Lecture slides of the course

Information hiding & secret sharing

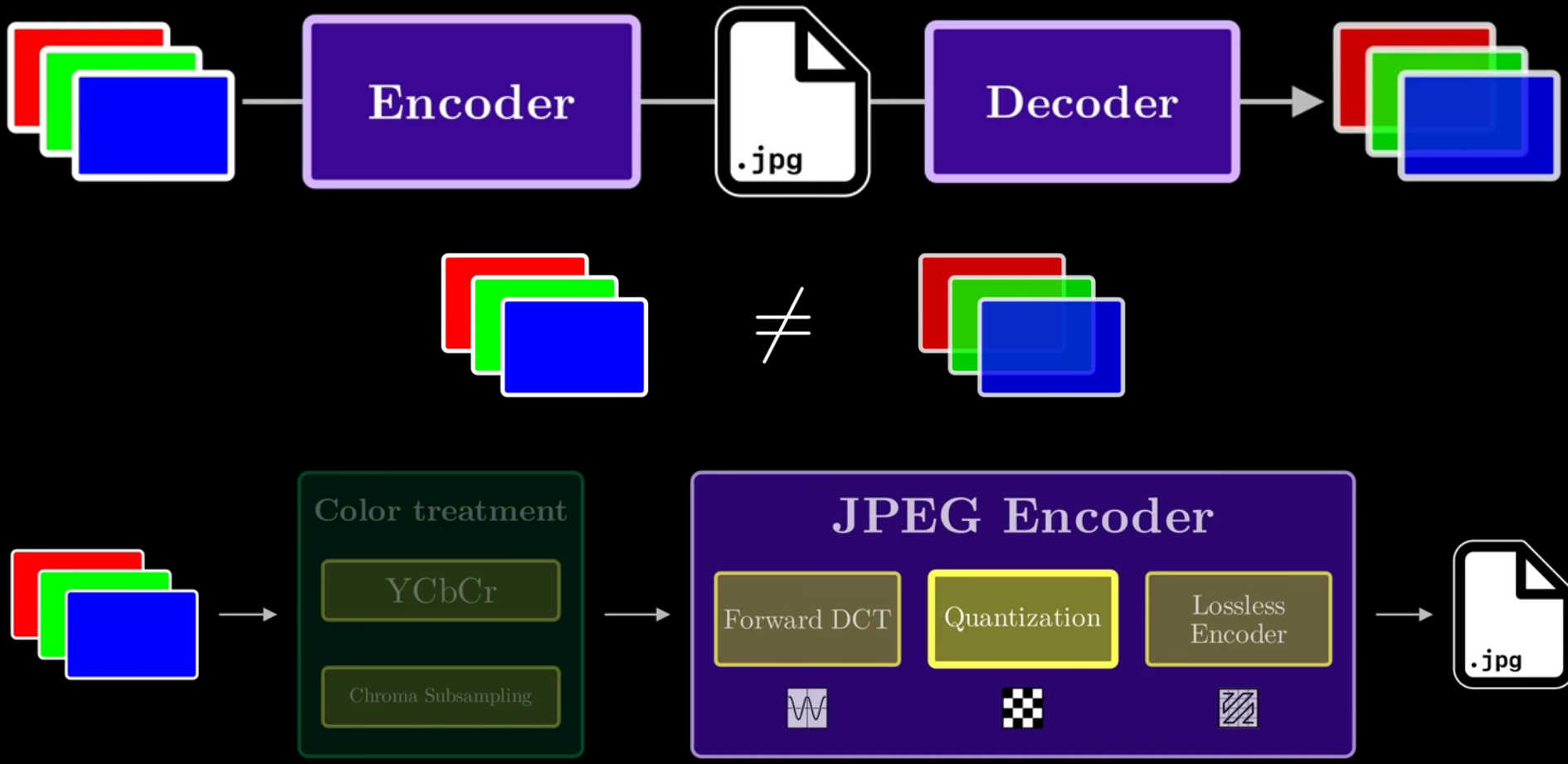
Image Steganography (P3) JPEG

Phạm Trọng Nghĩa

ptnghia@fit.hcmus.edu.vn

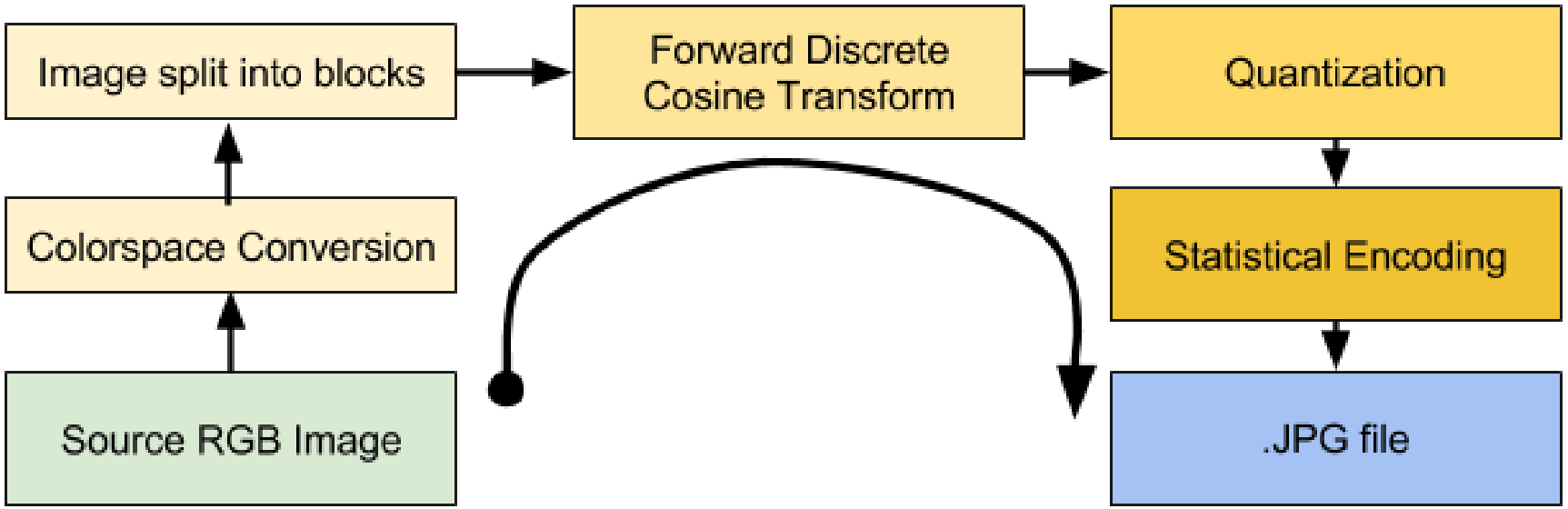
JPEG Compression workflow

- JPEG is a lossy compression



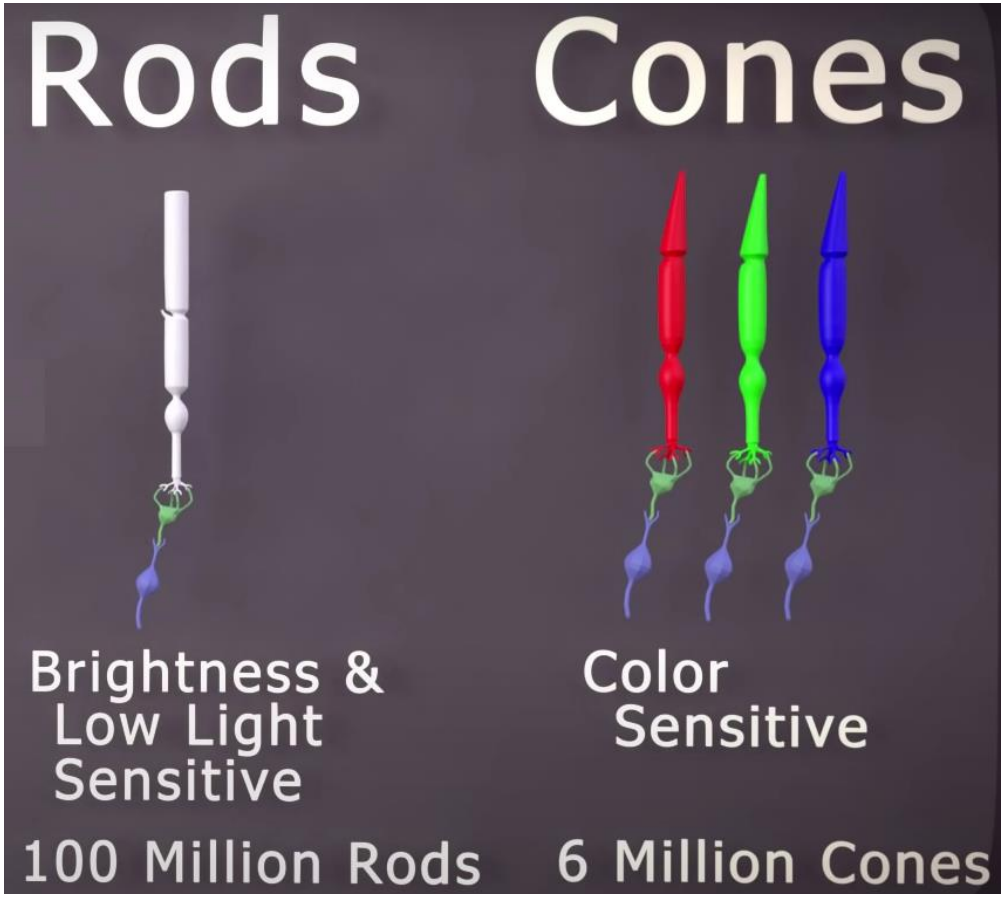
JPEG Compression workflow

- Step 1: Colorspace conversion (and downsampling)
- Step 2: Divide into block
- Step 3: Apply DCT
- Step 4: Quantization
- Step 5: Run length and Huffman encoding



Colorspace Conversion

- Convert RBG to YCbCr
- Human eye is more sensitive to **luminance** than **color**



RGB vs YCbCy color space

Y

Cb

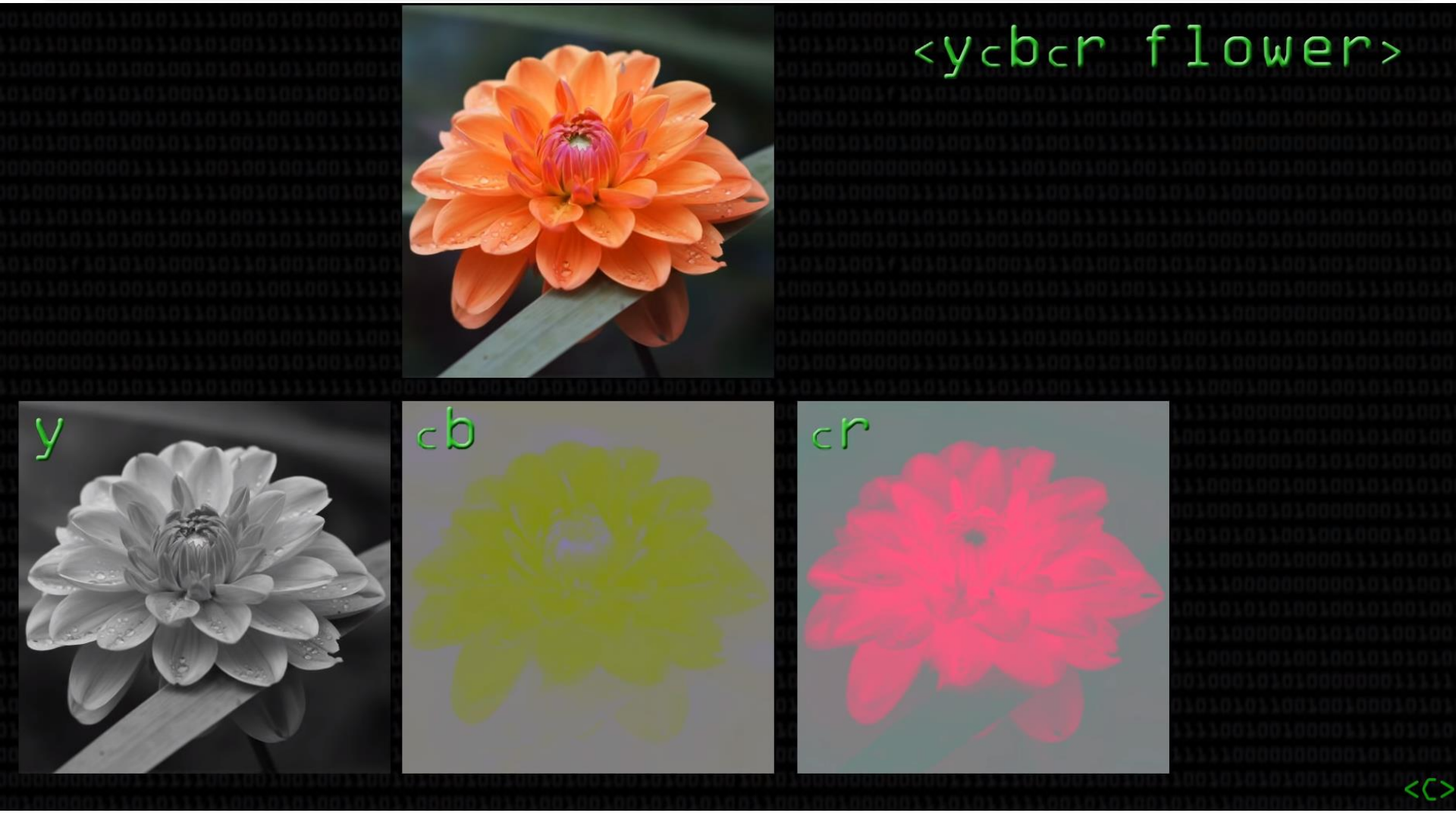
Cr

$Y = 0.299 * R + 0.587 * G + 0.114 * B$

$Cb = -0.169 * R - 0.331 * G + 0.500 * B$

$Cr = 0.500 * R - 0.419 * G - 0.081 * B$

YCbCy color space



Formular

$$\begin{pmatrix} Y \\ C_\mathrm{r} \\ C_\mathrm{b} \end{pmatrix} = \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.5 & -0.419 & -0.081 \\ -0.169 & -0.331 & 0.5 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Y

