Các kĩ thuật nén mất mát dữ liệu

Lossy Compression - 1

Image compression

- An image can be represented by a two-dimensional array (table) of picture elements (pixels).
  - A grayscale picture of 307,200 pixels is represented by 2,457,600 bits.
  - A color picture of 307,200 pixels is represented by 7,372,800 bits.

- Image compression
  - Goal: reduce the size of stored files and data while retaining all necessary perceptual information
  - Used to create an encoded copy of the original data with a (much) smaller size

LOSSY COMPRESSION METHODS

- Our eyes and ears cannot distinguish subtle changes

→ In such cases, we can use a lossy data compression method.

http://efilmvn.com/codec-phan-4-ky-thuat-nen/

LOSSY COMPRESSION METHODS

- Our eyes and ears cannot distinguish subtle changes

→ In such cases, we can use a lossy data compression method.

- These methods are cheaper—they take less time and space when it comes to sending millions of bits per second for images and video.
- Several methods have been developed using lossy compression techniques. JPEG (Joint Photographic Experts Group) encoding is used to compress pictures and graphics, MPEG (Moving Picture Experts Group) encoding is used to compress video, and MP3 (MPEG audio layer 3) for audio compression.

Nội dung

- Thuật toán JPEG encoding

Facts about JPEG

- JPEG - Joint Photographic Experts Group
- International standard: 1992
- Most popular format
  - Other formats (.bmp) use similar techniques
- **Lossy** image compression
  - **transform coding** using the DCT
- JPEG 2000
  - New generation of JPEG – well, never succeeds
  - DWT (*Discrete Wavelet Transform)*

Facts about JPEG

- Compression Ratio = Uncompressed Size/Compressed Size

	Typical Compression Ratios				
	GIF	JPEG(low)	JPEG(mid)	JPEG(high)	PNG
Min	4:1	10:1	30:1	60:1	10-3-%
Max	10:1	20:1	50:1	100:1	More than GIF

Visual Example

- The following JPEGs are compressed with different ratios

1:1(low) 1:10(low) 1:30(mid) 1:30 with 5Xzoom

(a) (b) (c) (d)

Three Major Observations

- **Observation 1**:
  - Useful image contents change relatively slowly across the image, i.e., it is unusual for intensity values to vary widely several times in a small area, for example, within an 8x8 image block.

Three Major Observations

- **Observation 1**:
  - Useful image contents change relatively slowly across the image, i.e., it is unusual for intensity values to vary widely several times in a small area, for example, within an 8x8 image block.

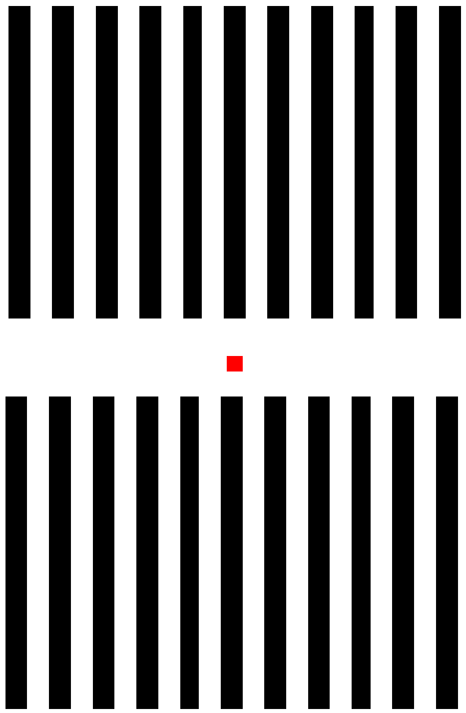
    - much of the information in an image is repeated, hence "spatial redundancy".

Compression Ratio: 7.7

Compression Ratio: 33.9

Observations

- **Observation 2**:
  - Psychophysical experiments suggest that humans are much less likely to notice the loss of very high spatial frequency components than the loss of lower frequency components.



Adapted from Blakemore & Sutton, 1969

Observations

- **Observation 2**:
  - Psychophysical experiments suggest that humans are much less likely to notice the loss of very high spatial frequency components than the loss of lower frequency components.

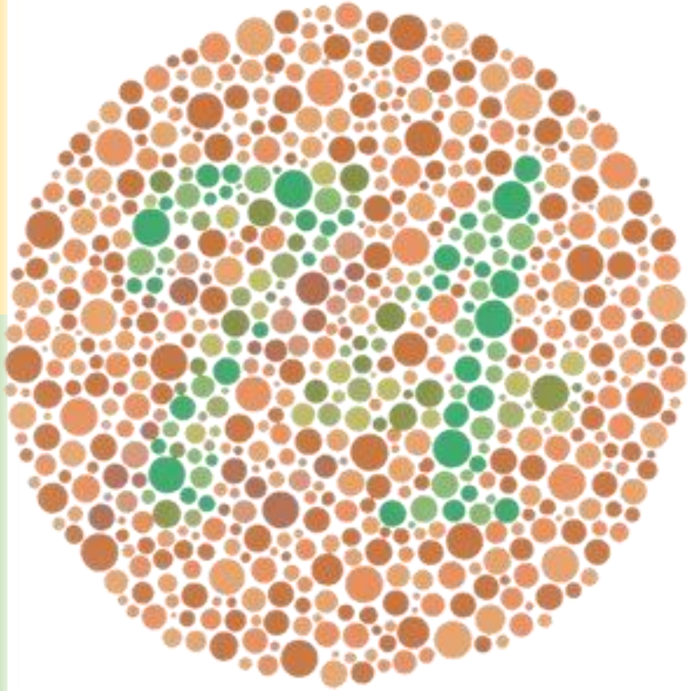
    - the spatial redundancy can be reduced by largely reducing the high spatial frequency contents.

