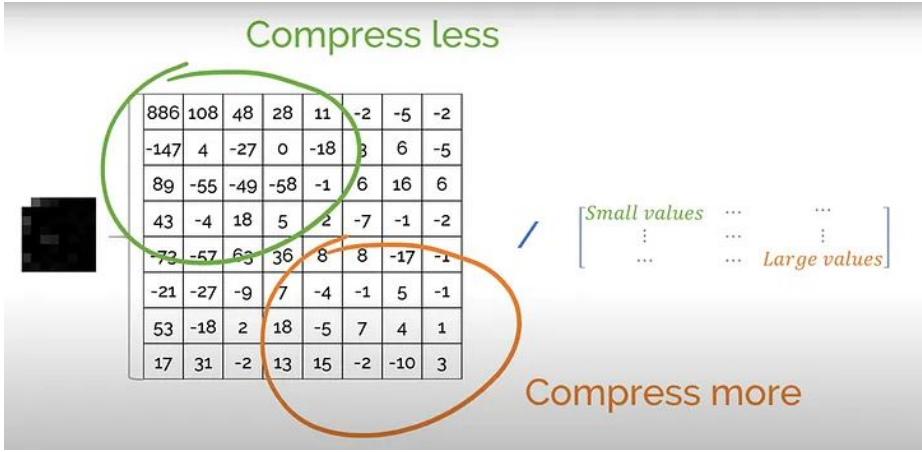
Quantization:

- The next step is quantization where the frequencies that are less visible to the human eye (which is spread more towards the bottom right corner of the image as shown in Fig) will be compressed more by dividing the frequency values with some constants.

- To achieve this, the frequency components that are less sensitive to the human visual system get divided by larger constants as compared to the ones that humans are more sensitive to.

- After diving the result gets rounded to the nearest integers and this makes most of the high-frequency component (spread in the bottom right-hand corner) values become zero.

- This process achieves high compression rates but at the expense of image quality (lossy compression).

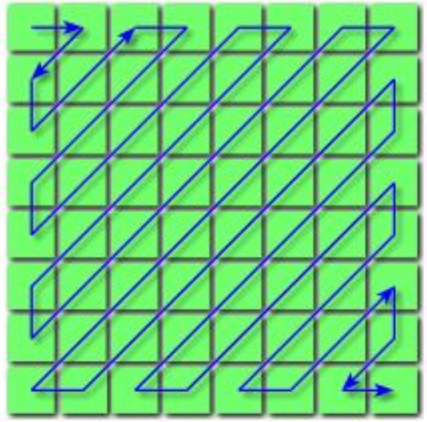


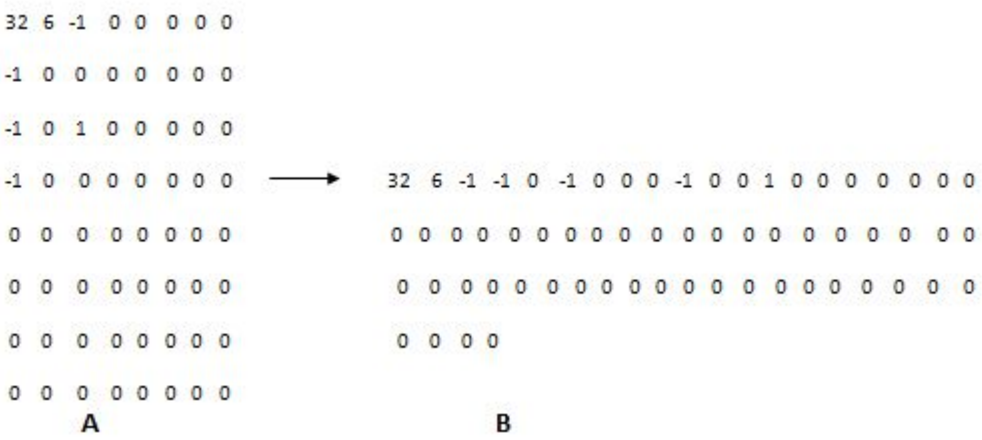
Compress less

886	108	48	28	11	-2	-5	-2
-147	4	-27	0	-18	3	6	-5
89	-55	-49	-58	-1	6	16	6
43	-4	18	5	2	-7	-1	-2
-73	-57	63	36	8	8	-17	-1
-21	-27	-9	7	-4	-1	5	-1
53	-18	2	18	-5	7	4	1
17	31	-2	13	15	-2	-10	3

Compress more

Zig Zag ordering of DCT coefficients





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

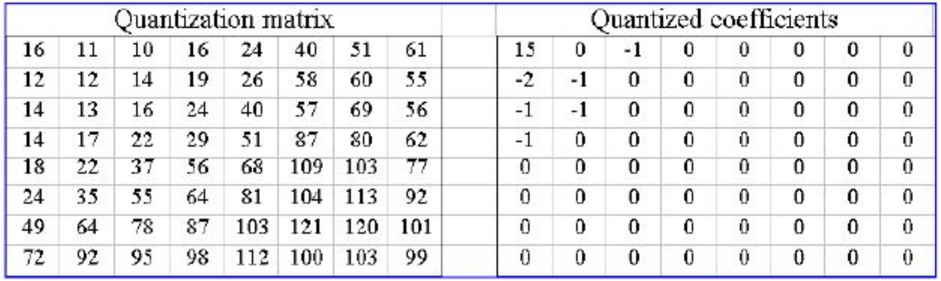
A

```
32 6 -1 -1 0 -1 0 0 0 -1 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 0 0 0 0 0 0 0 0
0 0 0 0
```

B

A) Quantization of DCT coefficients, B) zigzag order.

- Human vision is less sensitive to high frequencies
- Zigzag pushes high-frequency coefficients (often zero) to the end
- These can be aggressively compressed or discarded
- Zigzag ordering is used to arrange DCT coefficients from low frequency to high frequency so that zeros are grouped together, enabling efficient run-length encoding.
- Energy compaction in DCT
- After DCT + quantization:
  - Low-frequency coefficients → large, important
  - High-frequency coefficients → mostly zero
  - JPEG wants important values first, zeros later

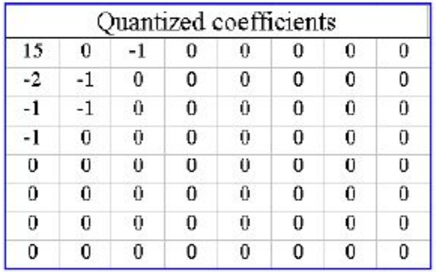


Quantization matrix							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantized coefficients							
15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	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

Ordered DCT coefficients: 15,0,-2,-1,-1,-1,0,0,-1,-1, 54{'0'}.