Y = 0.299 * R + 0.587 * G + 0.114 * B

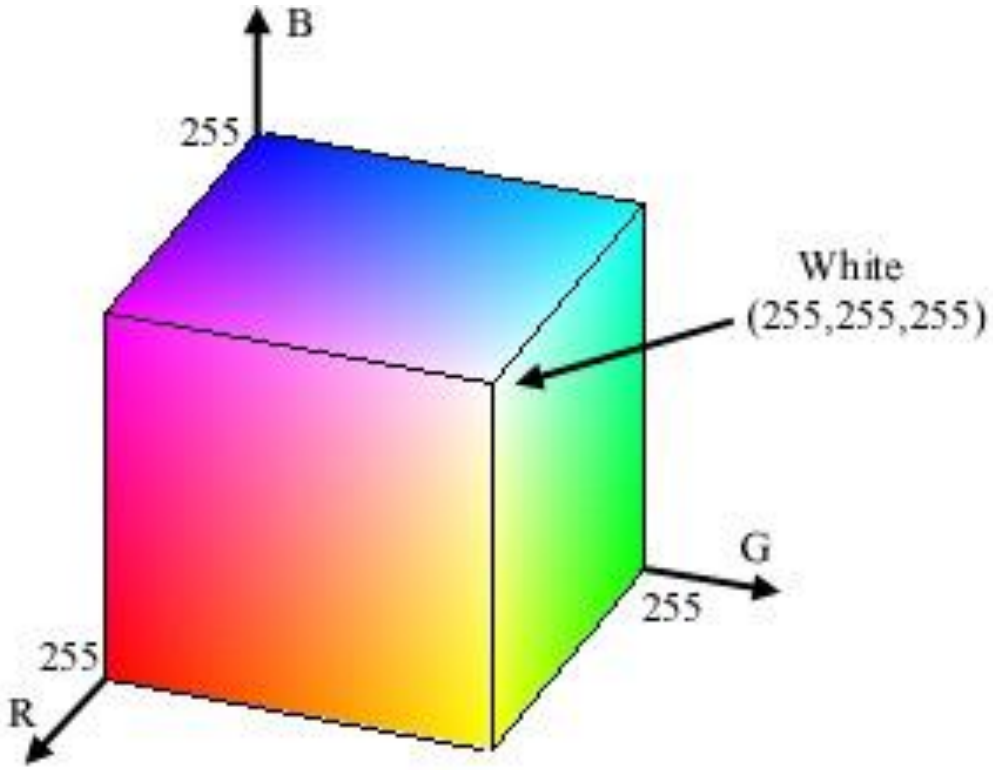
Cb

Cb = -0.169 * R - 0.331 * G + 0.500 * B

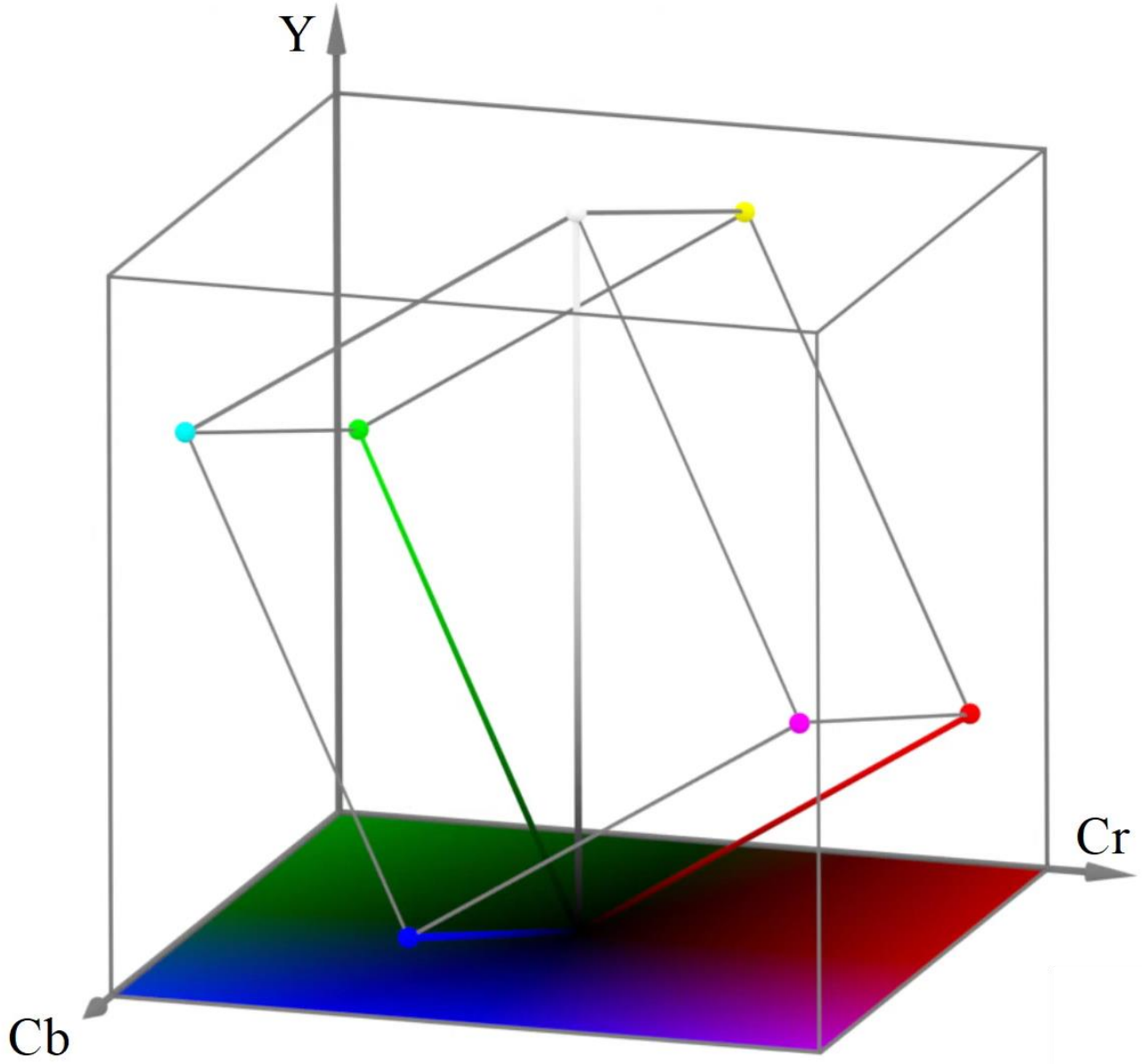
Cr

Cr = 0.500 * R - 0.419 * G - 0.081 * B

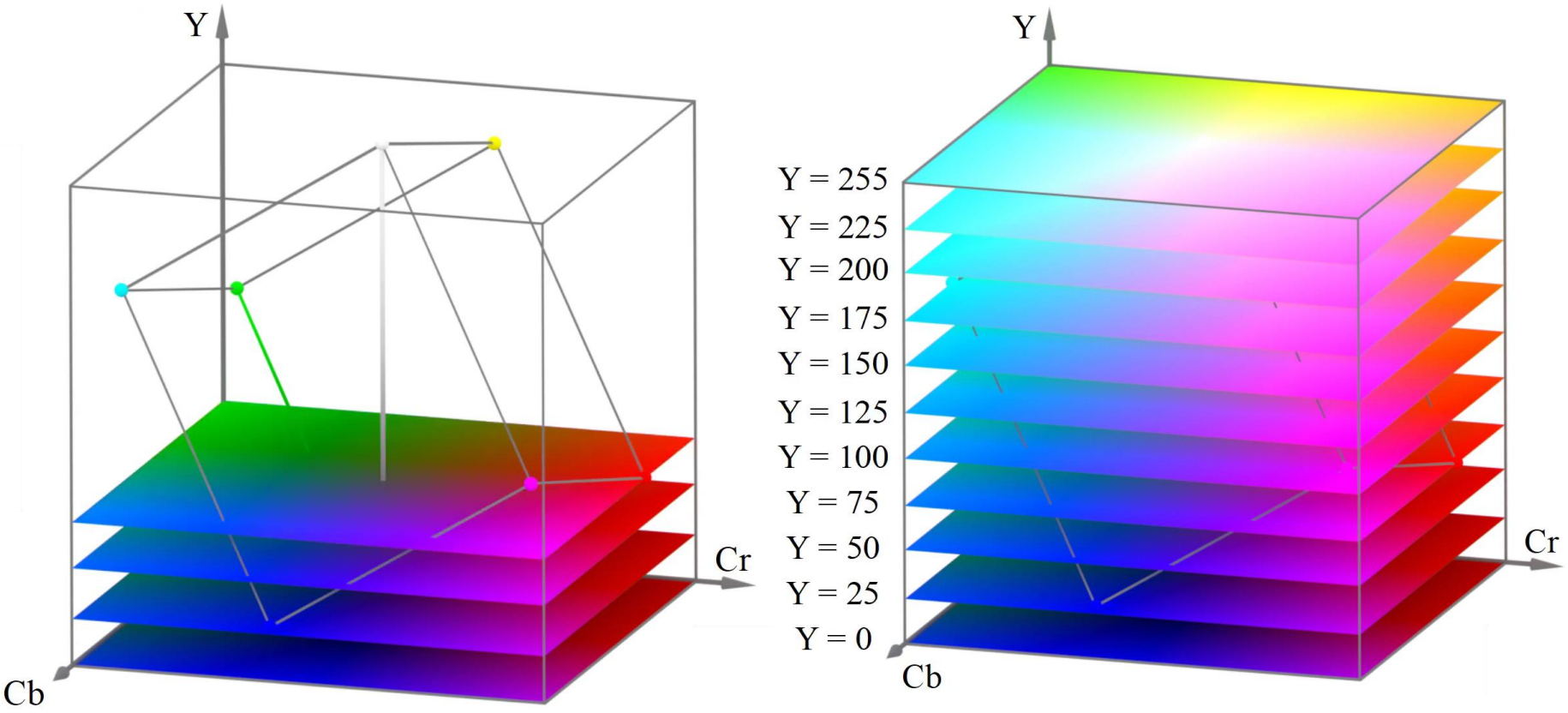
RGB Cube



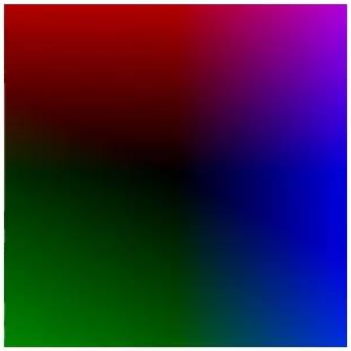
YCbCr cube

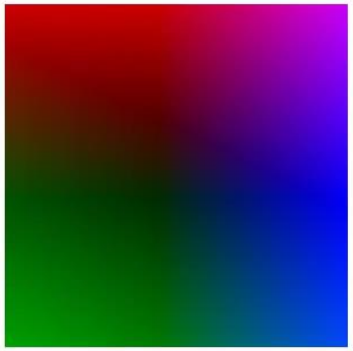


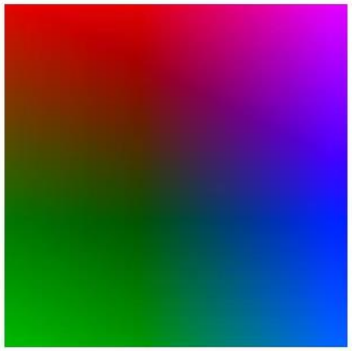
YCbCr cube

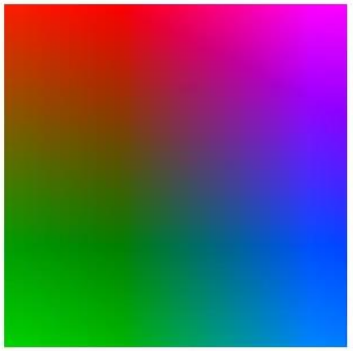


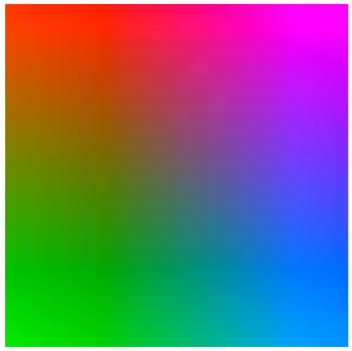
YCbCr cube

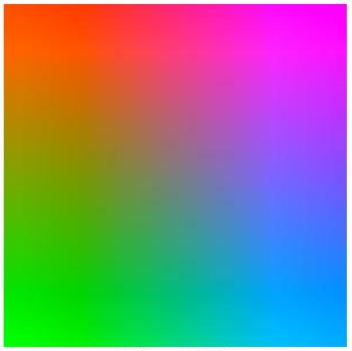

Y = 0

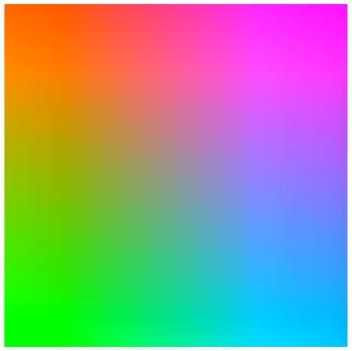

Y = 25