Compression Ratio: 7.7

Compression Ratio: 33.9

Observations

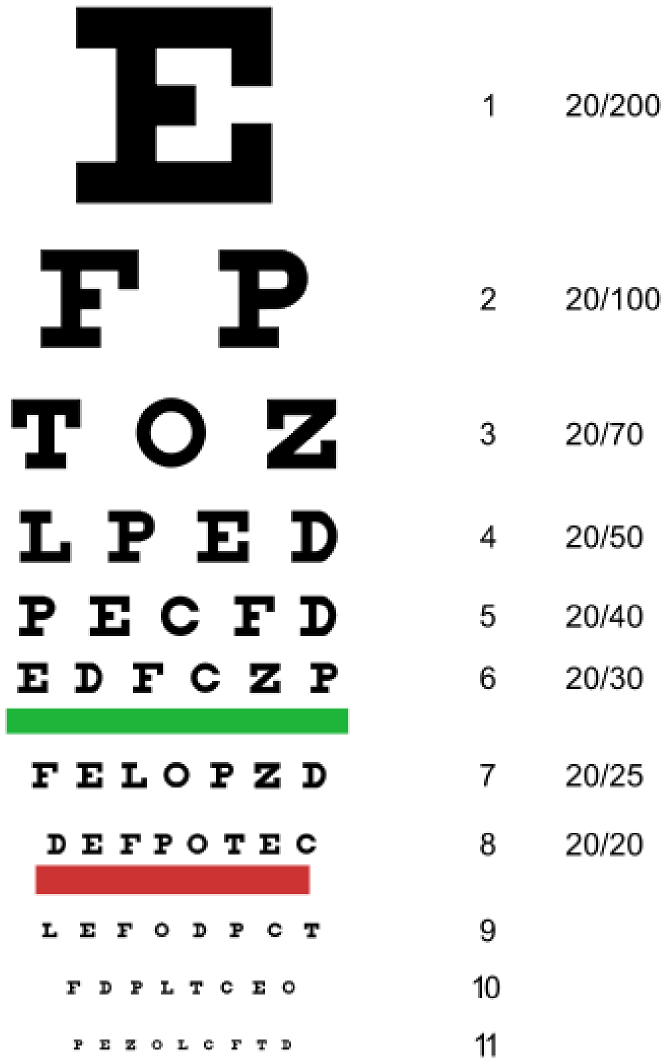
- **Observation 3**:
  - Visual acuity (accuracy in distinguishing closely spaced lines) is much greater for gray (black and white) than for color.



Example of an Ishihara color test plate. With properly configured computer displays, people with normal vision should see the number "74". Many people who are color blind see it as "21", and those with total color blindness may not see any numbers.

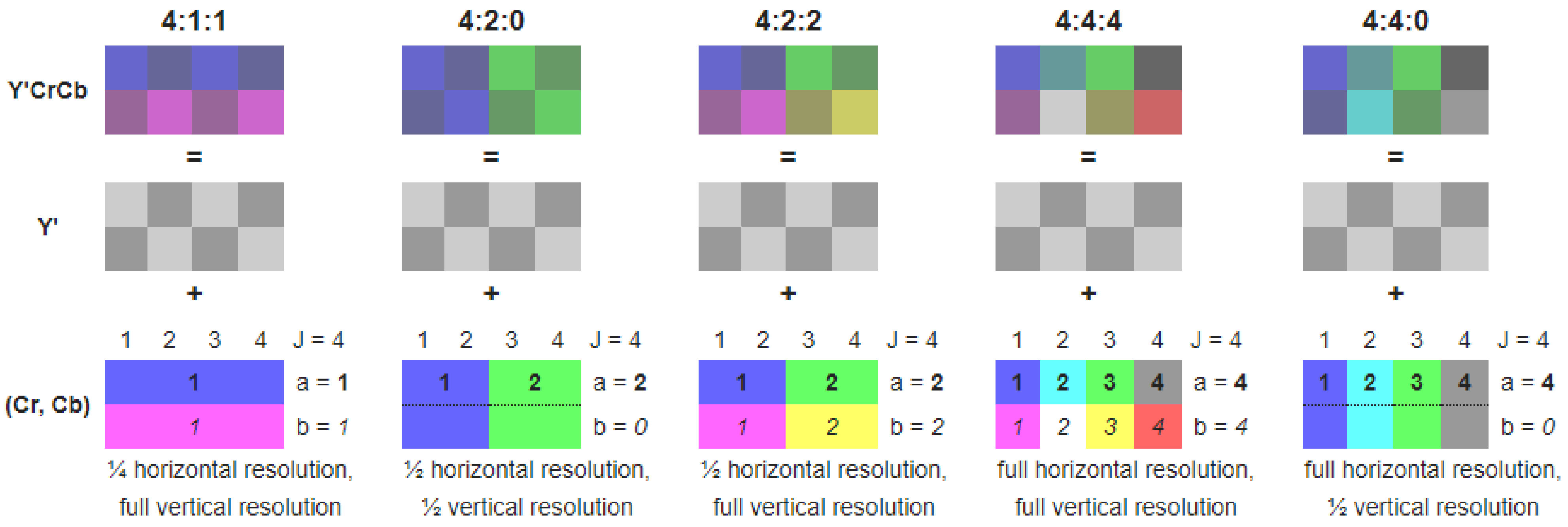
https://en.wikipedia.org/wiki/Color_blindness