Encoding of quantized DCT coefficients



Quantized coefficients							
15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	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

- Ordered data: 15,0,-2,-1,-1,-1,0,0,-1,-1, 54{'0'}.
- Encoding:
  - DC: ?
  - AC: ?

Encoding of quantized DCT coefficients
Entropy encoding

- Further compression is achieved by encoding the sequence of quantized DC and AC coefficients using entropy coding techniques, without introducing any additional loss.
- Separate DC from AC components, as DC components change slowly, thus will be encoded using difference encoding.
- DPCM in JPEG: Encodes the difference between DC coefficients of consecutive blocks to exploit spatial redundancy.
- RLC in JPEG: Encodes runs of zero AC coefficients after zigzag scanning to reduce redundancy.

DPCM

Assumed quantized DC coefficients of consecutive 8×8 blocks:

DC stream = [120, 125, 123, 130, 128]

DPCM rule (JPEG):

- ★ First DC is coded w.r.t 0
- ★ Others are coded w.r.t previous DC

$DPCM(i)=DC(i)-DC(i-1)$

$[120-0,\ 125-120,\ 123-125,\ 130-123,\ 128-130, \ldots.] \ = [120,\ +5,\ -2,\ +7,\ -2,\ \ldots]$

- ★ Smaller values stream leads to better Huffman coding
- ★ Lossless

AC Encoding

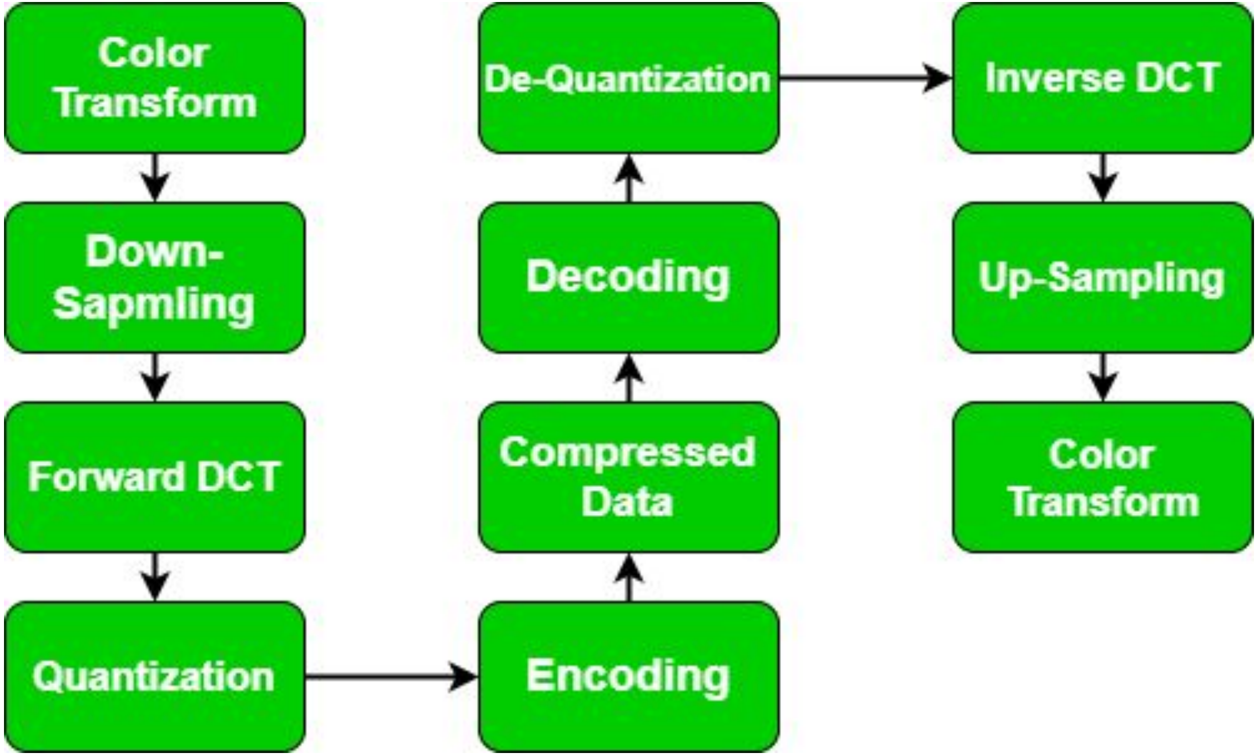
Assumed zigzag-ordered AC coefficients (DC excluded):

AC stream = [4, 0, 0, -3, 0, 0, 0, 2, 0, 0, 0, 0]

(Number of zeros before non-zero, Non-zero value)

(0,4), (2,-3), (3,2), EOB

EOB (End of Block) indicates all remaining coefficients are zero.



Color Transform

Down-Sapmling

Forward DCT

Quantization

Encoding

Compressed Data

Decoding

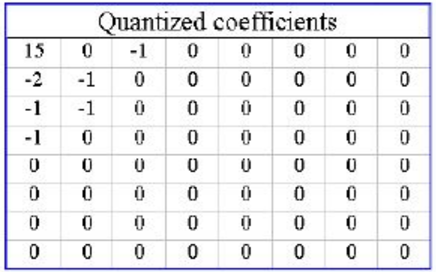
De-Quantization

Inverse DCT

Up-Sampling

Color Transform

Dequantization

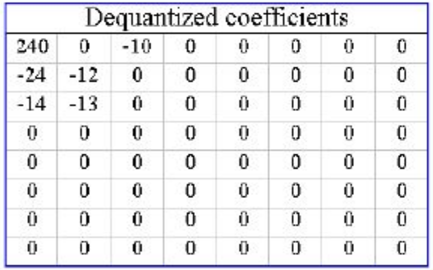


Quantized coefficients							
15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

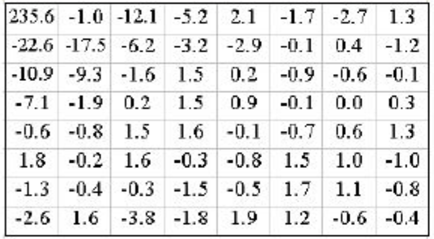
$$\boxed{z(k,l)=y_q(k,l)\cdot Q(k,l)}$$

Examples: 236/16 → 15
-22/11 → -2

Matlab: z=dequant (Qy)