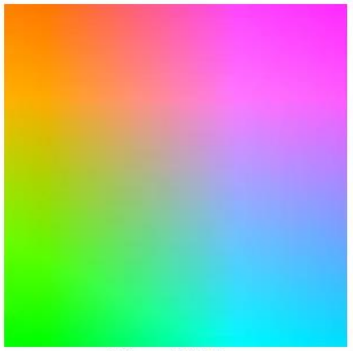

Y = 50

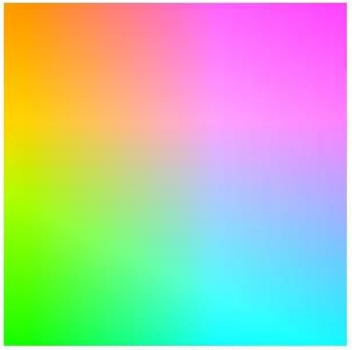

Y = 75

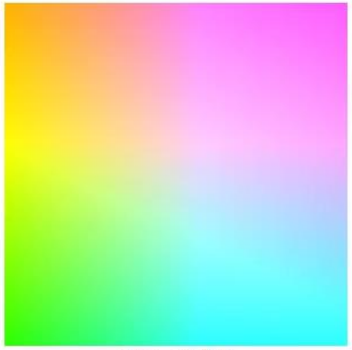

Y = 100

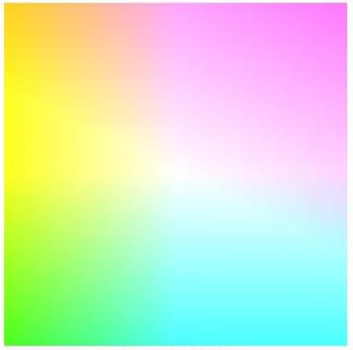

Y = 125


Y = 150


Y = 175

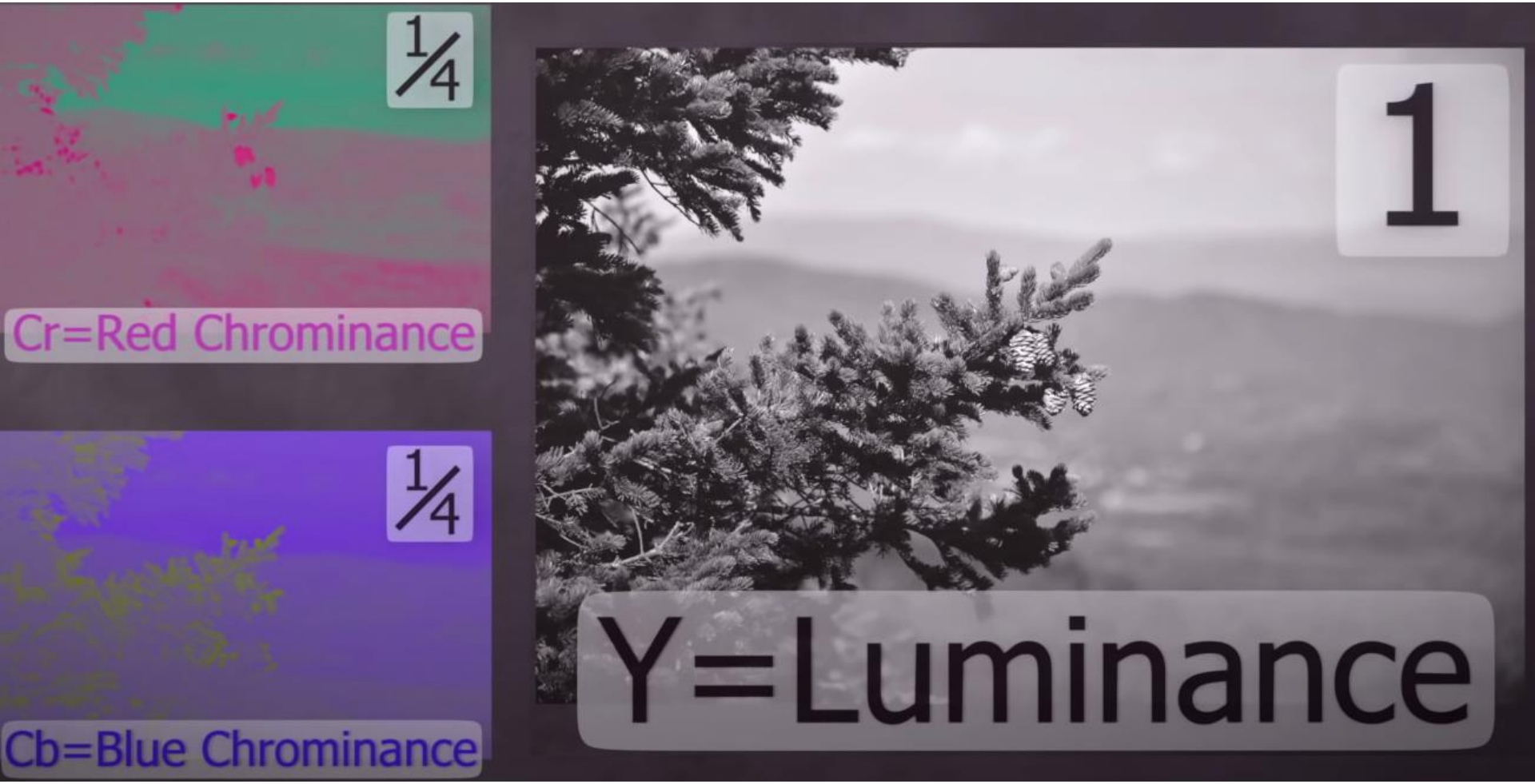

Y = 200


Y = 225

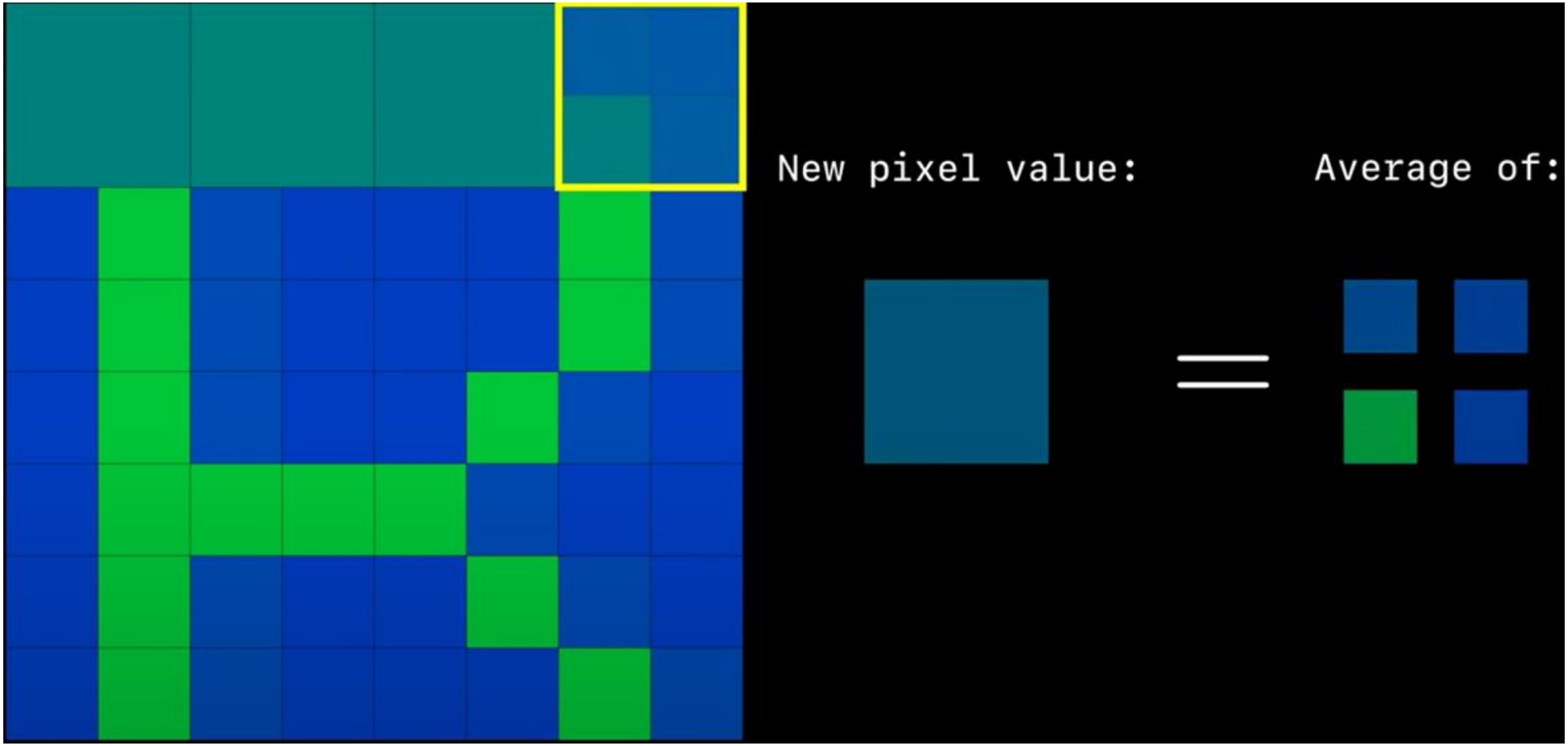

Y = 255

Chroma Downsampling (4:2:0)

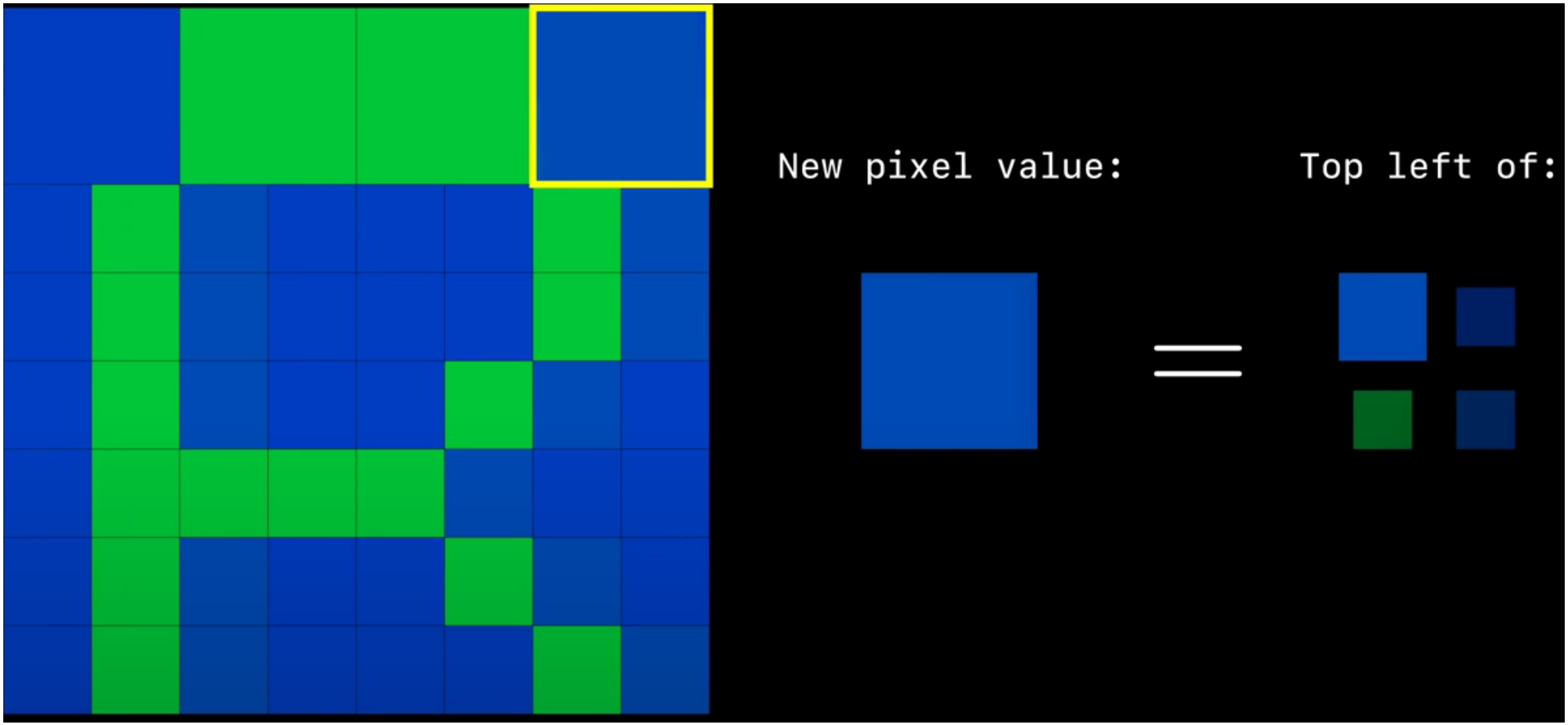
- [First step in compression image]



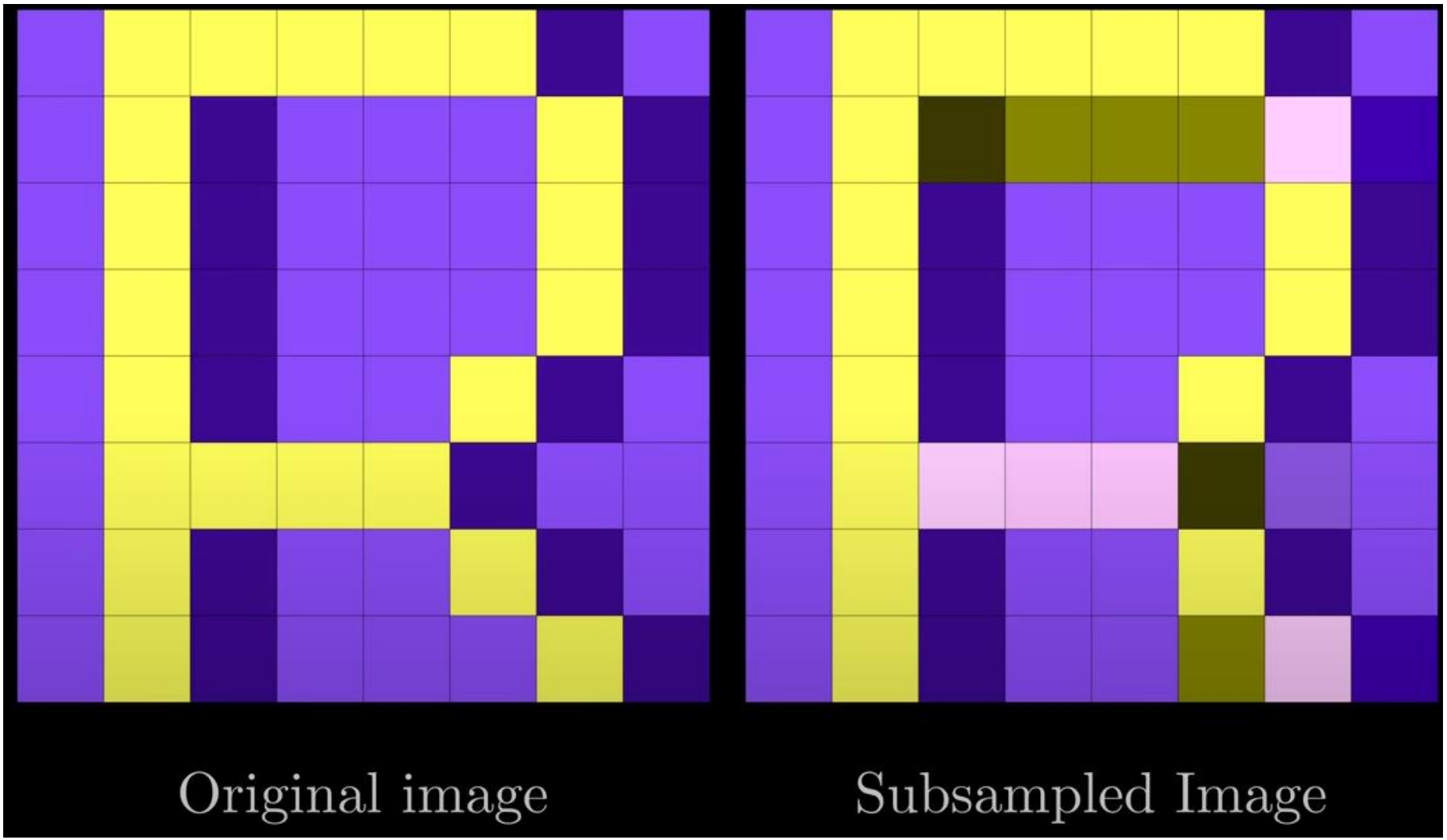
Chroma Downsampling (4:2:0)



Chroma Subsampling (4:2:0)



Chroma Subsampling (4:2:0)



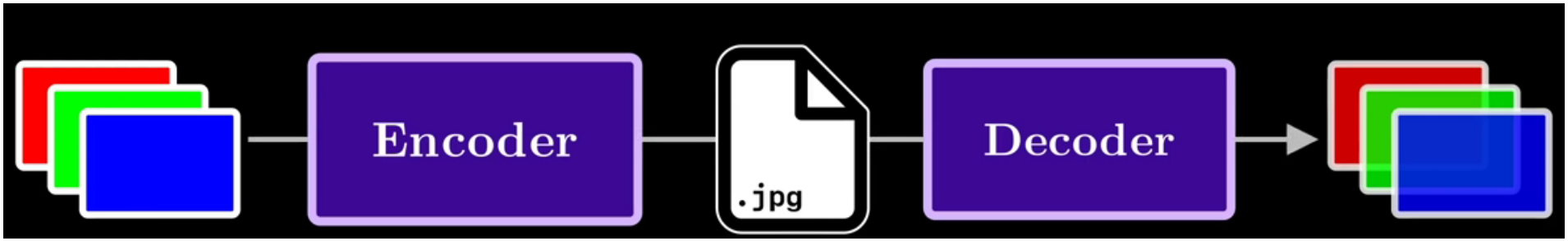
Chroma Subsampling (4:2:0)