Snellen chart



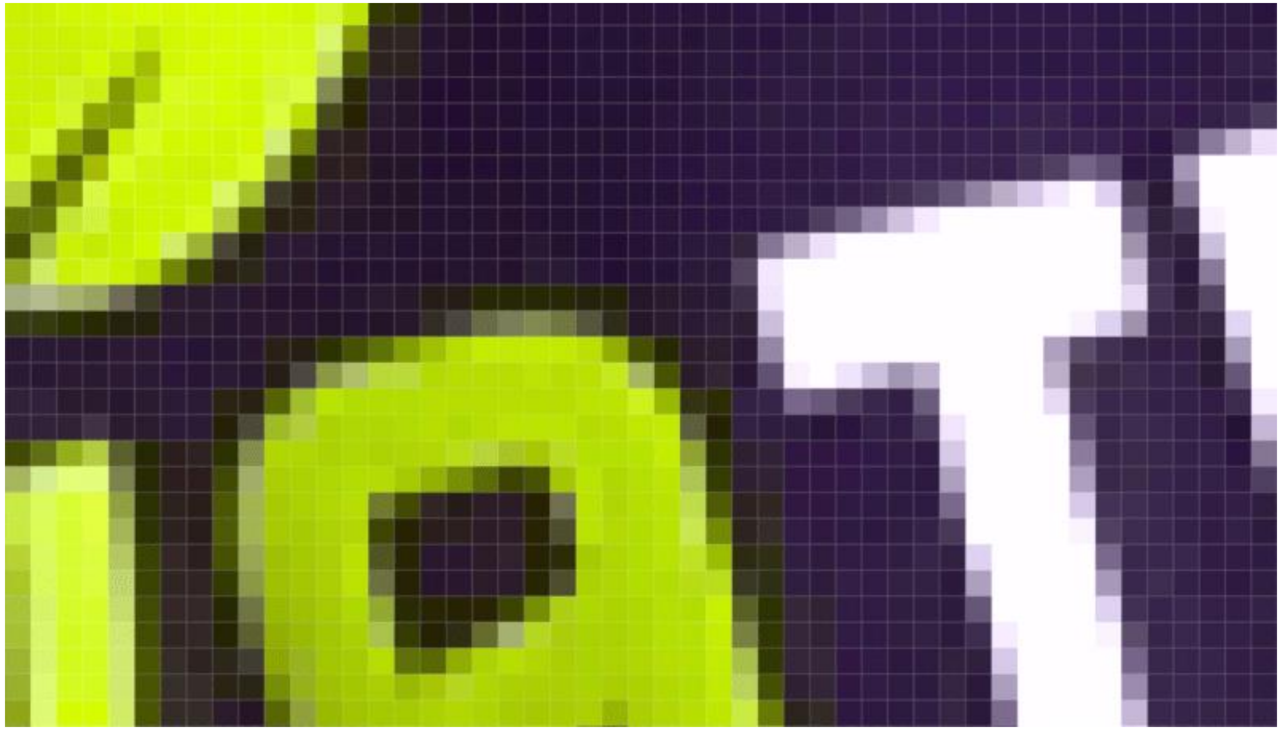
Observations

- **Observation 3**:
  - Visual acuity (accuracy in distinguishing closely spaced lines) is much greater for gray (black and white) than for color.
    - chroma subsampling (4:2:0) is used in JPEG.



Ex: chroma subsampling

- The dark green color visible in the edge pixels of the letter is not actually present on real life object. It is merely a result of the chroma subsampling removing the chroma data of the blue pixels and replacing it with the chroma of the green pixels, but with the same luma information. There is some data loss, obviously, but in normal viewing conditions (when you're not looking at massively zoomed in pixels), this will be hardly noticeable.



https://workflow.frame.io/guide/chroma-subsampling

JPEG encoding

JPEG encoding

- In JPEG, a grayscale picture is divided into blocks of 8 × 8 pixel blocks to decrease the number of calculations because, as we will see shortly, the number of mathematical operations for each picture is the square of the number of units.

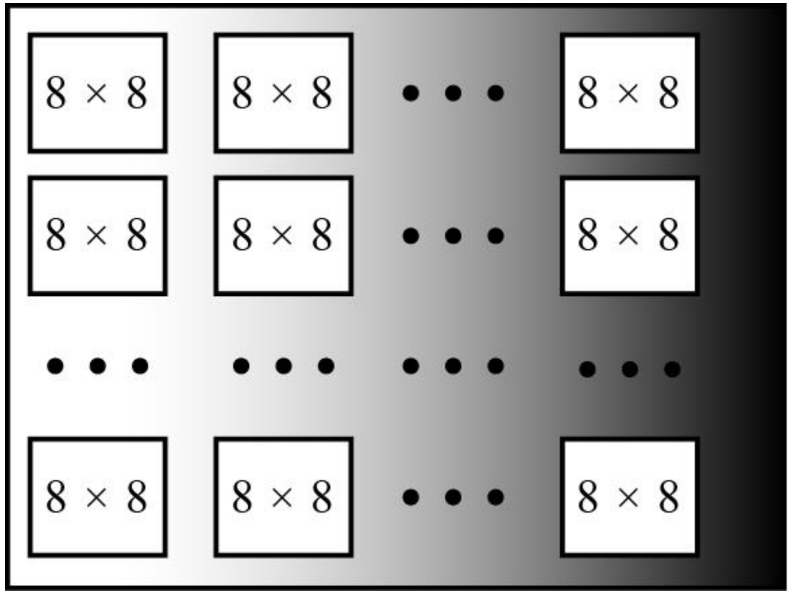


Figure 15.10 **JPEG grayscale example, 640 × 480 pixels**

Why does JPEG use 8x8 ?

- Discrete cosine transforms are most efficient computationally when working on a $2^n \times 2^n$ signal.
- → 8 X 8 was chosen after numerous experiments with other sizes.
- In theory you could also use 32x32 or 64x64 blocks, but that increases the risk that your blocks aren't just "part of a curtain" or "part of the sky" but may include parts of each, which destroys your ability to perform useful compression.
- Conversely, you could use 2x2 or 4x4 blocks, but then you're probably not going to achieve much compression.

Block Effect

- Using blocks, however, has the effect of isolating each block from its neighboring context.
  - choppy (“blocky") with high *compression ratio*

Compression Ratio: 7.7

Compression Ratio: 33.9

Compression Ratio: 60.1

JPEG encoding

- The whole idea of JPEG is to change the picture into a linear (vector) set of numbers that reveals the redundancies.
- The redundancies (lack of changes) can then be removed using one of the lossless compression methods we studied previously.
- A simplified version of the process is shown.