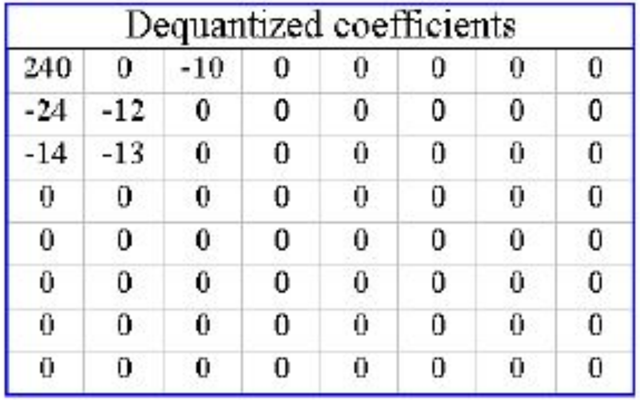
Dequantized coefficients							
240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



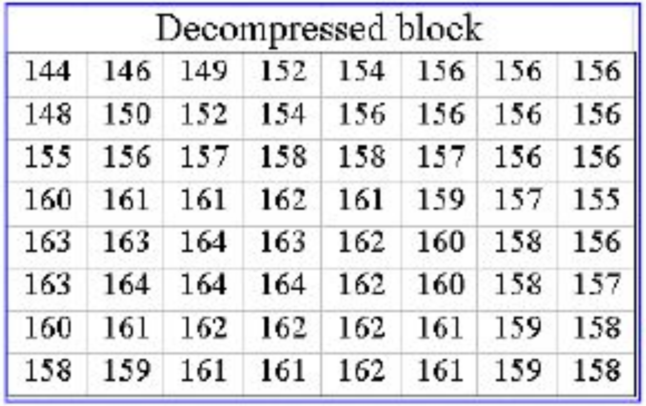
235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Original DCT block

Inverse DCT

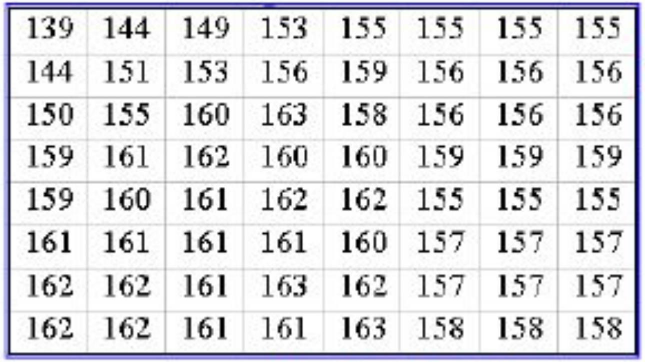


Dequantized coefficients							
240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Decompressed block							
144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	156
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

See: x=idct(y)



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

Original block

Assignment 01

Implement JPEG algorithm

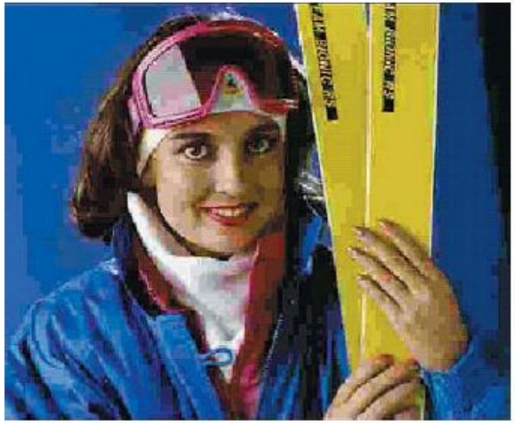
Ideas

Neural image codecs (e.g., Learned compression)