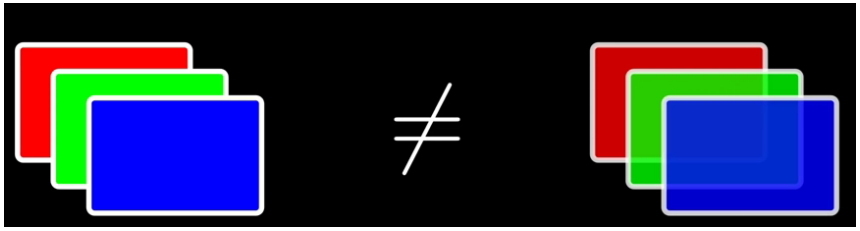
Original image

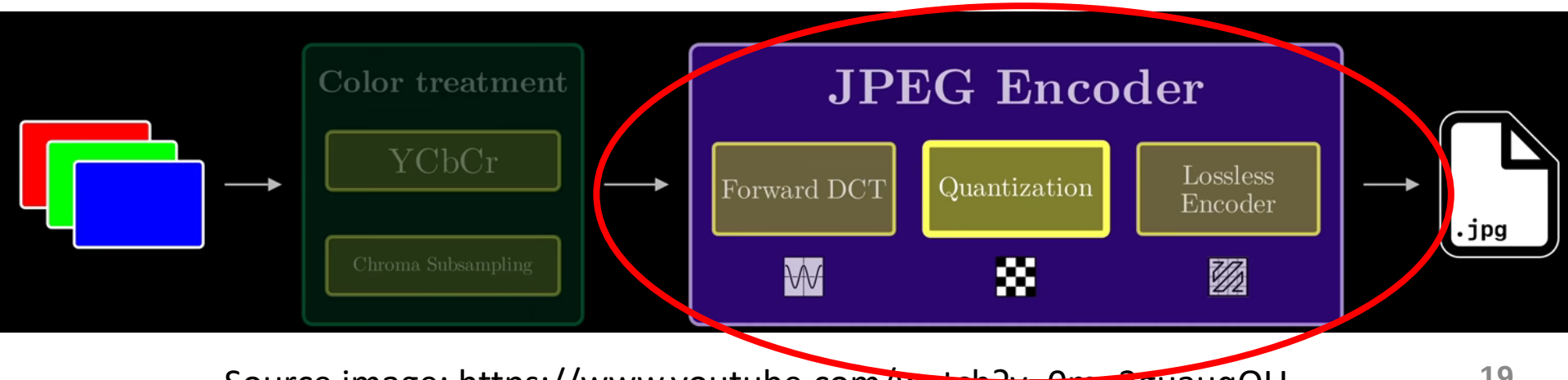
Subsampling 4:2:0

JPEG Compression workflow

- JPEG is a lossy compression

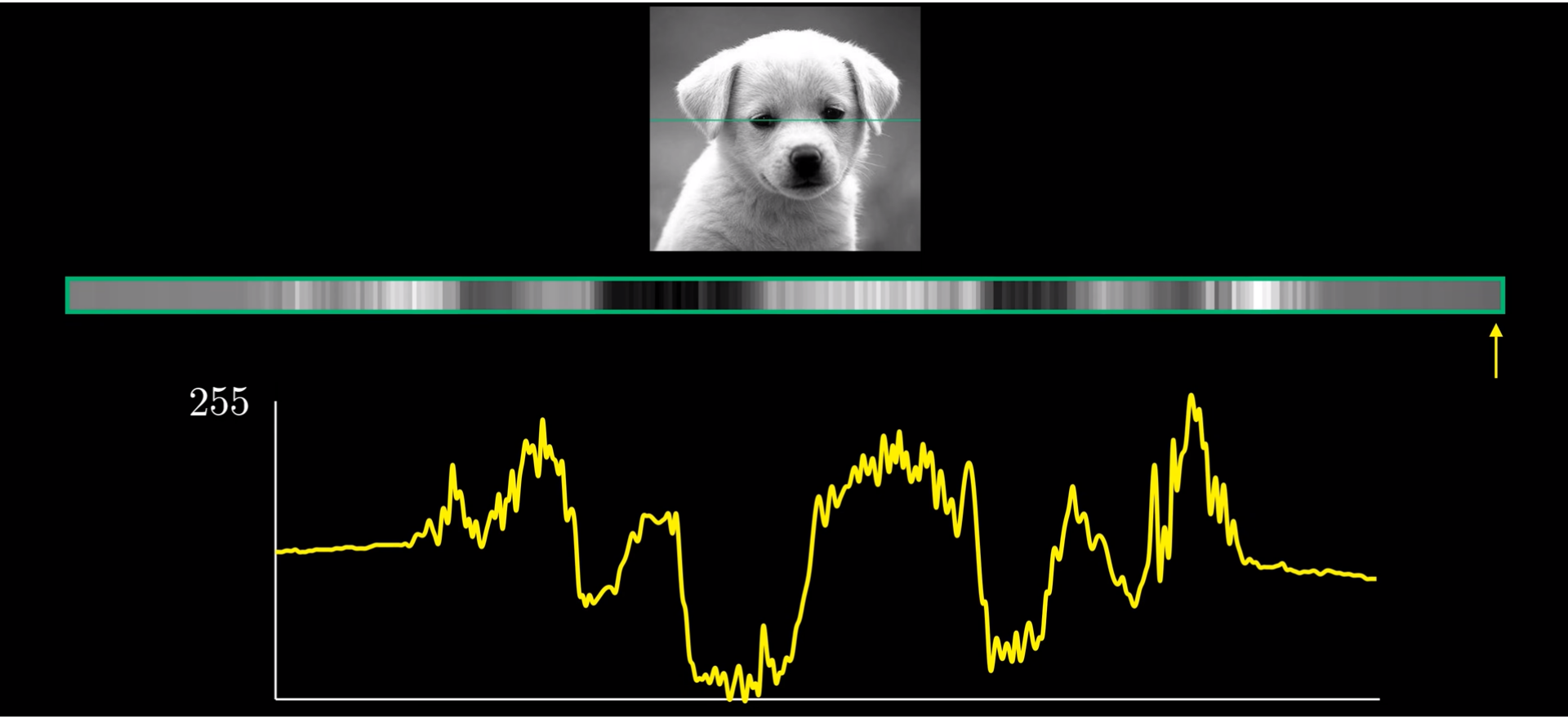




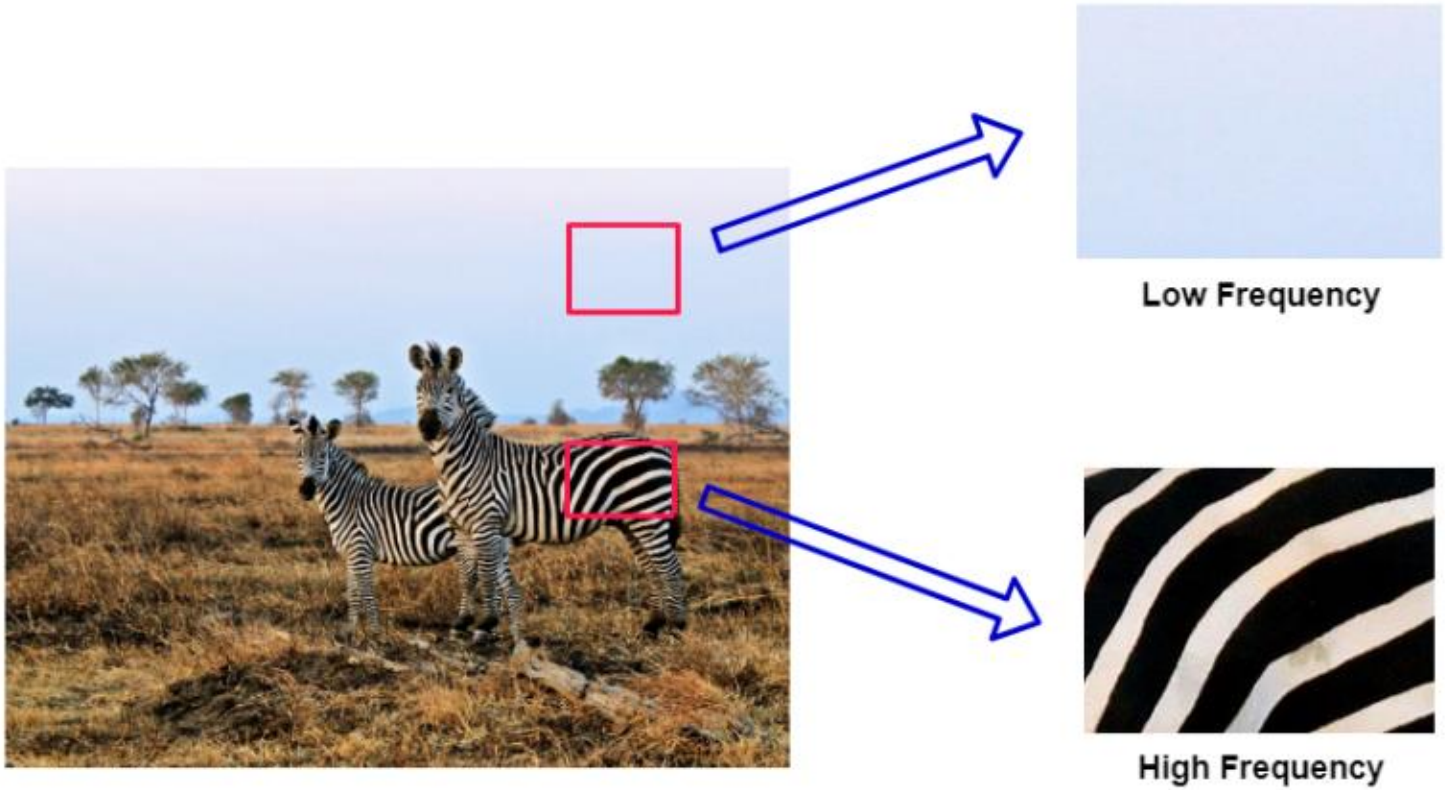


Source image: https://www.youtube.com/watch?v=0me3guauqOU

Image as a wave

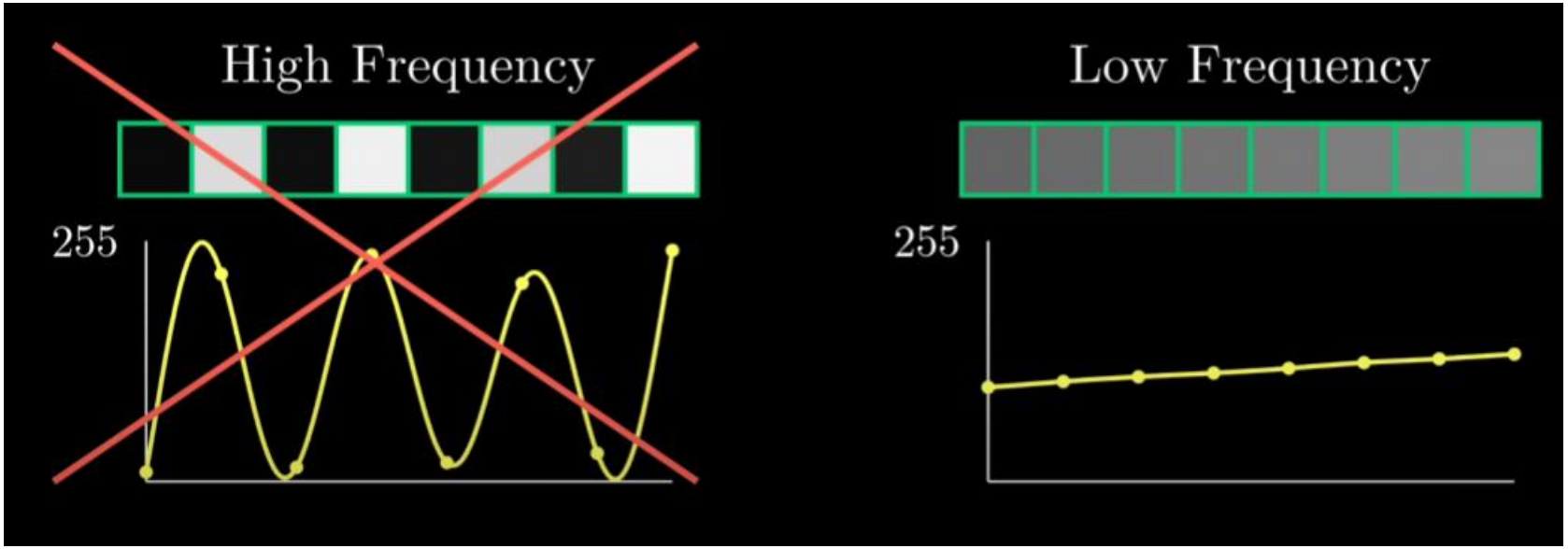


High frequency vs low frequency

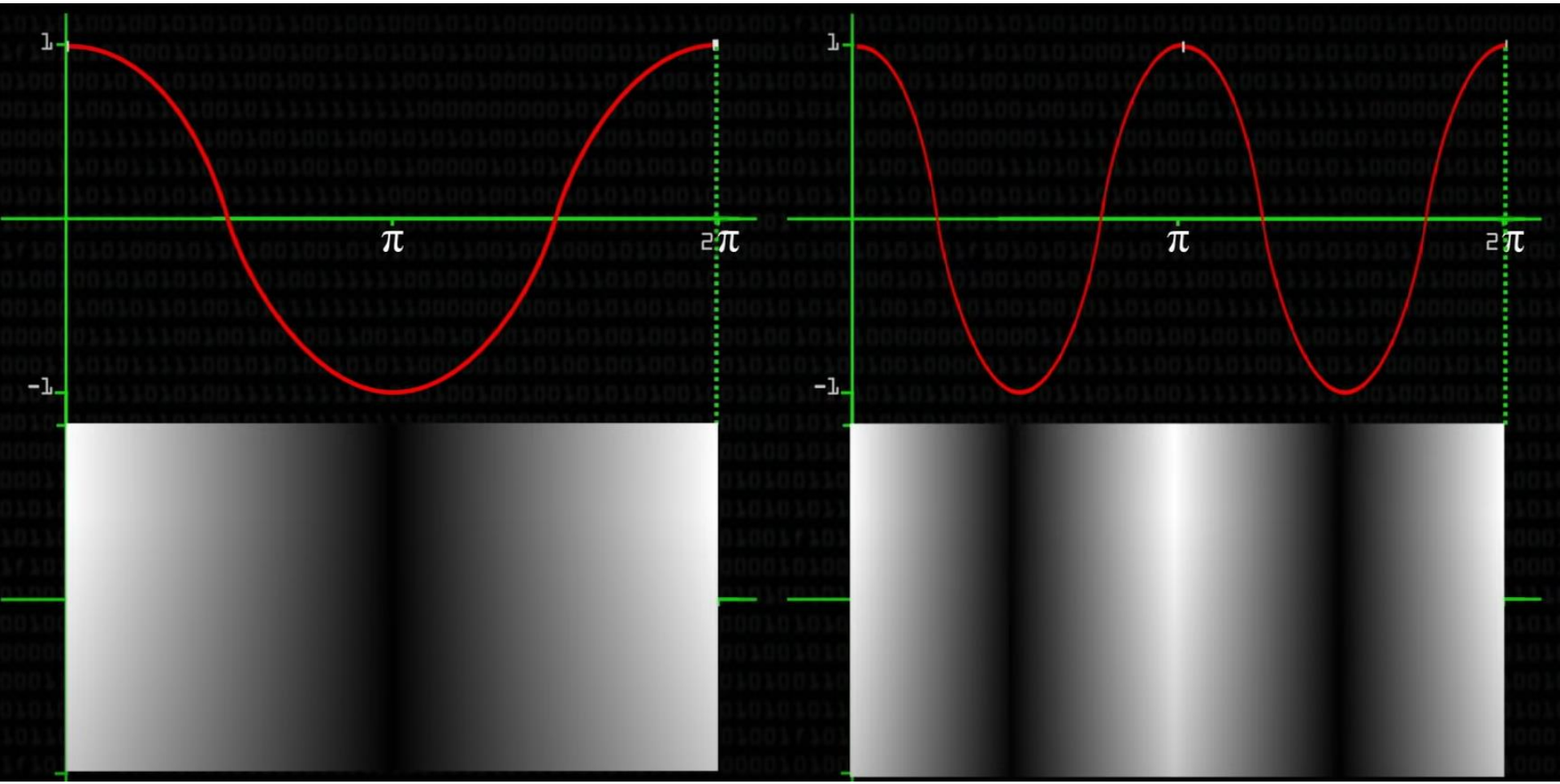


High frequency vs low frequency

- *Real world image* tend to have **more low frequency** component
- Experiments suggest that humans are **more immune to loss of higher spatial frequency components than loss of lower frequency components**