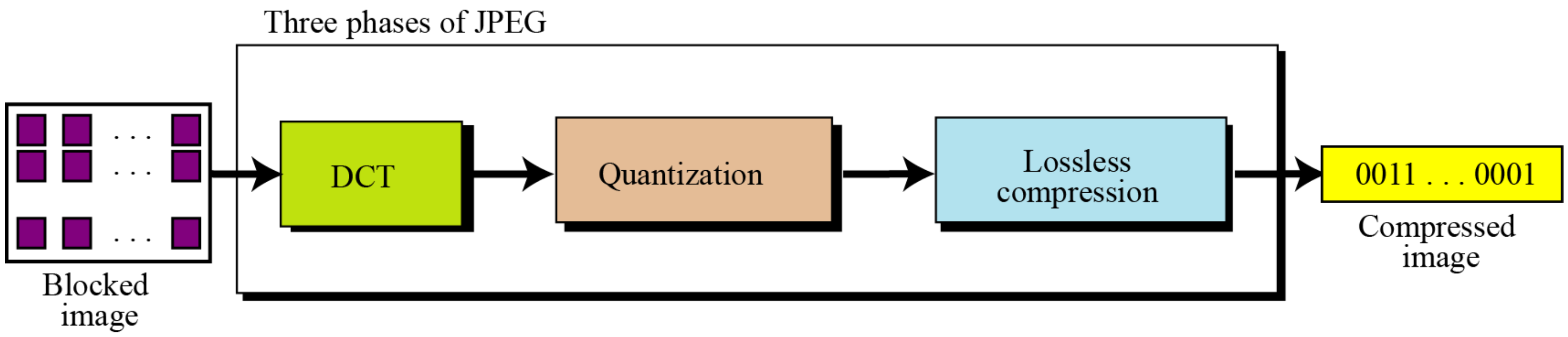
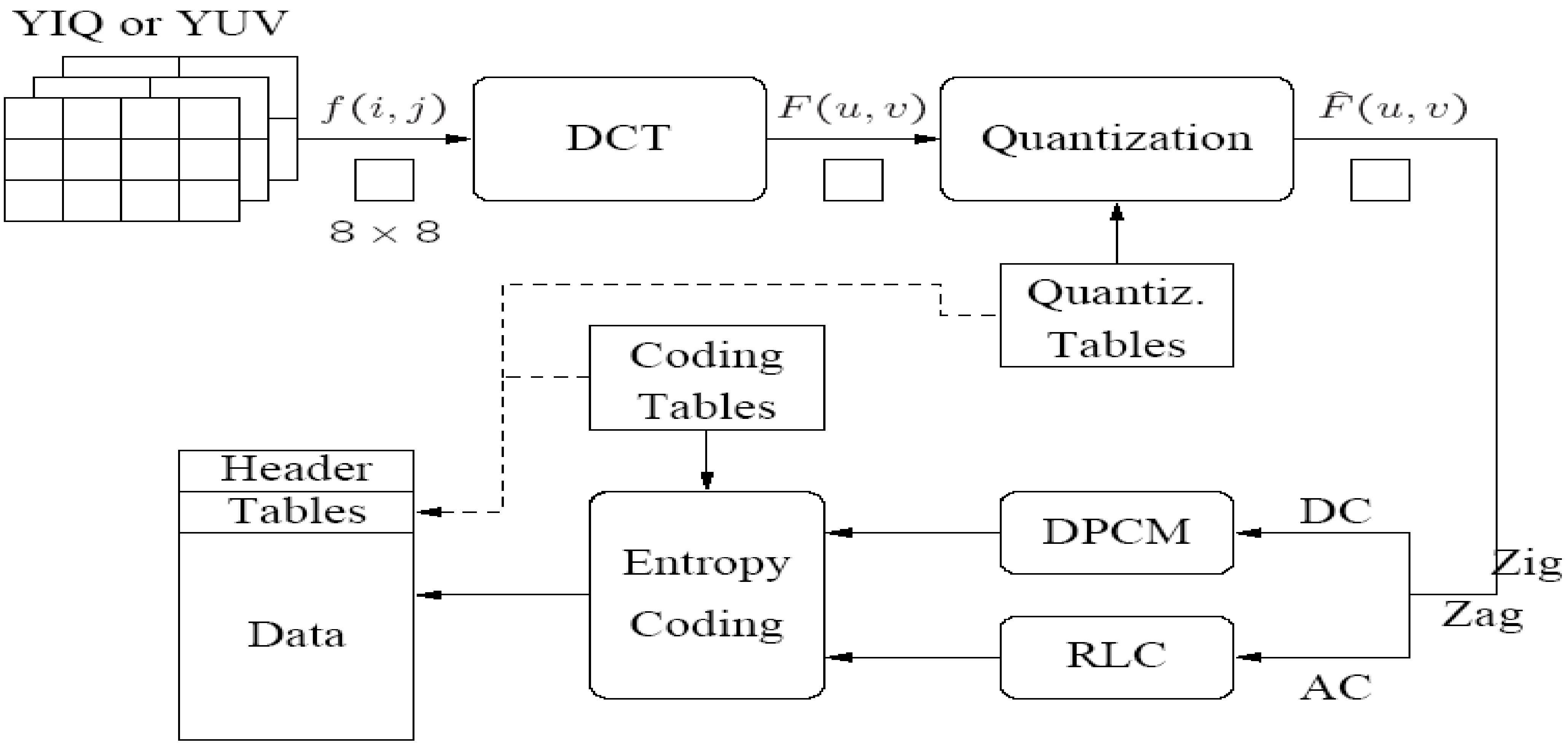