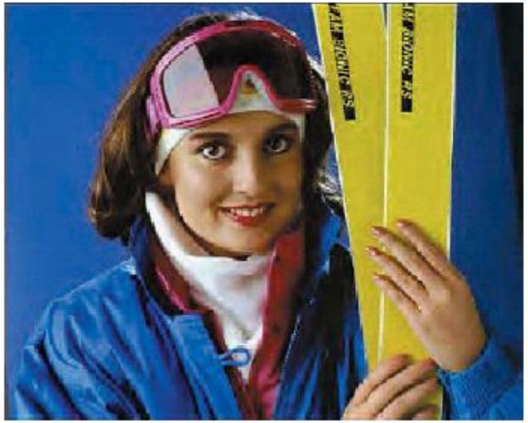
DCT coefficients fed into CNNs

Learned quantization instead of fixed tables

JPEG ⇒JPEG2000

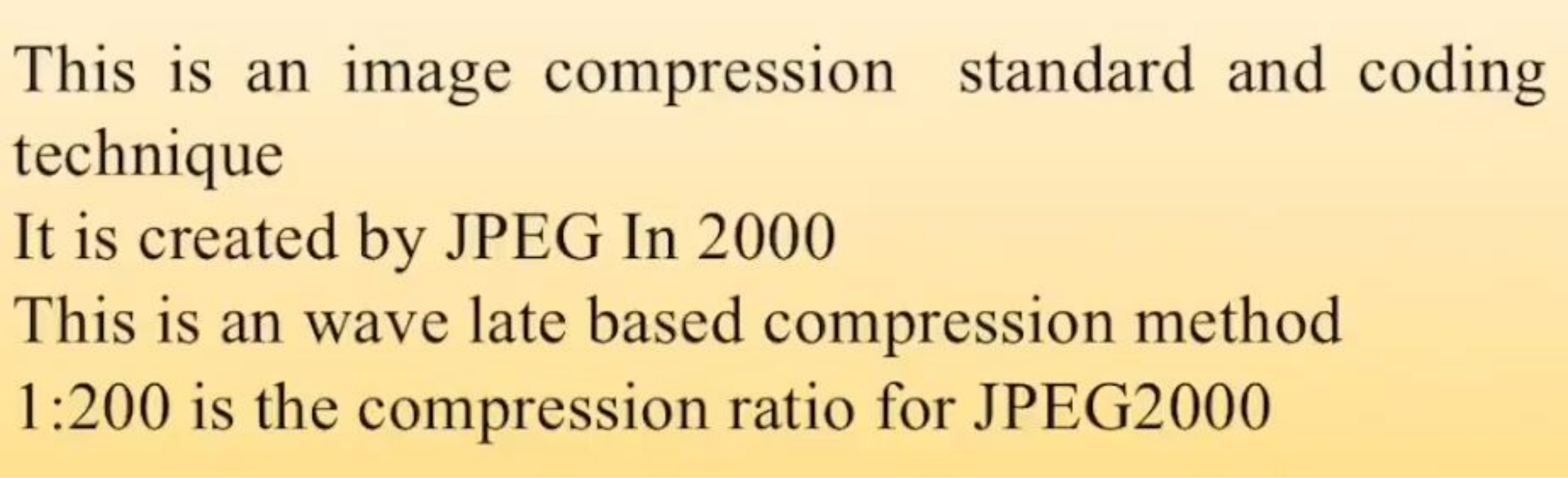


JPEG: 0.25 bpp



JPEG2000: 0.25 bpp

What is JPEG2000



This is an image compression standard and coding technique

It is created by JPEG In 2000

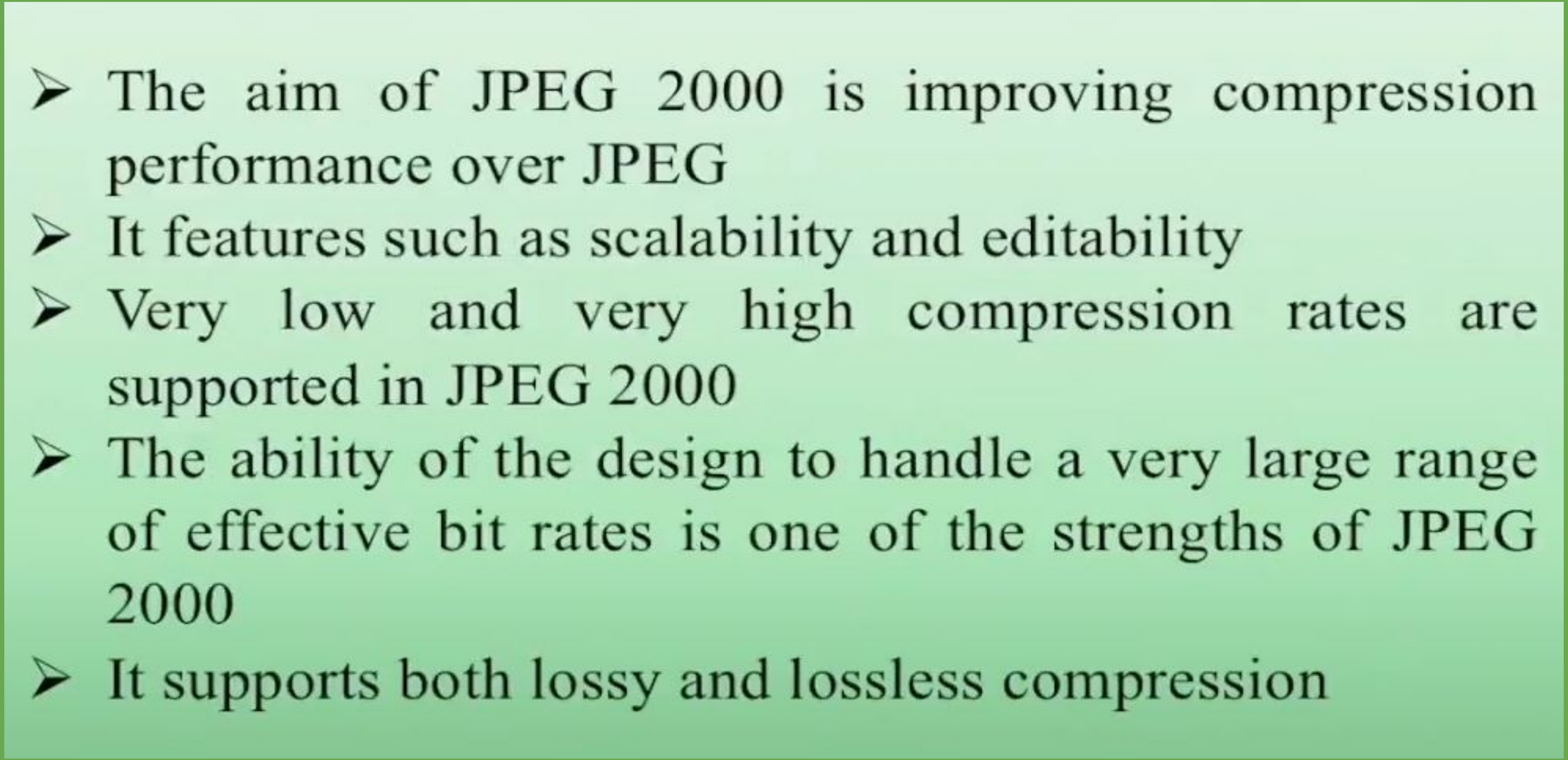
This is an wave late based compression method