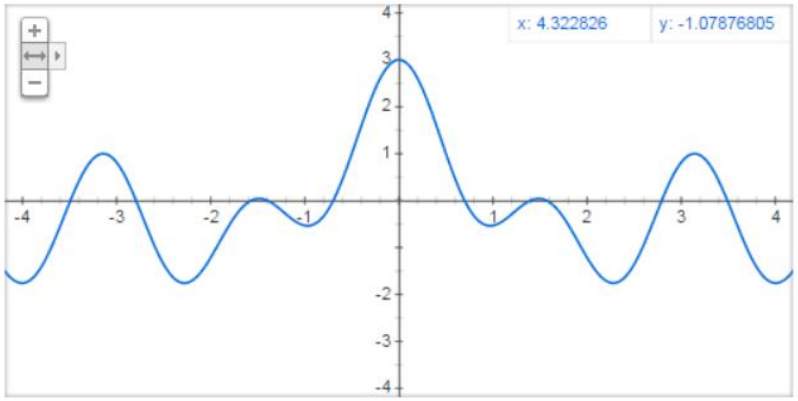


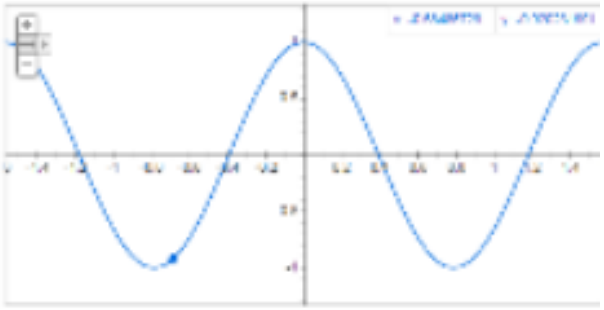
Discrete Cosine Transform



Discrete Cosine Transform

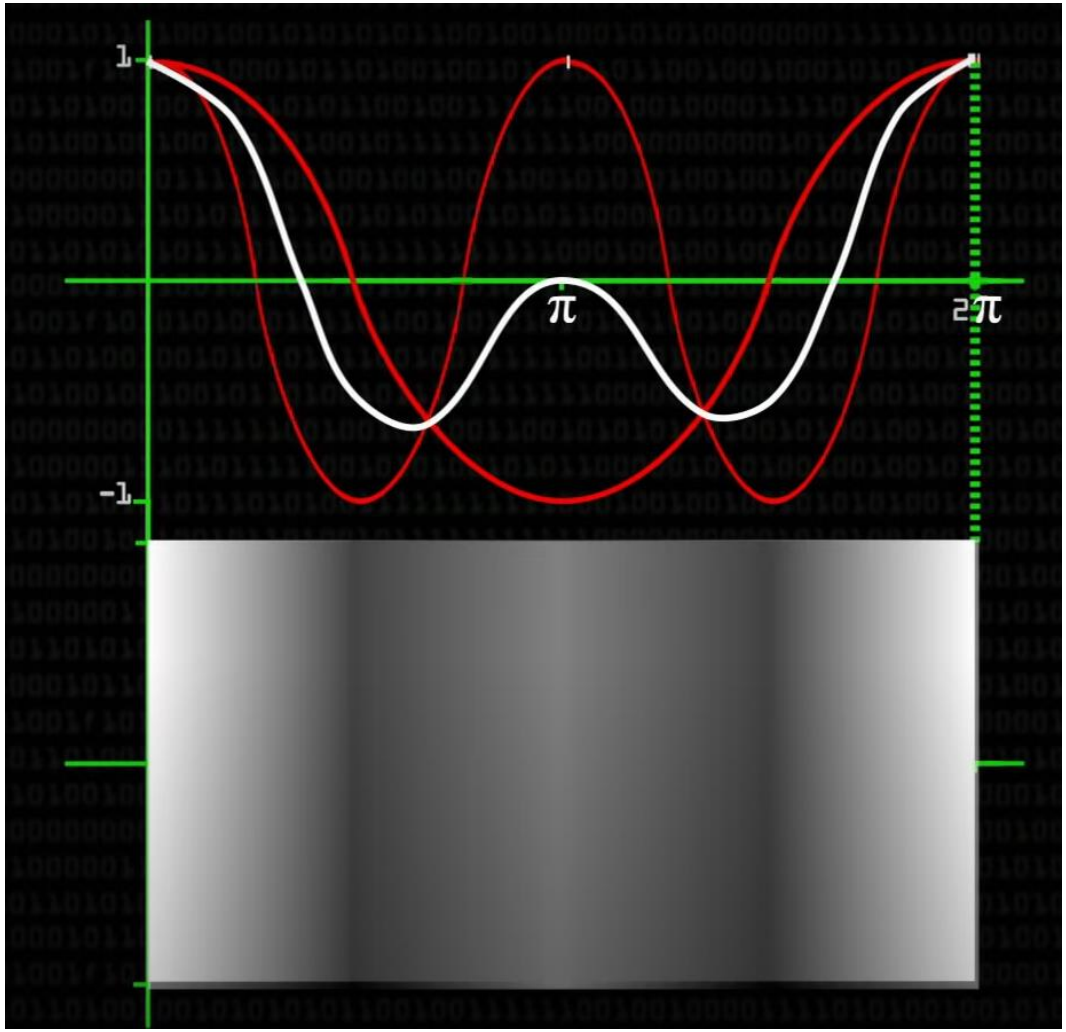
- Any signal can be represent by a linear combination of cosine wave.



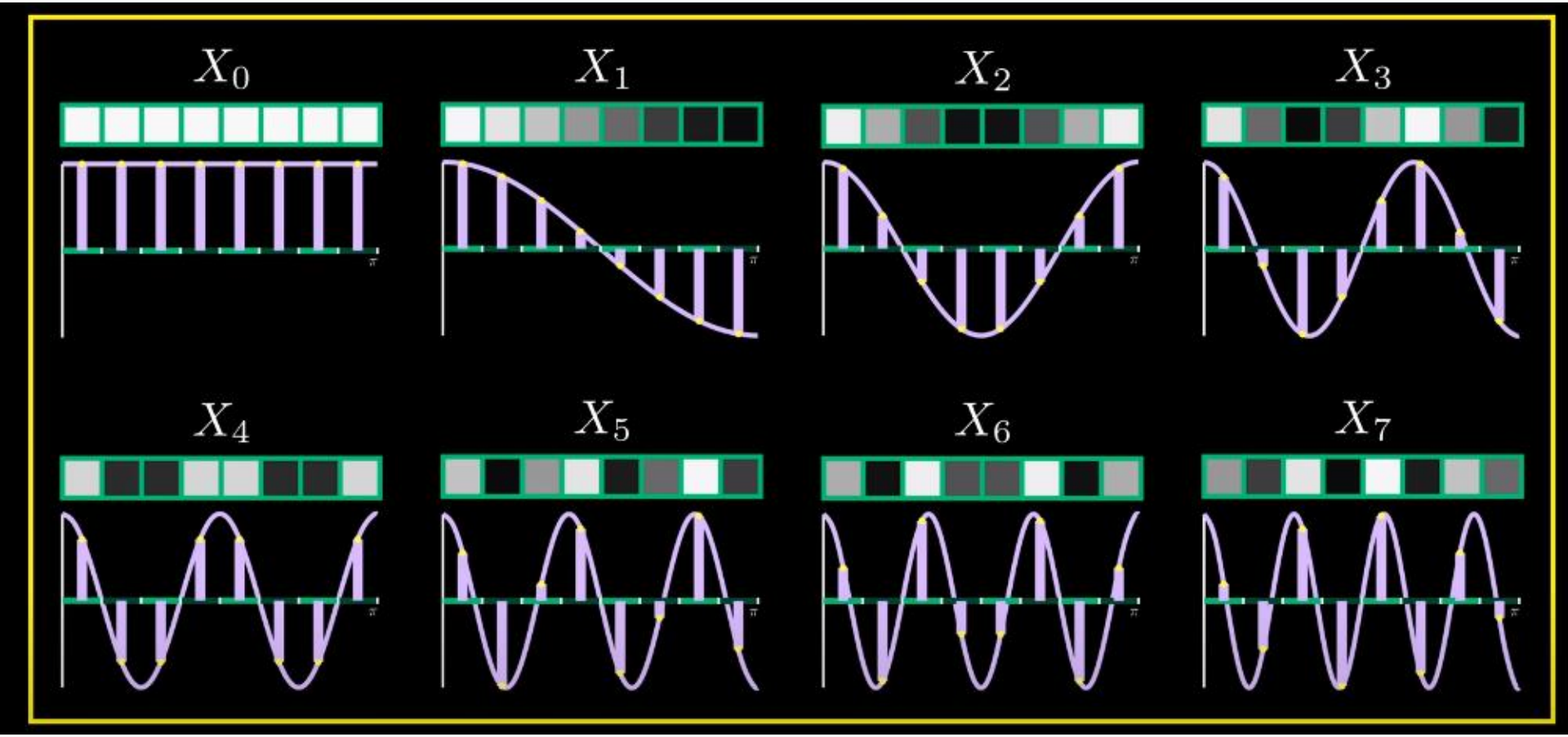


Discrete Cosine Transform

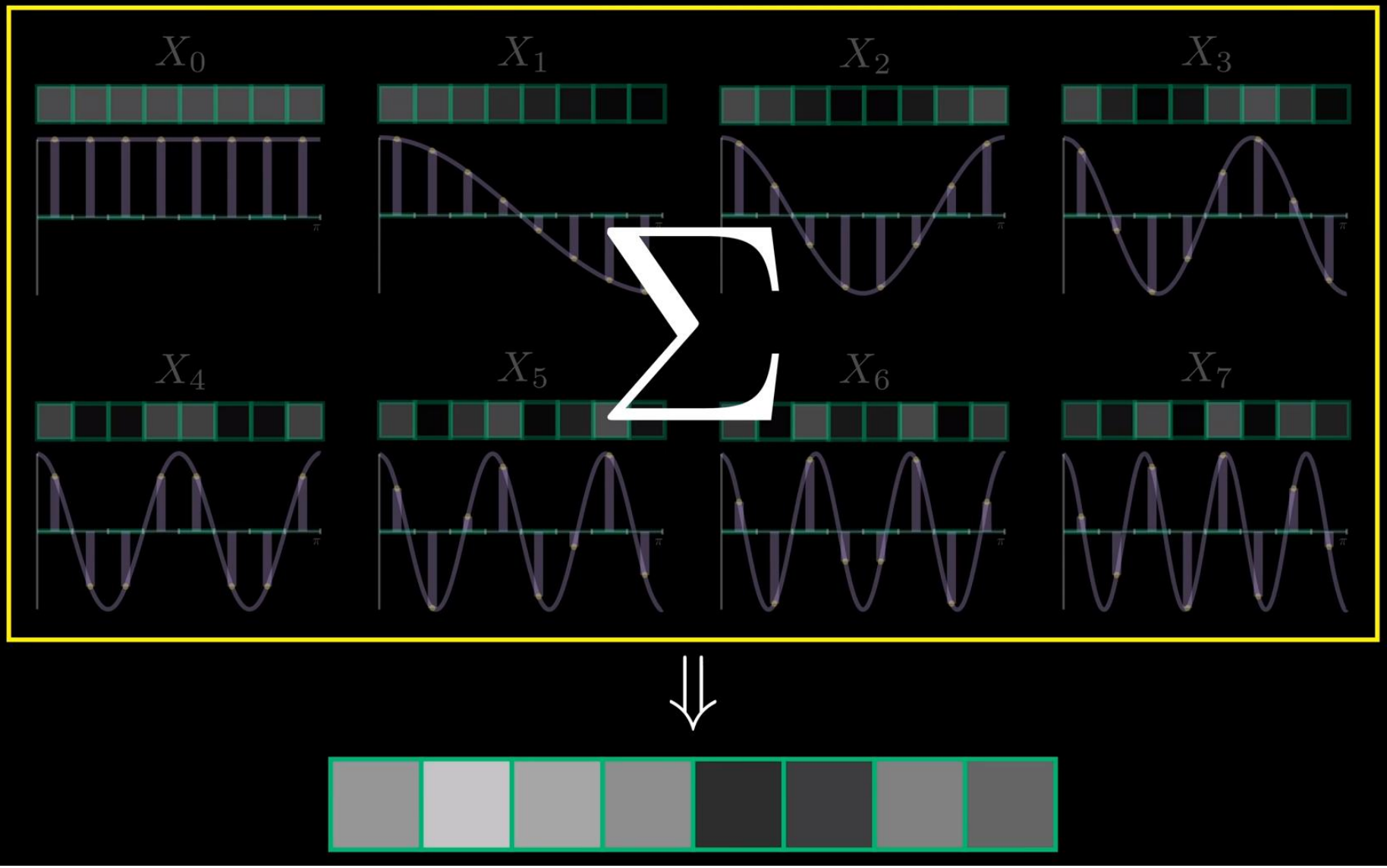
- Weighted linear combination of cosine wave.
- Mathemethically, if we have signal that 8 long, we can represented it with **8 cosine wave of different frequency**