Figure 15.11 **The JPEG compression process**

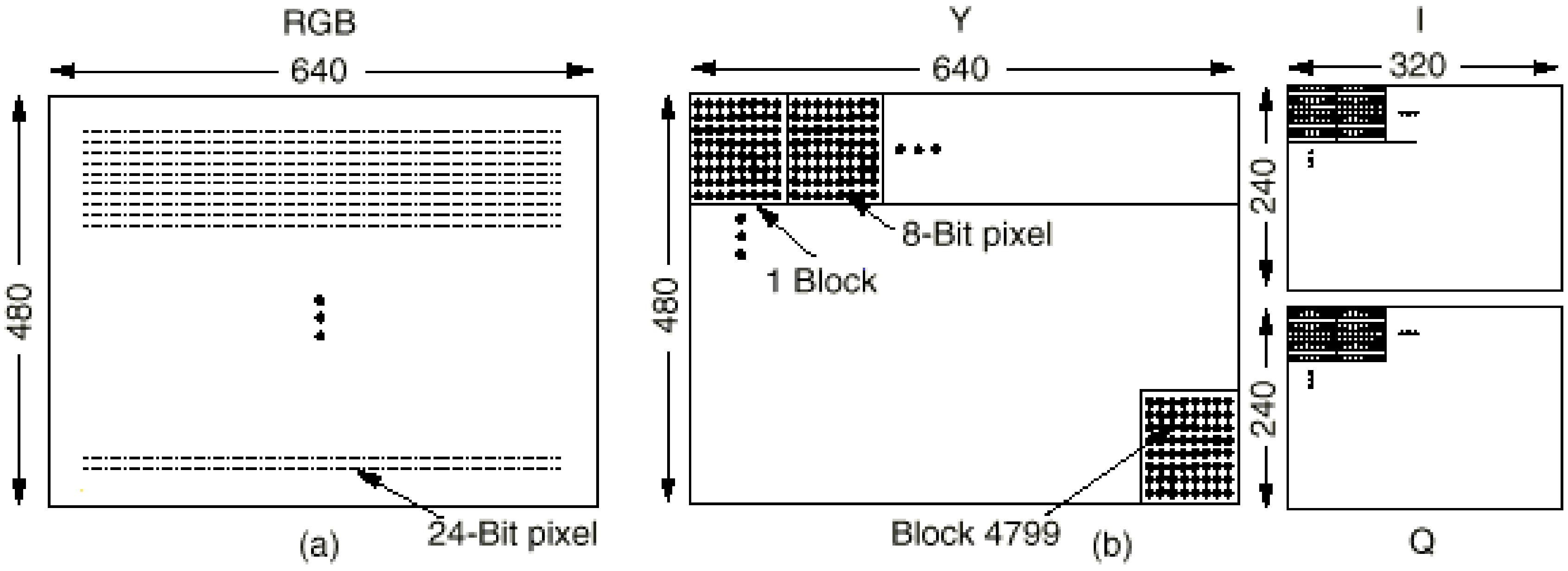
JPEG Steps

1. Block Preparation
    - RGB to YUV (YIQ) planes
2. Transform
    - 2D Discrete Cosine Transform (DCT) on 8x8 blocks.
3. Quantization
    - Quantized DCT Coefficients (lossy).
4. Encoding of Quantized Coefficients
    - Zigzag Scan
    - Differential Pulse Code Modulation (DPCM) on DC component
    - Run Length Encoding (RLE) on AC Components
    - Entropy Coding: Huffman or Arithmetic

JPEG Diagram



JPEG: Block Preparation



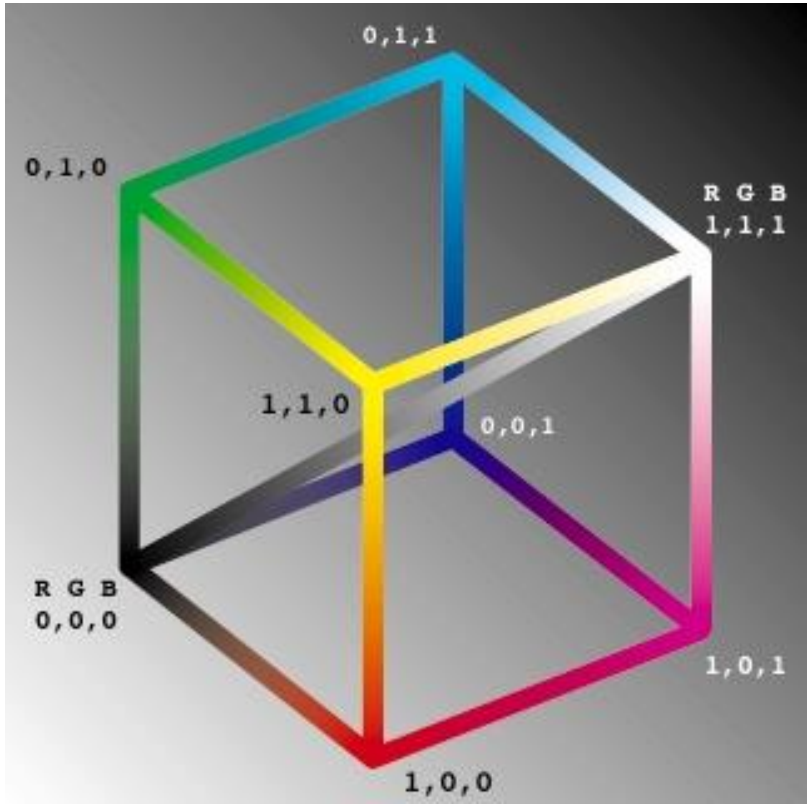
RGB Input Data **After Block Preparation**

Input image: 640 x 480 RGB (24 bits/pixel) transformed to three planes:

Y: (640 x 480, 8-bit/pixel) Luminance (brightness) plane.
U, V: (320 X 240 8-bits/pixel) Chrominance (color) planes.

Color Spacing

- A pixel's color is determined by its RGB (red blue green) value
  - Eg. R=30, G=100, B=50