1:200 is the compression ratio for JPEG2000

JPEG2000 1 to 38 bit per component

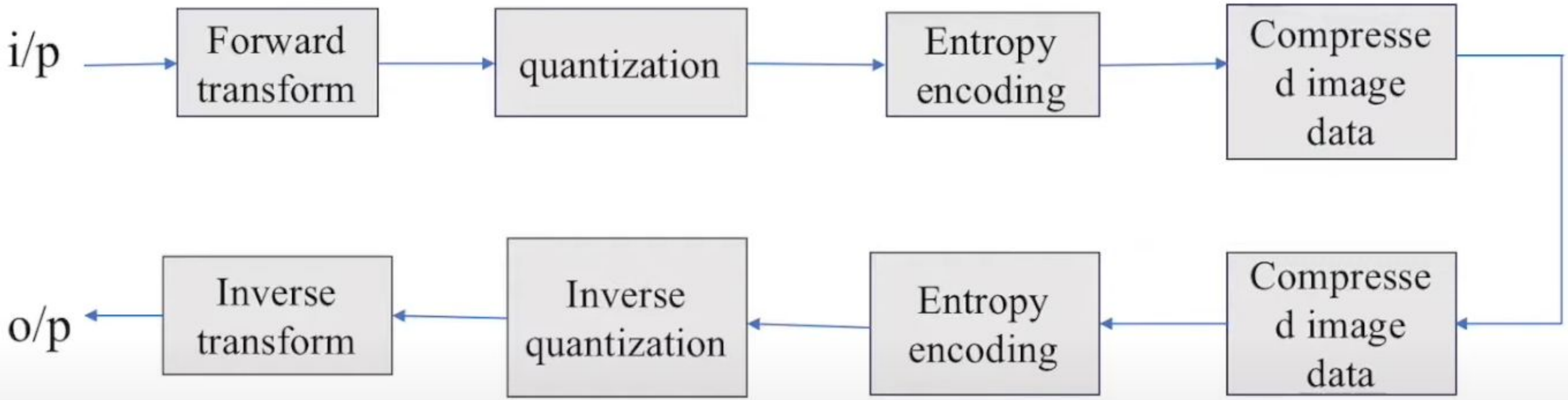
JPEG 12 bit

Why JPEG2000

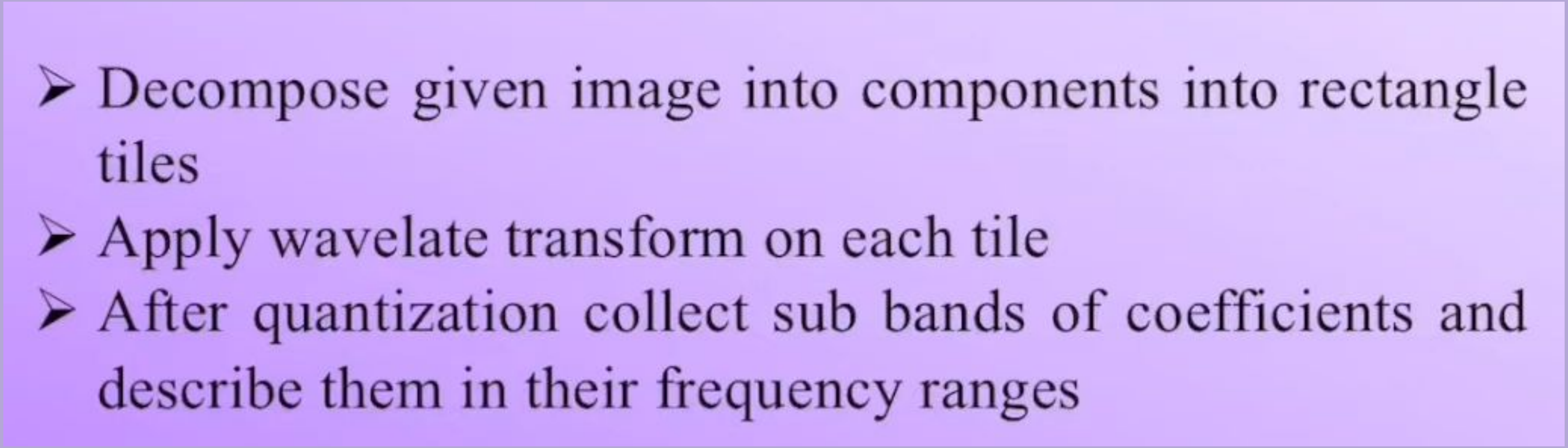


- The aim of JPEG 2000 is improving compression performance over JPEG
- It features such as scalability and editability
- Very low and very high compression rates are supported in JPEG 2000
- The ability of the design to handle a very large range of effective bit rates is one of the strengths of JPEG 2000
- It supports both lossy and lossless compression

Over all Steps of JPEG2000



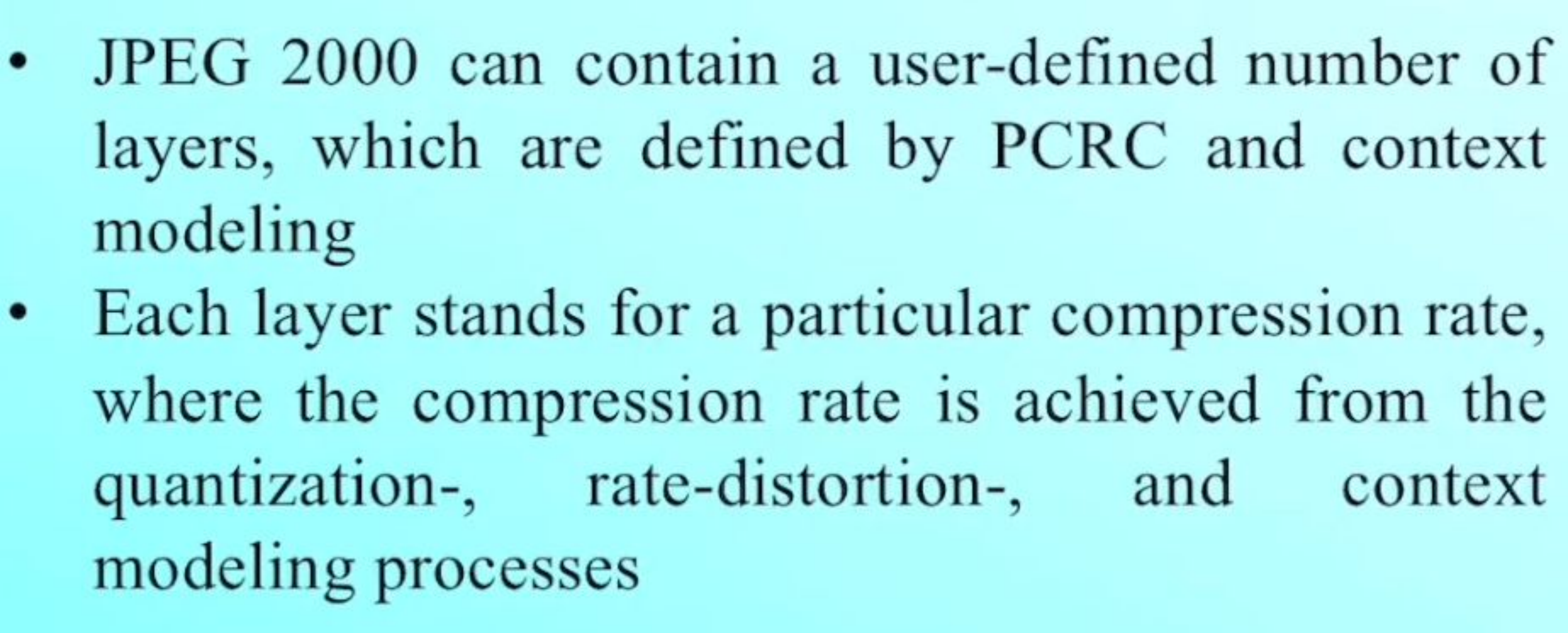
Theory



- Decompose given image into components into rectangle tiles
- Apply wavelate transform on each tile
- After quantization collect sub bands of coefficients and describe them in their frequency ranges