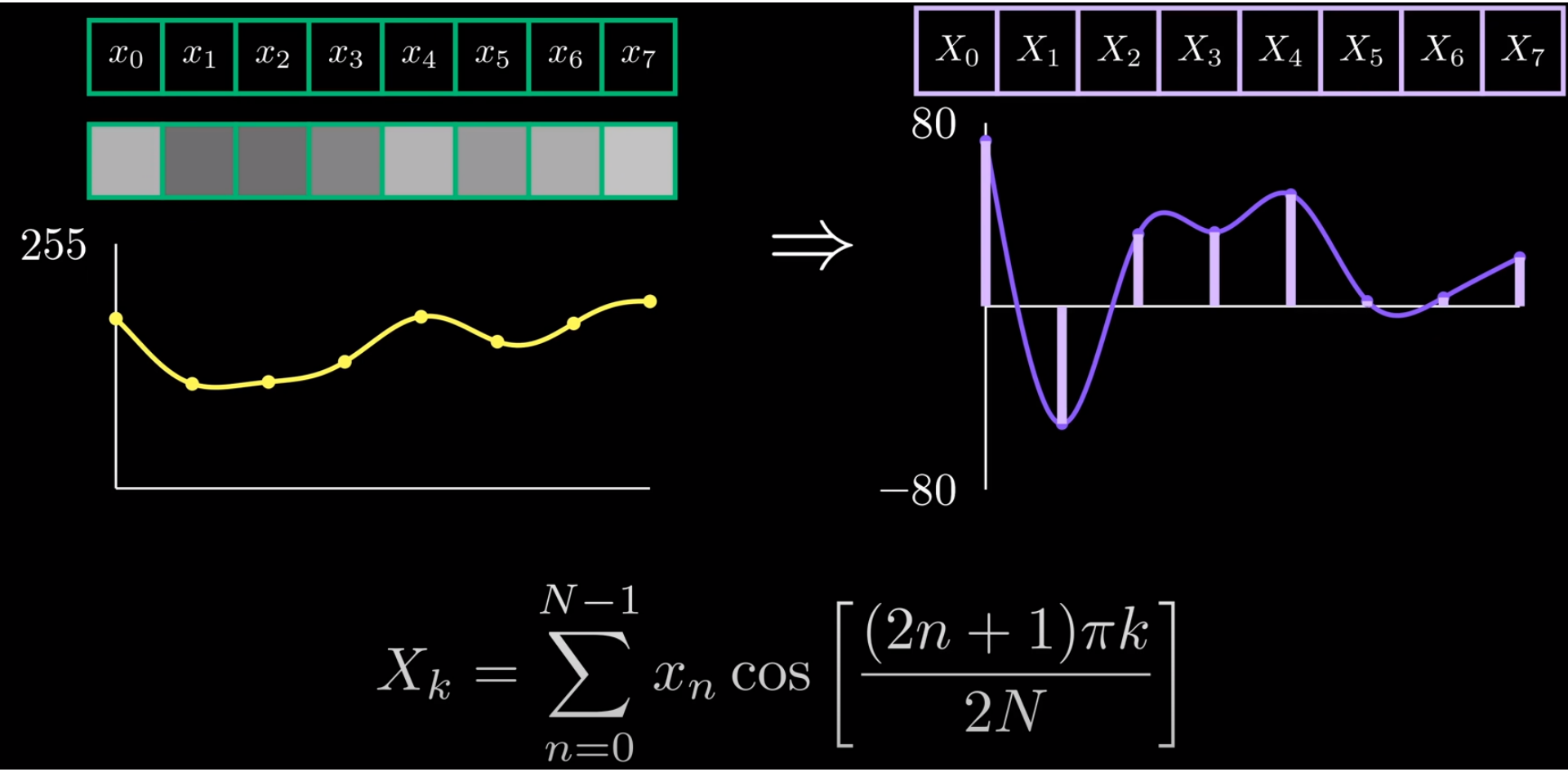
Discrete Cosine Transform



Discrete Cosine Transform

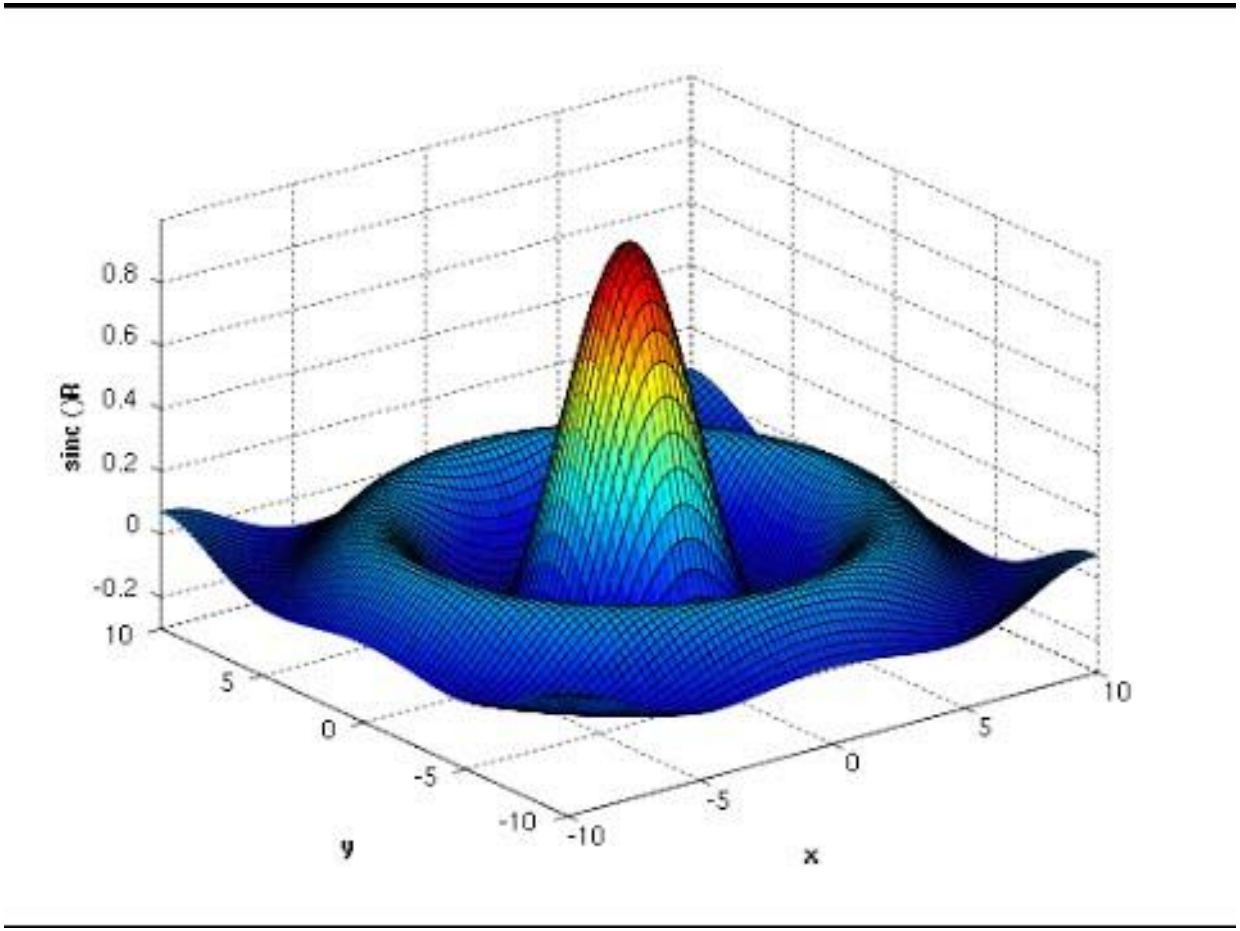


Discrete Cosine Transform

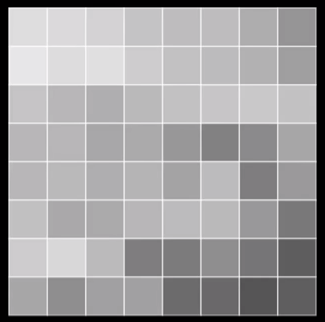


$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{(2n+1)\pi k}{2N}\right]$$

Signal in 3D



Discrete Cosine Transform

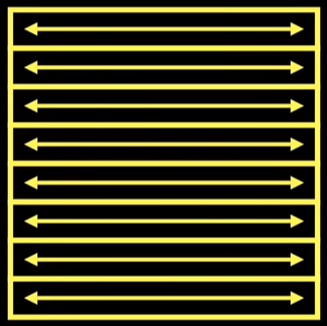


93	90	83	68	61	61	46	21
102	92	95	77	65	60	49	32
69	55	47	57	65	70	72	65
55	55	40	42	23	1	11	38
55	57	47	53	35	59	-2	26
64	41	42	55	60	57	25	-8
77	87	58	-2	-5	14	-10	-35
38	14	33	33	-21	-23	-43	-34

93	90	83	68	61	61	46	21
102	92	95	77	65	60	49	32
69	55	47	57	65	70	72	65
55	55	40	42	23	1	11	38
55	57	47	53	35	59	-2	26
64	41	42	55	60	57	25	-8
77	87	58	-2	-5	14	-10	-35
38	14	33	33	-21	-23	-43	-34

$8 \times$ DCT 1D

Rows

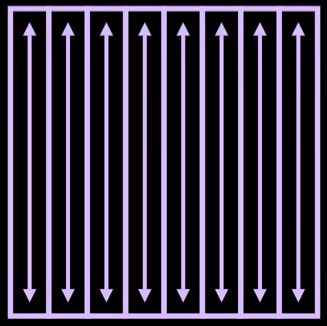


184	60	-8	12	-13	3	0	0
202	63	-6	4	-7	6	4	3
176	-12	7	16	4	3	-2	0
93	39	17	-21	18	-5	-6	-3
116	37	-12	7	2	-14	22	-27
118	37	-33	37	2	8	3	-1
65	107	23	14	-40	-10	7	0
-1	79	-11	-18	12	19	16	-12

184	60	-8	12	-13	3	0	0
202	63	-6	4	-7	6	4	3
176	-12	7	16	4	3	-2	0
93	39	17	-21	18	-5	-6	-3
116	37	-12	7	2	-14	22	-27
118	37	-33	37	2	8	3	-1
65	107	23	14	-40	-10	7	0
-1	79	-11	-18	12	19	16	-12

$8 \times$ DCT 1D

Columns



338	145	-8	18	-7	4	16	-14
162	-41	3	2	3	-1	-13	9
-17	57	-2	-2	-20	16	2	10
41	19	-24	31	-19	-8	4	-1
-59	7	-2	-32	21	-1	6	-15
-19	12	32	0	-16	-9	-15	12
7	-55	-24	17	20	15	-4	0
15	-11	10	11	-18	-13	10	-10

Any 8x8 image can be created by linearly combining the 64 basic images below

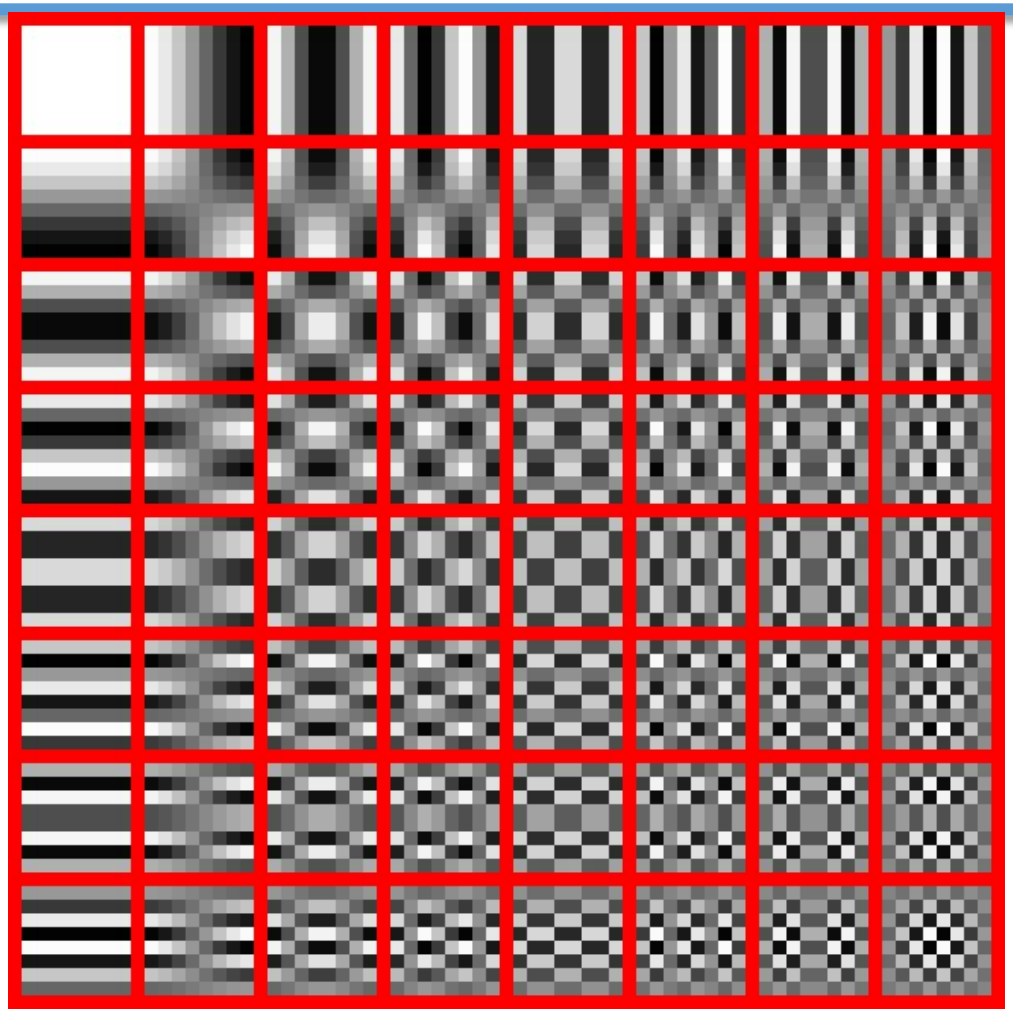
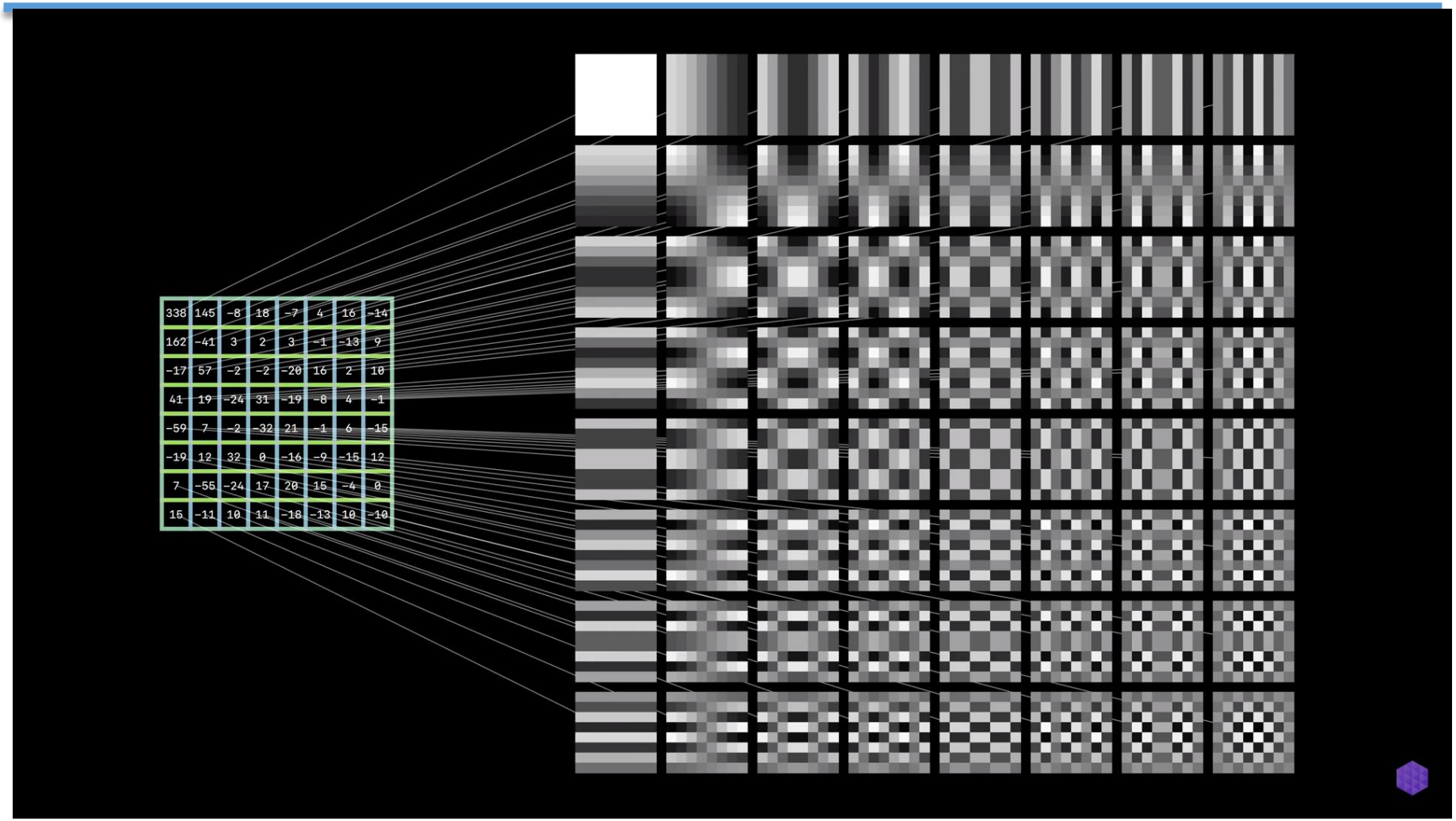


Image source: https://en.wikipedia.org/wiki/JPEG

- Each 8x8 base image can be viewed as a 64-D vector, which is normalized (length 1) and orthogonal to the vectors corresponding to other base images.
- These 64 basic images form a new coordinate system in 64D space

Any 8x8 image can be created by linearly combining the 64 basic images below



- Each 8x8 base image can be viewed as a 64-D vector, which is normalized (length 1) and orthogonal to the vectors corresponding to other base images.
- These 64 basic images form a new coordinate system in 64D space

Discrete Cosine Transform

- 2D Discrete Cosine Transform (DCT)
- For an 8 × 8 block.
  - $B[i,j], with\ i,j\ \epsilon\ [0,7]$ - pixel value
  - $d[k,l], with\ k,l\ \epsilon\ [0,7]$– DCT coefficient

$$\mathbf{d}[k,l] = \sum_{i,j=0}^{7} \mathbf{f}[i,j;k,l]\mathbf{B}[i,j]$$

$$= \sum_{i,j=0}^{7} \frac{\mathbf{w}[k]\mathbf{w}[l]}{4} \cos\frac{\pi}{16}k(2i+1) \cos\frac{\pi}{16}l(2j+1)\mathbf{B}[i,j],$$

where, $w[0] = \frac{1}{\sqrt{2}}, w[k>0] = 1$

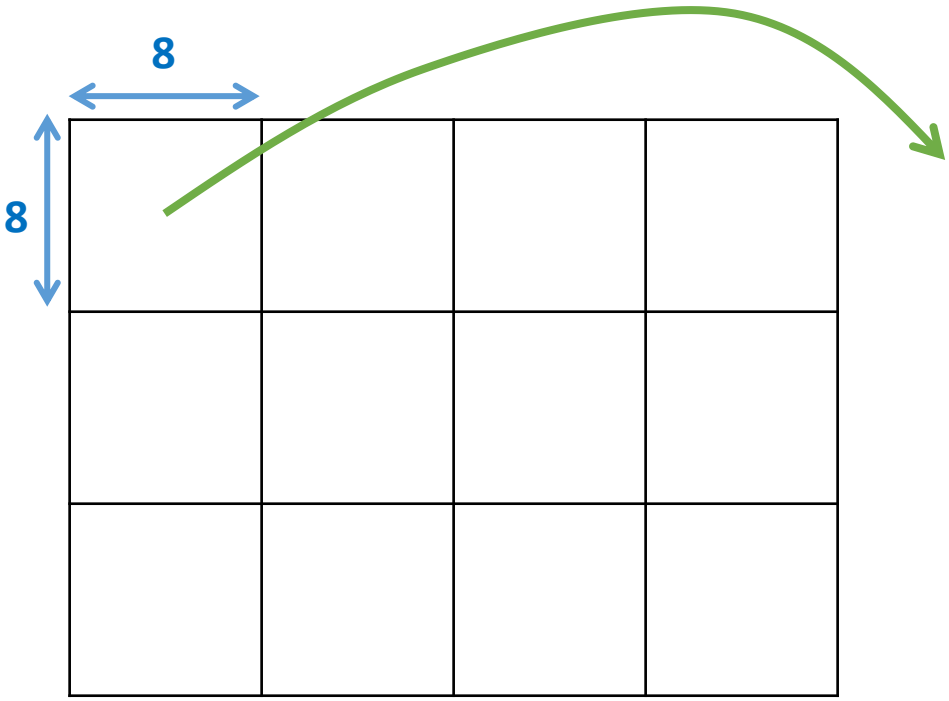
Invert Discrete Cosine Transform

- 2D Discrete Cosine Transform (DCT)
- For an 8 × 8 block.
  - $B[i,j], with\ i,j\ \epsilon\ [0,7]$ - pixel value
  - $d[k,l], with\ k,l\ \epsilon\ [0,7]$– DCT coefficient

$$\mathbf{B}[i,j] = \sum_{k,l=0}^{7} \frac{\mathbf{w}[k]\mathbf{w}[l]}{4} \cos\frac{\pi}{16}k(2i+1) \cos\frac{\pi}{16}l(2j+1)\mathbf{d}[k,l]$$

where, $w[0] = \frac{1}{\sqrt{2}}, w[k>0] = 1$

Divide into block 8×8 (grayscale image or one color channel)