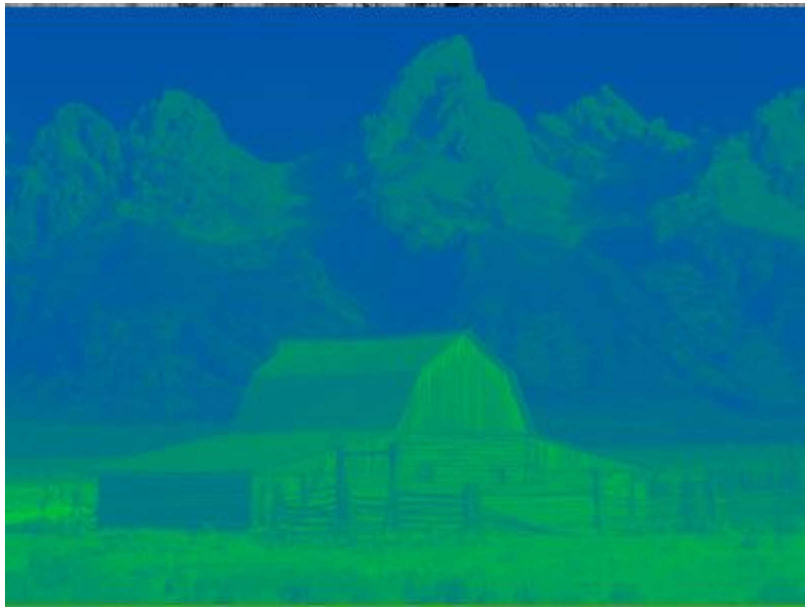
Color Spacing

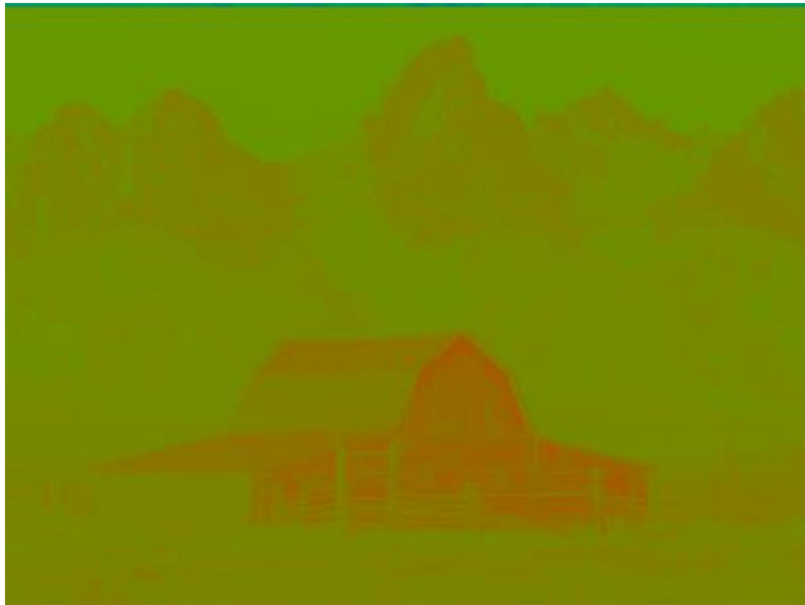
- A pixel's color is determined by its RGB (red blue green) value
  - Eg. R=30, G=100, B=50

- Image formats using lossy compression often convert this data into a format that separates luminance (brightness) and chrominance (hue)

YUV/YCbCr Coordinates

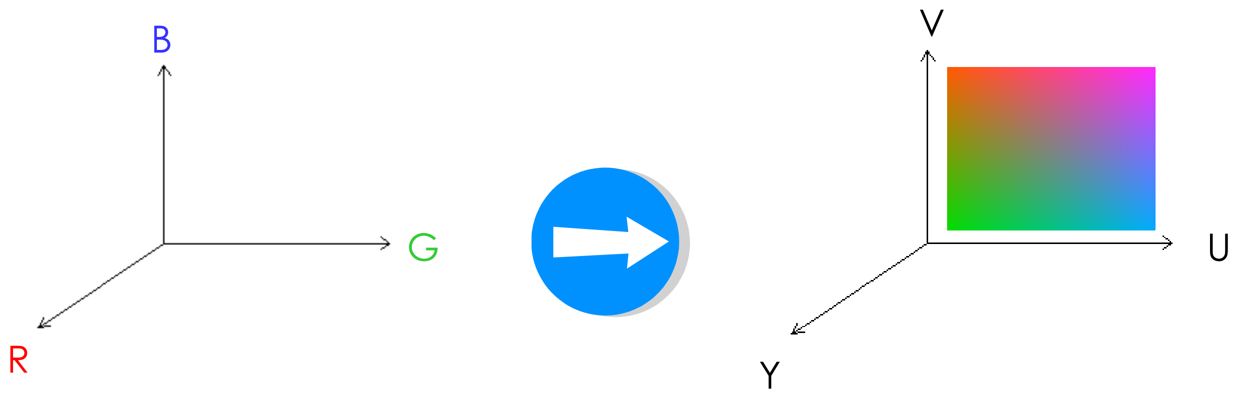
- Define the luminance coordinate to be:
  - Y = 0.299R + 0.587G + 0.114B (luma component)
- Define the color differences coordinates to be:
  - U = B – Y (blue projection)
  - V = R – Y (red projection)

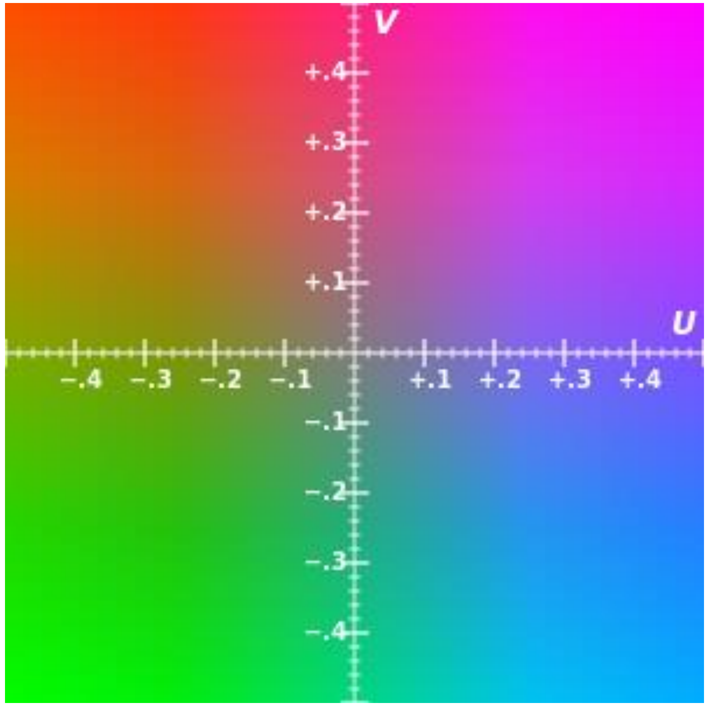




YUV/YCbCr Coordinates

- This transforms the RGB color data to the YUV system which is easily reversible.