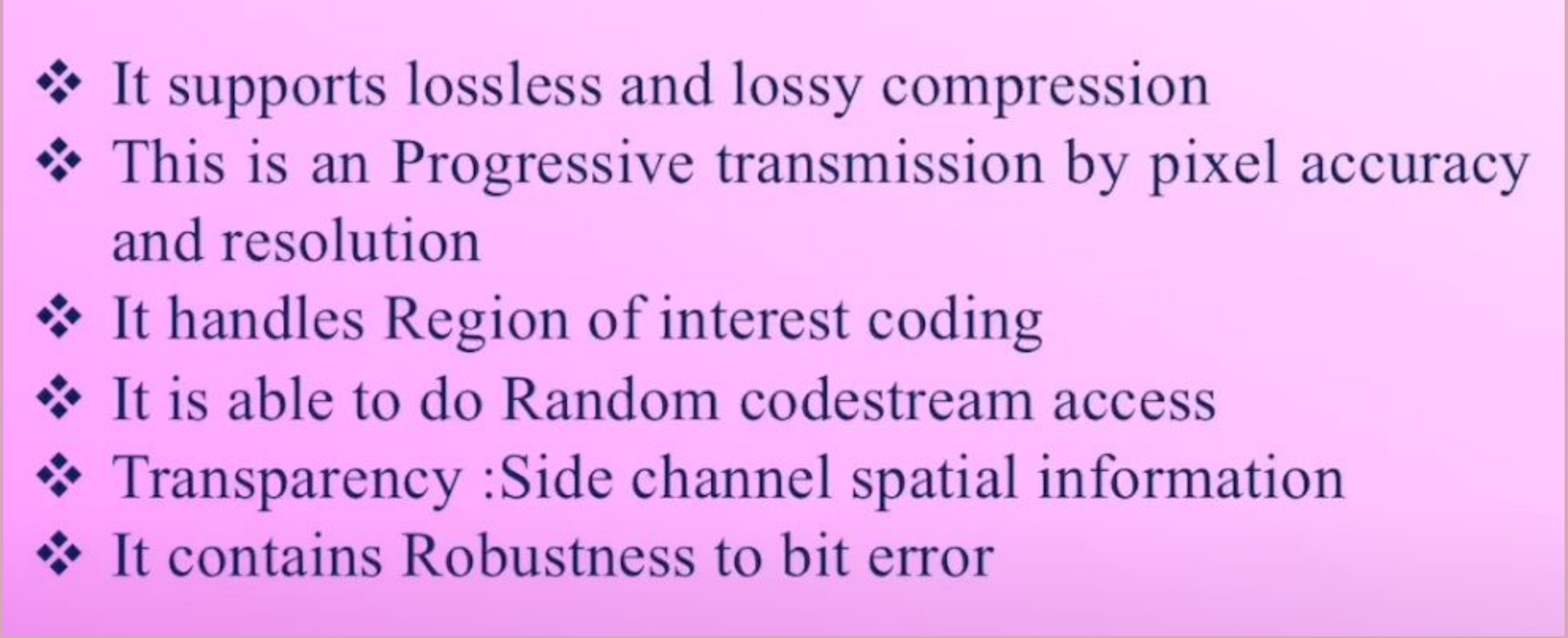
- Given input image or part of the image known as tile is sent to a set of wavelet filters, which transform the pixel information into wavelet coefficients,
- In the next step they are then grouped into several *subbands*
- Each subband contains wavelet coefficients that describe a specific horizontal and vertical spatial frequency range of the entire original image
- This means that lower-frequency, less-detailed information is contained in the first transform level, while more-detailed, higher-frequency information is contained in higher transform levels

- Equally sized code blocks, which are essentially bit streams of data, are generated within each subband
- This break-down is necessary for coefficient modeling and coding, and is done on a code-block-by-code-block basis
- the code blocks can then be grouped according to their significance
- On the decoding side it is then possible to extract information according to its significance, allowing the most significant information to be seen first



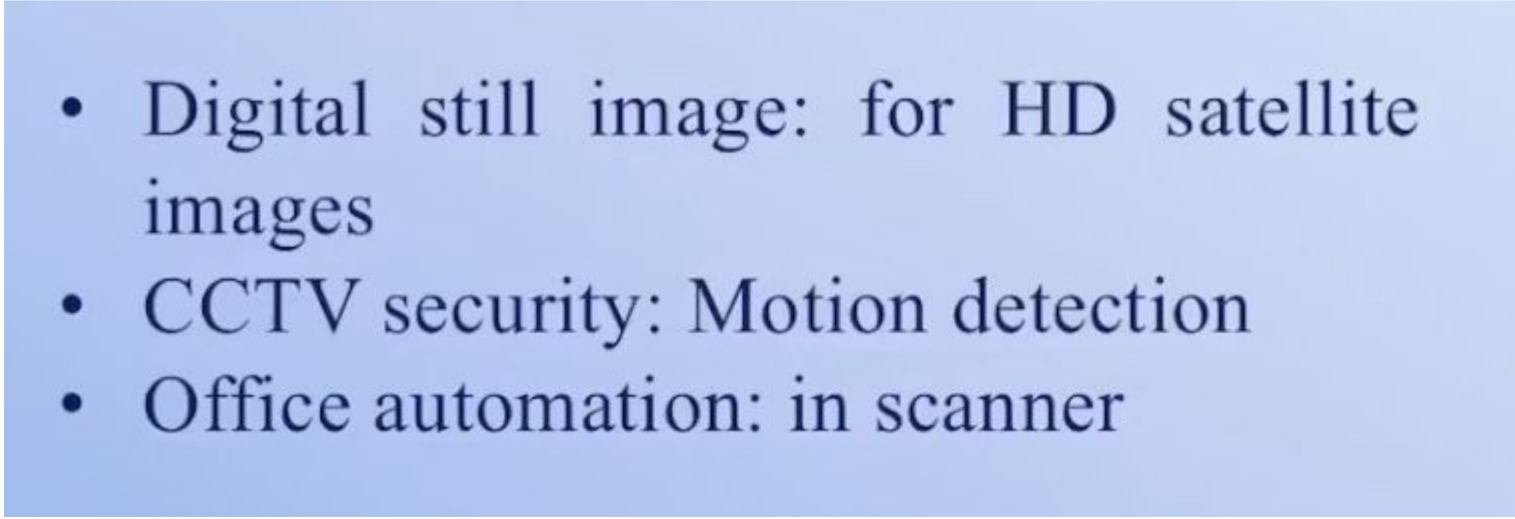
- JPEG 2000 can contain a user-defined number of layers, which are defined by PCRC and context modeling
- Each layer stands for a particular compression rate, where the compression rate is achieved from the quantization-, rate-distortion-, and context modeling processes