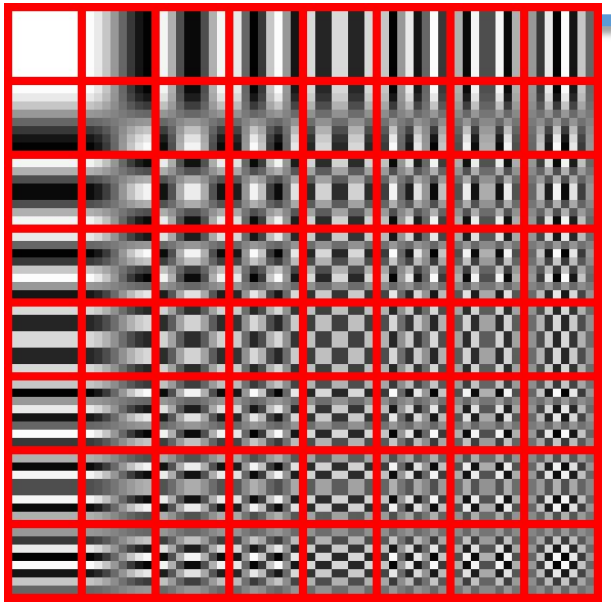
$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Apply DCT (Discrete Cosine Transform)

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \xrightarrow{-128} \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

$$\xrightarrow{\text{DCT}} \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

Nén JPEG – B2: biến đổi DCT



−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−73	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−60	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	50	34

DCT

How is the reverse?

−415.38	−30.19	−61.20	27.24	56.12	−20.10	−2.39	0.46
4.47	−21.86	−60.76	10.25	13.15	−7.09	−8.54	4.88
−46.83	7.37	77.13	−24.56	−28.91	9.93	5.42	−5.65
−48.53	12.07	34.10	−14.76	−10.24	6.30	1.83	1.95
12.12	−6.55	−13.20	−3.95	−1.87	1.75	−2.79	3.14
−7.73	2.91	2.38	−5.94	−2.38	0.94	4.30	1.85
−1.03	0.18	0.42	−2.42	−0.88	−3.02	4.12	−0.66
−0.17	0.14	−1.07	−4.19	−1.17	−0.10	0.50	1.68

These are the coefficients of the base images, called the DCT . coefficients

How to calculate DCT coefficient?

What is the value domain of the DCT coefficient?

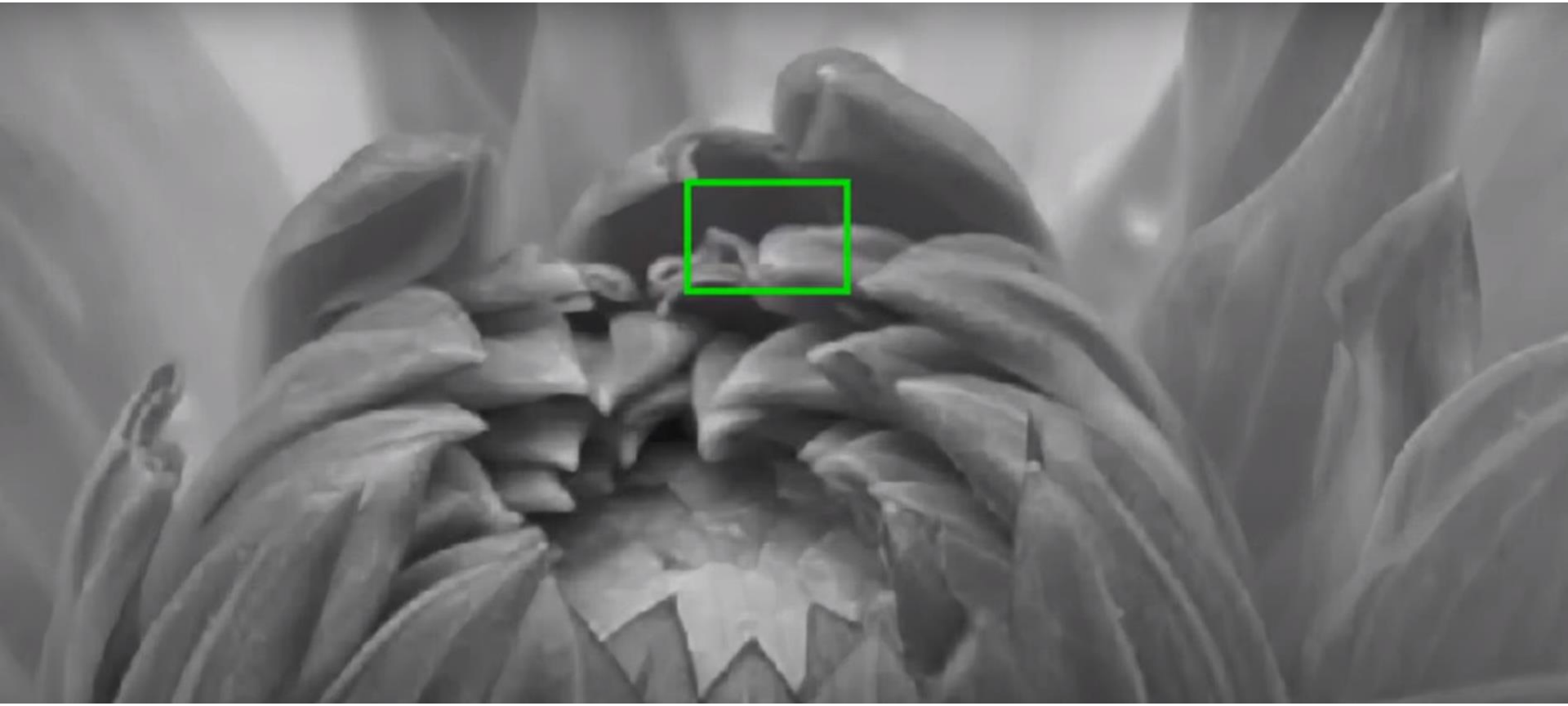
Quantization (lossy)

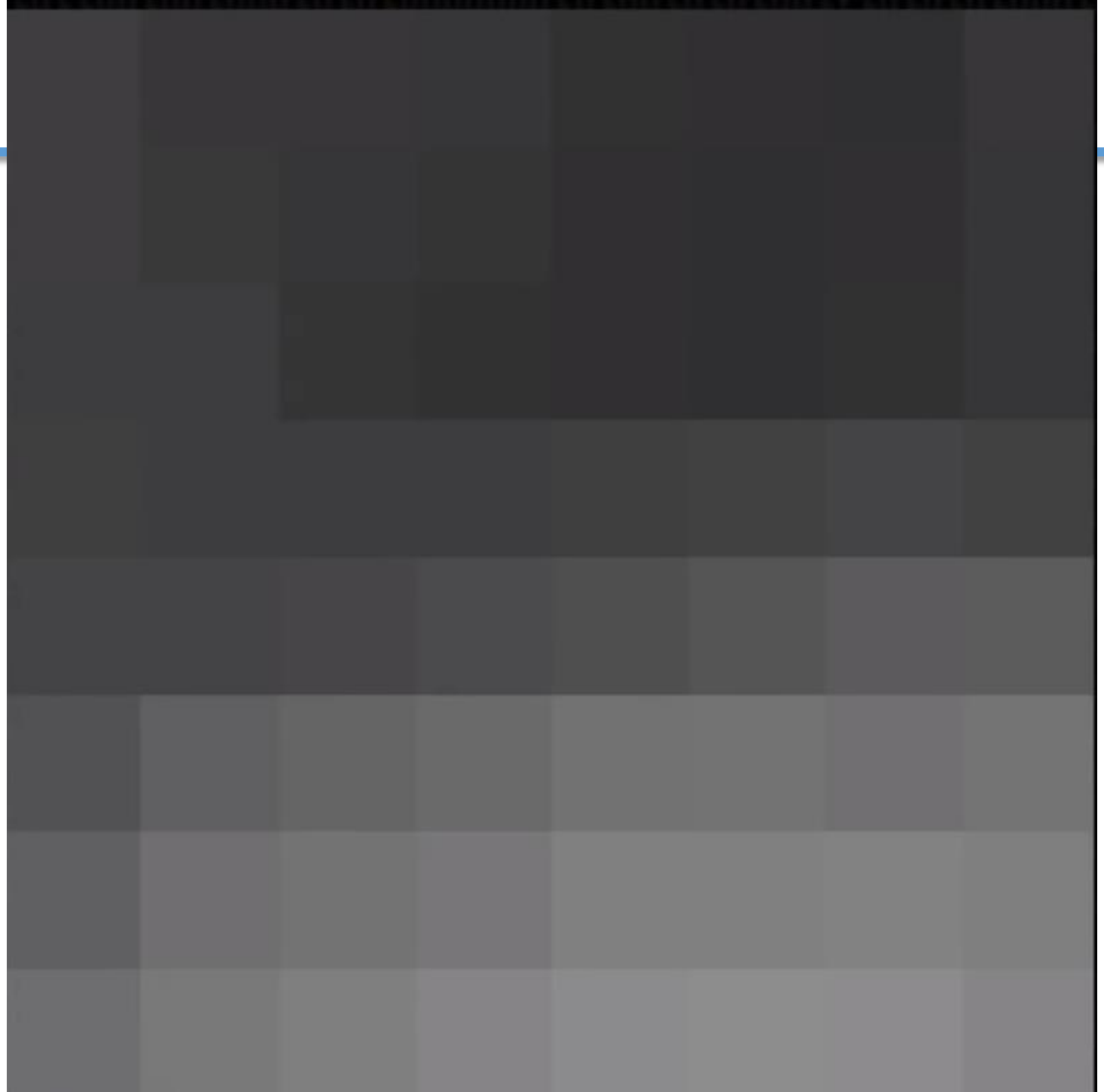
$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}.$$

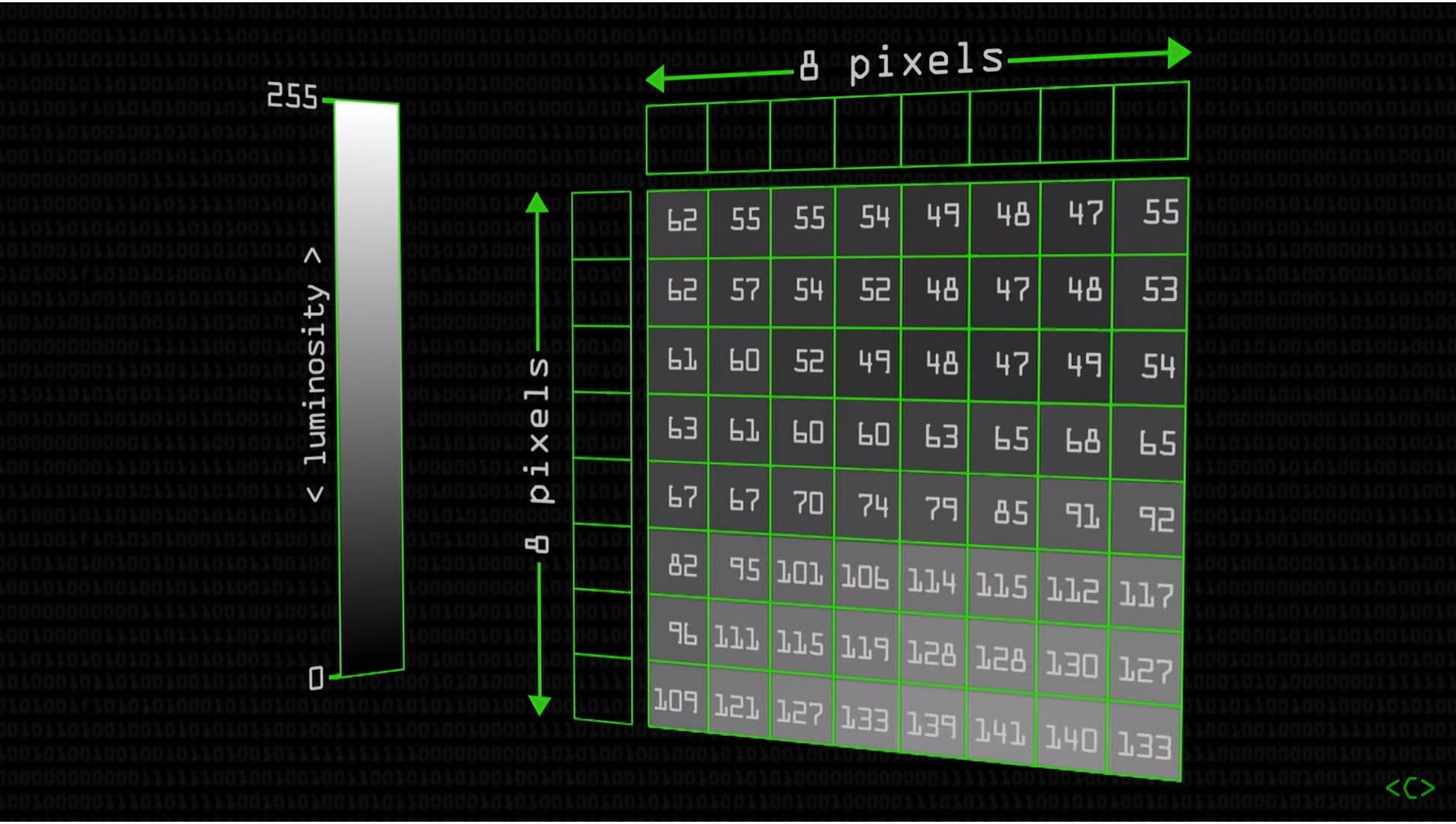
Divide by the corresponding element in Q and round to the nearest integer

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

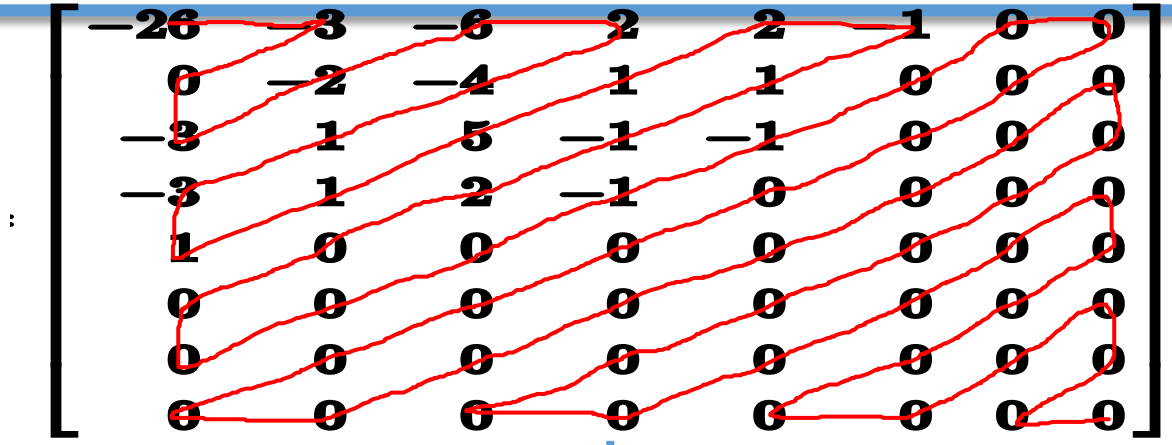
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$






8 pixels
255
< luminosity >
0
8 pixels
62 55 55 54 49 48 47 55
62 57 54 52 48 47 48 53
61 60 52 49 48 47 49 54
63 61 60 60 63 65 68 65
67 67 70 74 79 85 91 92
82 95 101 106 114 115 112 117
96 111 115 119 128 128 130 127
109 121 127 133 139 141 140 133

Lossless compression



Zig-zac scan to 1D array

−26, −3, 0, −3, −2, −6, 2, −4, 1, −3, 1, 1, 5, 1, 2, −1, 1, −1, 2, 0, 0, 0, 0, 0, −1, −1, 0, 0, ...