- It applies the DCT filtering independently to Y, U, and V

Nasir Ahmed, the inventor of the discrete cosine transform (DCT), which he first proposed in 1972.

Discrete cosine transform (DCT)

One-dimensional DCT

Definition: Let n be a positive integer. The one-dimensional **DCT** of order n is defined by an n x n matrix C whose entries are

$$C_{ij} = a_i \cos \frac{i(2j+1)\pi}{2n}$$

The Advantage of Orthogonality

- C **orthogonal**: $C^TC = I$
- Implies $C^{-1} = C^T$

- Makes solving matrix equations easy

- Solve $Y = CXC^T$ for X:
- $C^TY = C^TCXC^Y = XC^T$
- $C^TYC = XC^TC = X$

One-dimensional DCT

- The discrete cosine transform, C, has one basic characteristic: it is a real orthogonal matrix.

$$C = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{(2n-1)\pi}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \cos\frac{(n-1)\pi}{2n} & \cos\frac{(n-1)3\pi}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

One-dimensional DCT

- The discrete cosine transform, C, has one basic characteristic: it is a real orthogonal matrix.

$$C = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{(2n-1)\pi}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \cos\frac{(n-1)\pi}{2n} & \cos\frac{(n-1)3\pi}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

$$C^{-1} = C^T = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\frac{\pi}{2n} & \cdots & \cos\frac{(n-1)\pi}{2n} \\ \frac{1}{\sqrt{2}} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{(n-1)3\pi}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{\sqrt{2}} & \cos\frac{(2n-1)\pi}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

One-dimensional DCT

- **DCT Interpolation Theorem**

Suppose we are given a vector

$$x = [x_0, \ldots, x_{n-1}]^T$$

The Discrete Cosine Transform of x is the n-dimensional vector

$$y = [y_0, \ldots, y_{n-1}]^T$$

Where C is defined as $y = Cx$

One-dimensional DCT

- **DCT Interpolation Theorem**

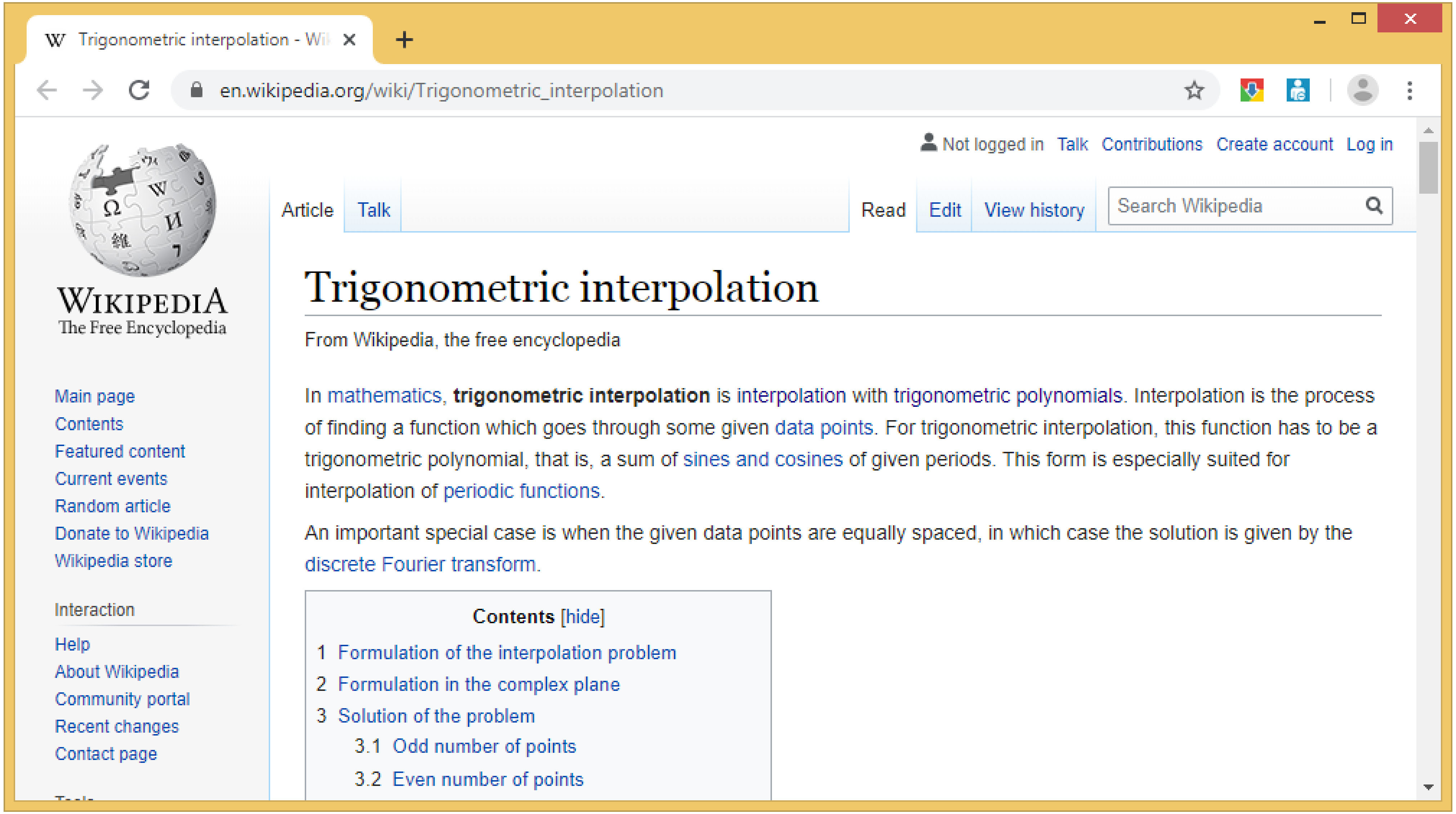
The DCT interpolating function $P_n(x)$ is given by,

$$P_n(t) = \frac{1}{\sqrt{n}} y_0 + \frac{\sqrt{2}}{\sqrt{n}} \sum_{k=1}^{n-1} y_k \cos \frac{k(2t+1)\pi}{2n}$$

satisfies $P_n(j)=x_j$ for $j=0,\ldots, n-1$

C transforms the n data points into n interpolation coefficients. The DCT provides coefficients for the <u>trigonometric interpolation</u> function using only cosine terms.