Advantages of JPEG2000



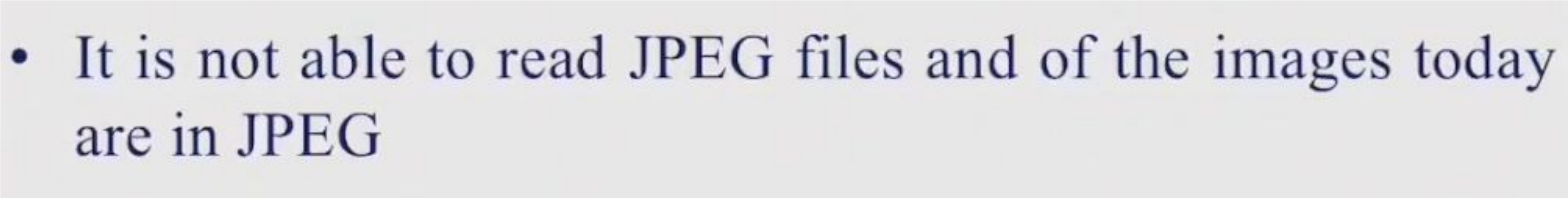
- ❖ It supports lossless and lossy compression
- ❖ This is an Progressive transmission by pixel accuracy and resolution
- ❖ It handles Region of interest coding
- ❖ It is able to do Random codestream access
- ❖ Transparency :Side channel spatial information
- ❖ It contains Robustness to bit error

Applications of JPEG 2000:



- Digital still image: for HD satellite images
- CCTV security: Motion detection
- Office automation: in scanner

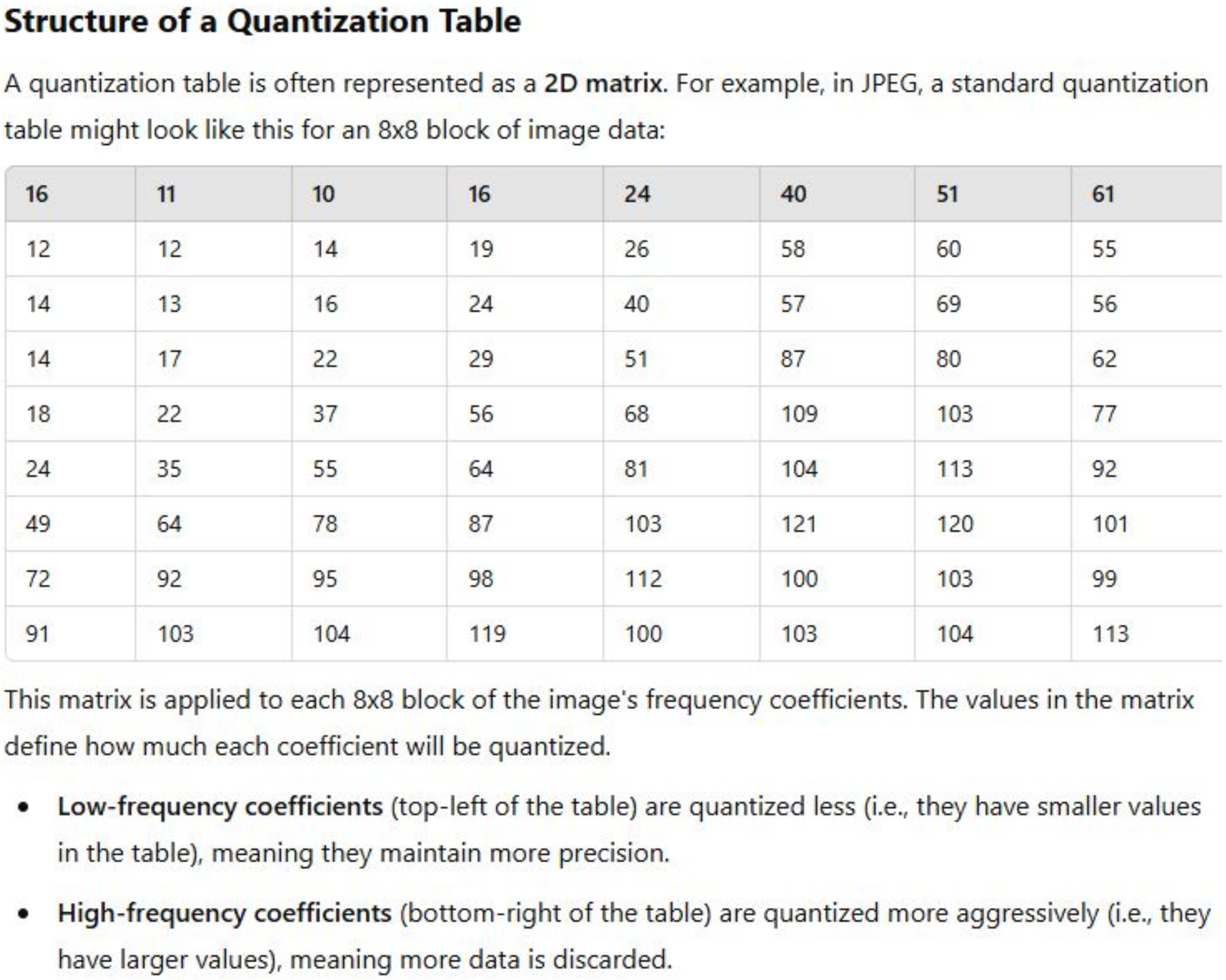
Drawback of JPEG 2000:



- It is not able to read JPEG files and of the images today are in JPEG

JPEG2000 key features

- Transform: Wavelet, Wavelet packet, Wavelet in tiles
- Quantization: Scalar (with dead zones)
- Entropy coding: EBCOT(Embedded Block Coding with Optimized Truncation)
- Region of interest (ROI)
- Lossless color transform



Structure of a Quantization Table

A quantization table is often represented as a **2D matrix**. For example, in JPEG, a standard quantization table might look like this for an 8x8 block of image data:

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
91	103	104	119	100	103	104	113

This matrix is applied to each 8x8 block of the image's frequency coefficients. The values in the matrix define how much each coefficient will be quantized.

- **Low-frequency coefficients** (top-left of the table) are quantized less (i.e., they have smaller values in the table), meaning they maintain more precision.
- **High-frequency coefficients** (bottom-right of the table) are quantized more aggressively (i.e., they have larger values), meaning more data is discarded.

Division by Quantization Table:

- The quantization table defines a series of values that are used to divide each of the transformed coefficients. The table typically contains values that vary depending on the frequency of the coefficients.
- Higher frequency coefficients (which represent finer details in the image) are often given larger values in the quantization table, meaning they will be quantized more aggressively (resulting in more loss).

Rounding the Results:

- After dividing the coefficients by the values in the quantization table, the results are rounded to the nearest integer. This rounding step leads to a loss of precision and contributes to compression.
- The more aggressive the quantization (i.e., the larger the values in the table), the higher the compression but the greater the loss in image quality.

Bit-Plane Decomposition

- Describe bit-plane decomposition and its role in organizing the quantized coefficients.
- Provide a visual showing the bit planes of a sample coefficient.