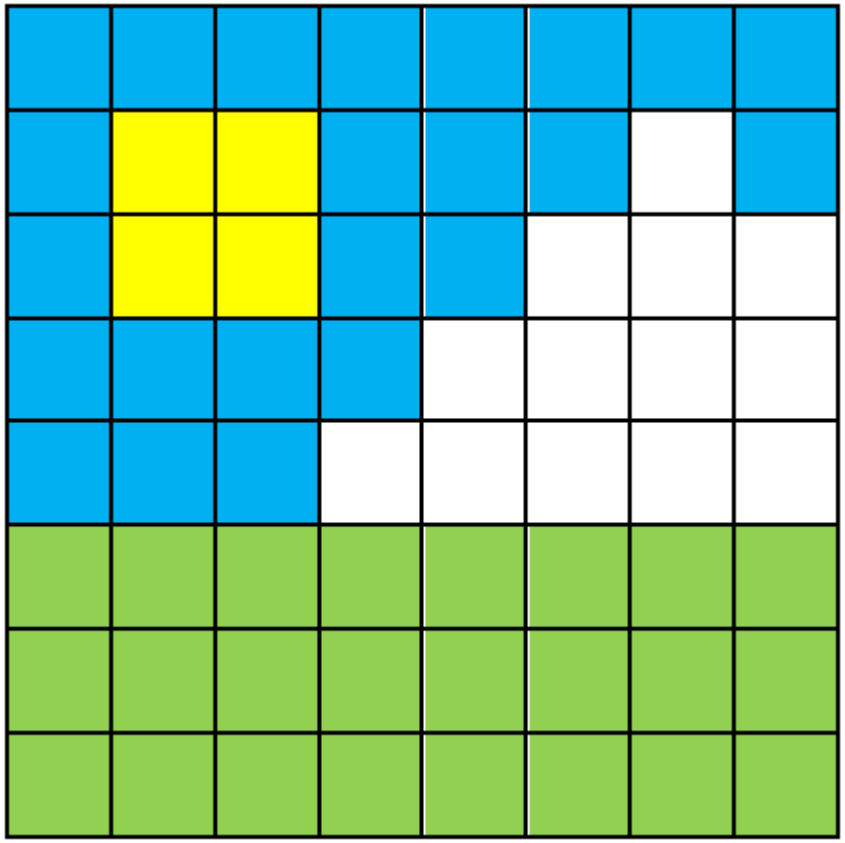
Lossless compression

Compression technique:

- LZW
- RLE
- Huffman

Run Length Encoding

- Imagine that we record each pixel, not by its bit pattern but by its colour blue (B), yellow (Y), white (W) and green (G)



B	B	B	B	B	B	B	B
B	Y	Y	B	B	B	W	B
B	Y	Y	B	B	W	W	W
B	B	B	B	W	W	W	W
B	B	B	W	W	W	W	W
G	G	G	G	G	G	G	G
G	G	G	G	G	G	G	G
G	G	G	G	G	G	G	G

Run Length Encoding

- List of color from top-left
- BBBBBBBBBYYBBBWBBYYBBWWWBBBBWWWWBBBW WWWWGGGGGGGGGGGGGGGGGGGGGGGG
- There are 4 colours, so we need 2 bits per pixel minimum, and we have 64 pixels in total; so we have 2 x 64 bits or **128** bits in total
- Count each of the blocks of colour and you get

  9B 2Y 3B 1W 2B 2Y 2B 3W 4B 5W 3B 5W 24G

Run Length Encoding

- Count each of the blocks of colour and you get

  9B 2Y 3B 1W 2B 2Y 2B 3W 4B 5W 3B 5W 24G

- Largest number is 24, so we need 5 bits to store the value and 2 bits to store the colour. We have 13 colour groups (9B is 1 group, 2Y is another group, etc.).
- So, we know we need 7 bits per group (5 for quantity, 2 for colour) and we have 13 groups; so this file will take up 7 bits x 13 or 91 bits.
- Our new file uses only 91 bits, compared to 128 bits for the original file. We have saved 37 bits in total

JPEG

Compress:

Block 8 × 8

↓

DCT

↓

Quantization (lossy)

↓

Lossless compress

↓

Compresed bit string

Decompress:

Block 8 × 8

↑

DCT^{-1}

↑

"Quantization^{-1}"

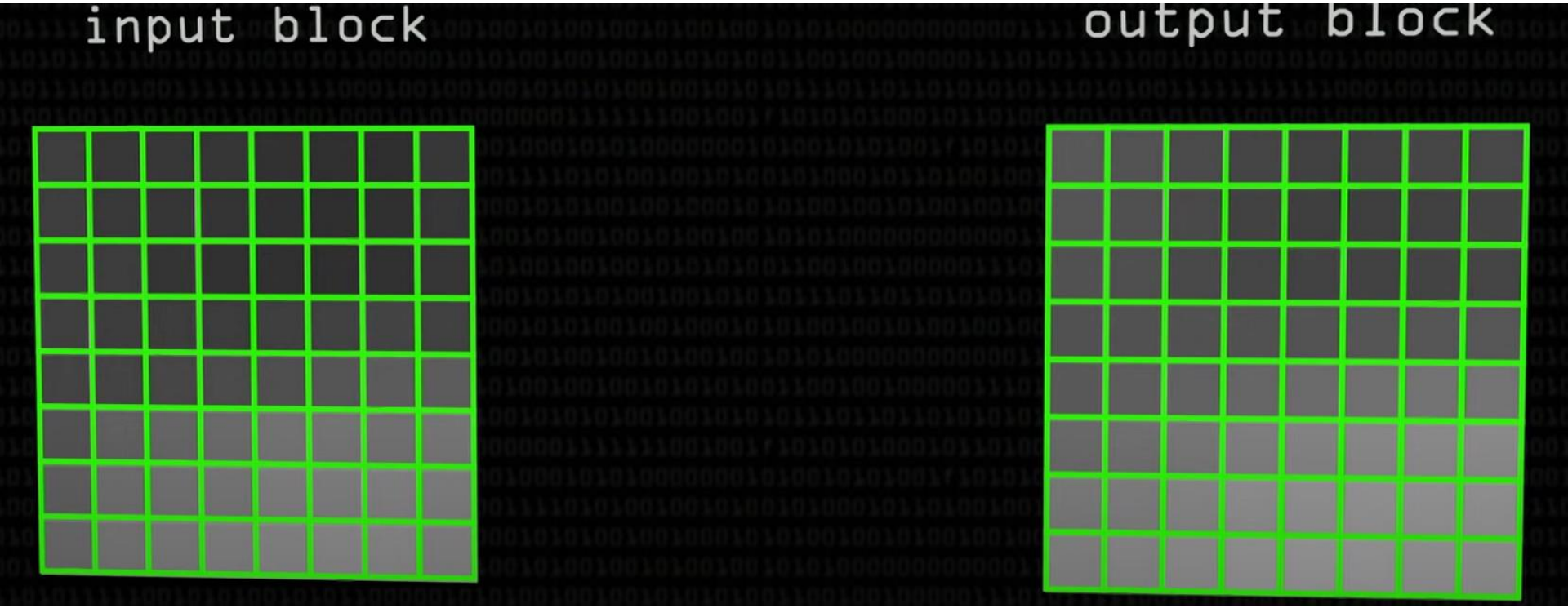
↑

Lossless decompress

↑

Compresed bit string

JPEG



JPEG

Compress:

Block 8 × 8

↓

DCT

↓

Quantization (lossy)

↓

Lossless compress

↓

Compresed bit string

Decompress:

Bl

Compresed bit string

Where to hide secret information during JPEG compression?

- Hide confidential information in the DCT quantized coefficient matrix, for example using the LSB .method
- Embedded LSB with all quantized coefficients DCT?
  - Invisibility: ☹
  - Compacity: ☹

Jpeg–Jsteg

- Source code can be found here.
- The secret messages are embedded in LSB of quantized DCT coefficients whose values are not 0, 1, or -1
- Capacity depends on number of non-zero coefficients
- If used zero:
  - Increase capacity, distorton, and image size
  - A clear indication of data hiding would be a low number of DCT coefficient with a zero value
- Cannot use 1. Because it can become 0
- -1 is ok.

Jpeg–Jsteg

$$\begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

(b)

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

(c)

$$\begin{bmatrix} 78 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(d)

(a) A block of 8 x 8 pixel values. (b) The DCT coefs (c) The quantized DCT coefficients. (d) The result of the coefs after the embedding step

Jpeg–Jsteg

- **Capacity**: limited.
  - If there are many quantifized coefficients equal to 0, 1, or 1, then the message capacity of Jpeg–Jsteg will be decreased.
- **Visibility**: not good
  - In DCT transformation, most important coefficients are located around the low-frequency part.
  - Jpeg–Jsteg modifies the quantized DCT coefficients right in the low-frequency part.

➔The image quality of Jpeg–Jsteg is degraded, especially when the cover-image undergoes a high compression ratio

Outguess

- Outguess is available for free download
- Hides in LSBs of DCT coefficients. Similar to Jsteg
- **First**: randomly select half of DCT coefficients (that are not **0** or **1)** to embedding LSB
  - To scatter the distortion
- **Second**: use 2nd haft to correct the alteration made to the histogram of DCT coefficients and adjust this histogram to its original value such that DCT histogram is preserved
  - Eg, changing “0” to “1” to embed a single bit requires changing “1” to “0” at the same time to correct the histogram of quantised DCT coefficients
  - Reduces capacity as some coefficients are used for correction
  - Makes detection more difficult

Where to hide secret information during JPEG compression?

Chang, Chen, & Chung proposed:

- Adjust the values in the quantization table corresponding to the 1 frequencies
- Embed the LSB in the quantized DCT coefficients for these mid frequencies

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 1 & 1 & 1 & 1 \\ 12 & 12 & 14 & 1 & 1 & 1 & 1 & 55 \\ 14 & 13 & 1 & 1 & 1 & 1 & 69 & 56 \\ 14 & 1 & 1 & 1 & 1 & 87 & 80 & 62 \\ 1 & 1 & 1 & 1 & 68 & 109 & 103 & 77 \\ 1 & 1 & 1 & 64 & 81 & 104 & 113 & 92 \\ 1 & 1 & 78 & 87 & 103 & 121 & 120 & 101 \\ 1 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Modified quantization table

Example

1260	-1	-12	-5	2	-2	-3	1
-23	-17	-6	-3	-3	0	0	1
-11	-9	-2	2	0	-1	-1	0
-7	-2	0	1	1	0	0	0
-1	-1	1	2	0	-1	1	1
2	0	2	0	-1	1	1	-1
-1	0	0	-1	0	2	1	-1
-3	2	-4	-2	2	1	-1	0

DCT coefficients

16	11	10	16	**1**	**1**	**1**	**1**
12	12	14	**1**	**1**	**1**	**1**	55
14	13	**1**	**1**	**1**	**1**	69	56
14	**1**	**1**	**1**	**1**	89	80	62
1	**1**	**1**	**1**	68	109	103	77
1	**1**	**1**	64	81	104	113	92
1	**1**	78	87	103	121	120	101
1	92	95	98	112	100	103	99

Modified quantization table

Secret bits 1001110011100100100100000

79	0	-1	0	2	-2	-3	1
-2	-1	0	-3	-3	0	0	0
-1	-1	-2	2	0	-1	0	0
-1	-2	0	1	1	0	0	0
-1	-1	1	2	0	0	0	0
2	0	2	0	0	0	0	0
-1	0	0	0	0	0	0	0
-3	0	0	0	0	0	0	0

Results of the quantizer

79	0	-1	0	2	-1	-3	0
-2	-1	0	-3	-2	1	0	0
-1	-1	-2	1	0	0	0	0
0	-2	0	1	1	0	0	0
-1	-1	1	2	0	0	0	0
2	0	2	0	0	0	0	0
-1	0	0	0	0	0	0	0
-3	0	0	0	0	0	0	0

After embedded

CCC Algorithm

Embedding algorithm

Input:

- msg_bits ***M***: bit string corresponding to secret message
- cover_image ***O***: image used to shield secret information
- quant_table ***Q***: quantization table

Output:

- stego_img_file ***E***: cover_image after embedded msg_bits

CCC Algorithm

Embedding algorithm

- **Step 1**: Input a cover-image ***O***. Suppose its size is $N \times N$ pixels. Partition the cover-image into non-overlapping blocks $\{O_1, O_2, \ldots, O_{\frac{N}{8}\times\frac{N}{8}}\}$. Each O_i contains 8×8 pixels.
- **Step 2**: Use DCT to transform each block O_i into DCT coefficient matrix F_i, where $F_i[a,b] = DCT(O_i[a,b])$, where $1 \leq a, b \leq 8$ and $O_i[a,b]$ is the pixel value in O_i
- **Step 3**: Use *modified quantization table* ***Q*** to quantize each F_i. The result can be represented as $C_i[a,b] = truncate(F_i[a,b]/\ Q_i[a,b])$

CCC Algorithm

Embedding algorithm

- **Step 4**: Select $C_i[a, b]$ to hide secret bit **S** respectively, where [a,b] equals to [0,4], [0,5], [0,6], [0,7], [1,3], [1,4], [1,5], [1,6], [2,2], [2,3], [2,4], [2,5], [3,1], [3,2], [3,3], [3,4], [4,0], [4,1], [4,2], [4,3], [5,0], [5,1], [5,2], [6,0], [6,1], and [7,0]

  Each $C_i[a, b]$ can embeds 1 or 2 secret bits into it.

- **Step 5**: Apply *JPEG entropy coding*, which contains *Huffman coding*, Run Length Encoding, and DPCM, to compress each block C_i. Collect the above results and generate a JPEG file ***E*** that contains the quantization table ***Q*** and all the compressed data.
- **Step 6**: Transfer the JPEG stego-image ***E*** to the receiver.

CCC Algorithm

Extracting algorithm

Input:

- stego_img_file ***E***: cover_image after embedded msg_bits

Output:

- msg_bits ***M***: bit string corresponding to secret message

CCC Algorithm

Extracting algorithm

Step 1: Use the first phase of JPEG decoding procedure to decompression the JPEG file. The decoding procedure contains Huffman decoding, RunLength decoding, and DPCM decoding.

Step 2: Extract the secret message ***S*** from the 26 middle-frequency coefficients Ci[0,4], Ci[0,5], Ci[0,6], Ci[0,7], Ci[1,3], Ci[1,4], Ci[1,5], Ci[1,6], Ci[2,2], Ci[2,3], Ci[2,4], Ci[2,5], Ci[3,1], Ci[3,2], Ci[3,3], Ci[3,4], Ci[4,0], Ci[4,1], Ci[4,2], Ci[4,3], Ci[5,0], Ci[5,1], Ci[5,2], Ci[6,0], Ci[6,1], and Ci[7,0], Where $1 \leq i \leq \frac{N}{8} \times \frac{N}{8}$. Collect those secret bits to regenerate the secret message

References

- 1. How are images Compressed, https://www.youtube.com/watch?v=Kv1Hiv3ox8I
- 2. JPEG DCT, Discrete Cosine Transform, https://www.youtube.com/watch?v=Q2aEzeMDHMA