Bit-Plane Decomposition in Image Compression

Bit-plane decomposition is a technique used in image processing and compression where an image (or more specifically, the image's pixel values or coefficients) is broken down into individual bit planes. Each bit plane represents one of the bits in the binary representation of the pixel values or coefficients.

What is Bit-Plane Decomposition?

In any digital image, the pixel values are represented in binary form. For example, in an 8-bit grayscale image, each pixel value is represented by 8 bits (from 00000000 to 11111111), where each bit represents a different level of significance (from the least significant bit (LSB) to the most significant bit (MSB)).

Bit-plane decomposition involves separating these bits into multiple planes, each corresponding to one specific bit position in the binary representation of the pixel value.

These planes represent the significance of each individual bit (from MSB to LSB) for every pixel in the image.

For an 8-bit image, the bit planes would look as follows:

Bit plane 0: Represents the least significant bit (LSB).
Bit plane 1: Represents the second least significant bit.
Bit plane 2: Represents the third least significant bit.
...
Bit plane 7: Represents the most significant bit (MSB).

How Bit-Plane Decomposition Works:

Convert each pixel to binary: Each pixel in the image is represented by a binary number. In an 8-bit grayscale image, this means each pixel has 8 bits.

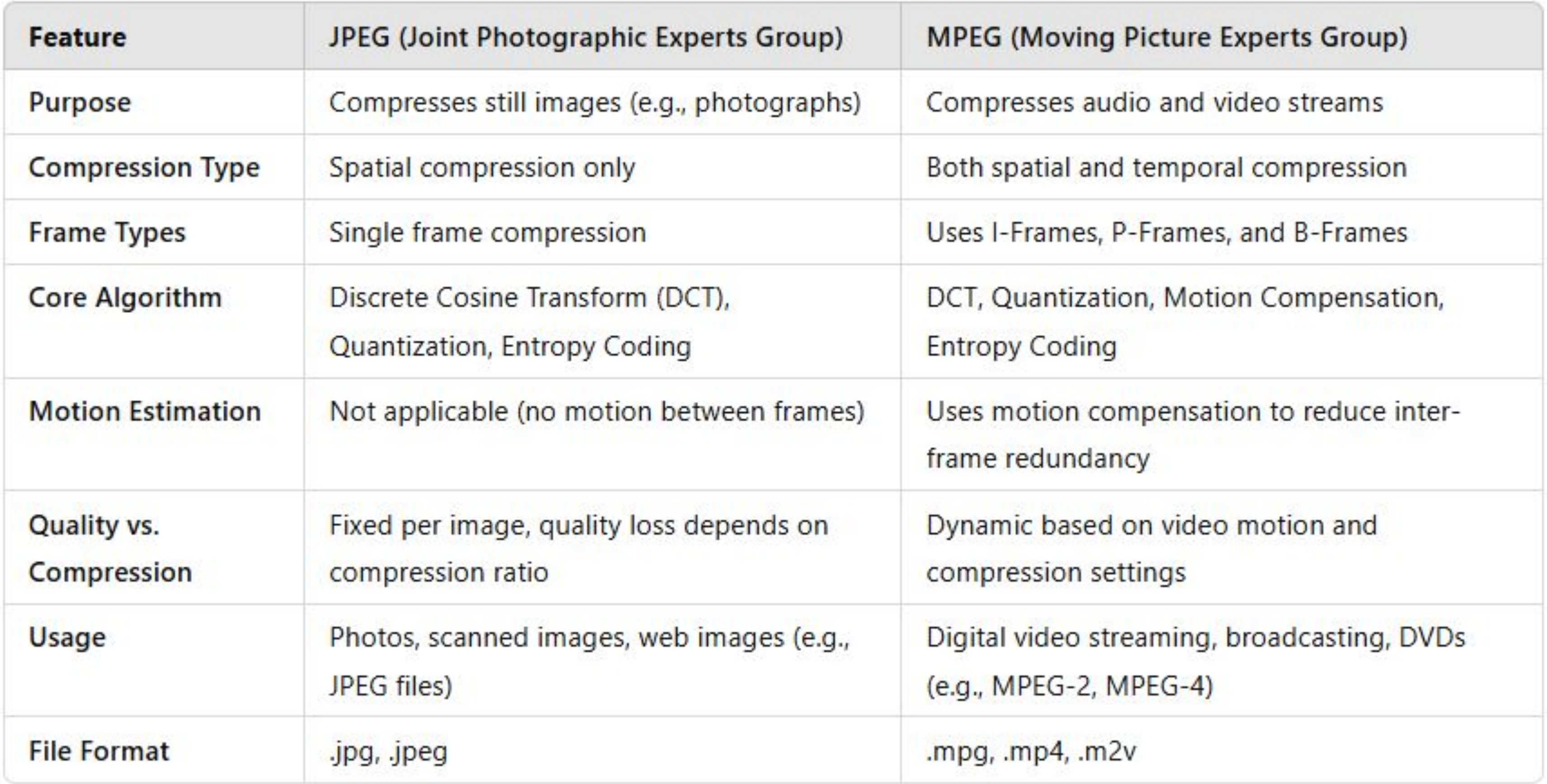
Create bit planes: The bits in the binary representation of each pixel are extracted and placed into separate planes.

For bit plane 0, all the least significant bits (LSB) of every pixel are extracted and placed into a new image (or matrix) where each pixel represents the 0th bit of the original image.

For bit plane 1, the next bit (the second least significant) of each pixel is extracted and placed into another plane, and so on until all 8 bit planes are formed.

Resulting bit planes: You will have a set of binary images, one for each bit plane. In each bit plane, the pixels will contain either a 0 or 1, depending on whether the corresponding bit in the original image was 0 or 1.

JPEG vs MPEG



Feature	JPEG (Joint Photographic Experts Group)	MPEG (Moving Picture Experts Group)
Purpose	Compresses still images (e.g., photographs)	Compresses audio and video streams
Compression Type	Spatial compression only	Both spatial and temporal compression
Frame Types	Single frame compression	Uses I-Frames, P-Frames, and B-Frames
Core Algorithm	Discrete Cosine Transform (DCT), Quantization, Entropy Coding	DCT, Quantization, Motion Compensation, Entropy Coding
Motion Estimation	Not applicable (no motion between frames)	Uses motion compensation to reduce inter-frame redundancy
Quality vs. Compression	Fixed per image, quality loss depends on compression ratio	Dynamic based on video motion and compression settings
Usage	Photos, scanned images, web images (e.g., JPEG files)	Digital video streaming, broadcasting, DVDs (e.g., MPEG-2, MPEG-4)
File Format	.jpg, .jpeg	.mpg, .mp4, .m2v

MPEG (Moving Picture Experts Group)