One-dimensional DCT



Trigonometric interpolation - Wi

en.wikipedia.org/wiki/Trigonometric_interpolation

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Trigonometric interpolation

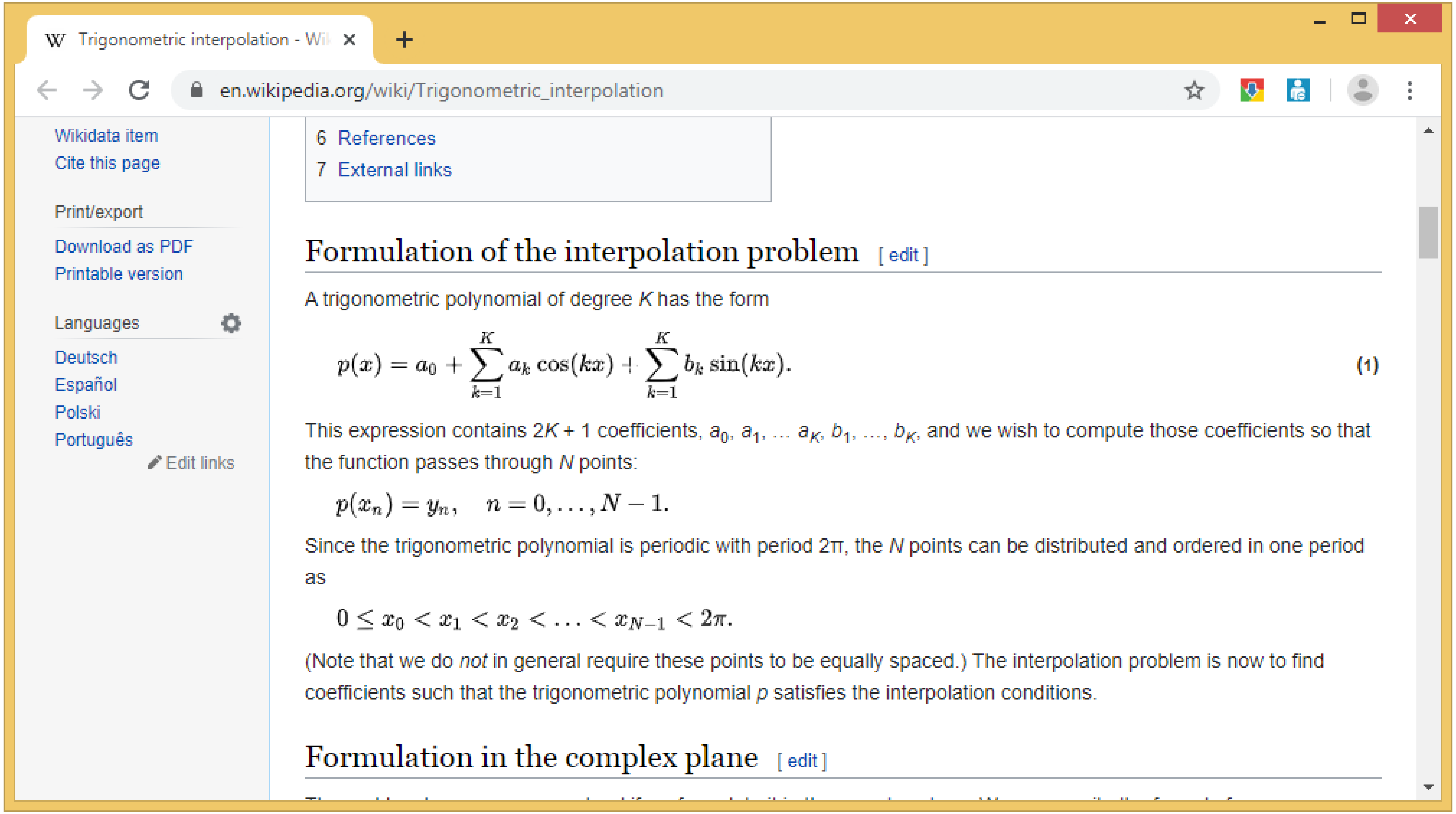
From Wikipedia, the free encyclopedia

In mathematics, **trigonometric interpolation** is interpolation with trigonometric polynomials. Interpolation is the process of finding a function which goes through some given data points. For trigonometric interpolation, this function has to be a trigonometric polynomial, that is, a sum of sines and cosines of given periods. This form is especially suited for interpolation of periodic functions.

An important special case is when the given data points are equally spaced, in which case the solution is given by the discrete Fourier transform.

Contents [hide]

One-dimensional DCT



Trigonometric interpolation - Wi

en.wikipedia.org/wiki/Trigonometric_interpolation

Wikidata item
Cite this page

Print/export

Download as PDF
Printable version

Languages

Deutsch
Español
Polski
Português

Edit links

Formulation of the interpolation problem [edit]

A trigonometric polynomial of degree K has the form

$$p(x) = a_0 + \sum_{k=1}^{K} a_k \cos(kx) + \sum_{k=1}^{K} b_k \sin(kx). \quad (1)$$

This expression contains $2K + 1$ coefficients, a_0, a_1, … a_K, b_1, …, b_K, and we wish to compute those coefficients so that the function passes through N points:

$$p(x_n) = y_n, \quad n = 0, \ldots, N-1.$$

Since the trigonometric polynomial is periodic with period 2π, the N points can be distributed and ordered in one period as

$$0 \le x_0 < x_1 < x_2 < \ldots < x_{N-1} < 2\pi.$$

(Note that we do *not* in general require these points to be equally spaced.) The interpolation problem is now to find coefficients such that the trigonometric polynomial p satisfies the interpolation conditions.

Formulation in the complex plane [edit]

One-dimensional DCT

- **Least Squares Approximation Theorem**

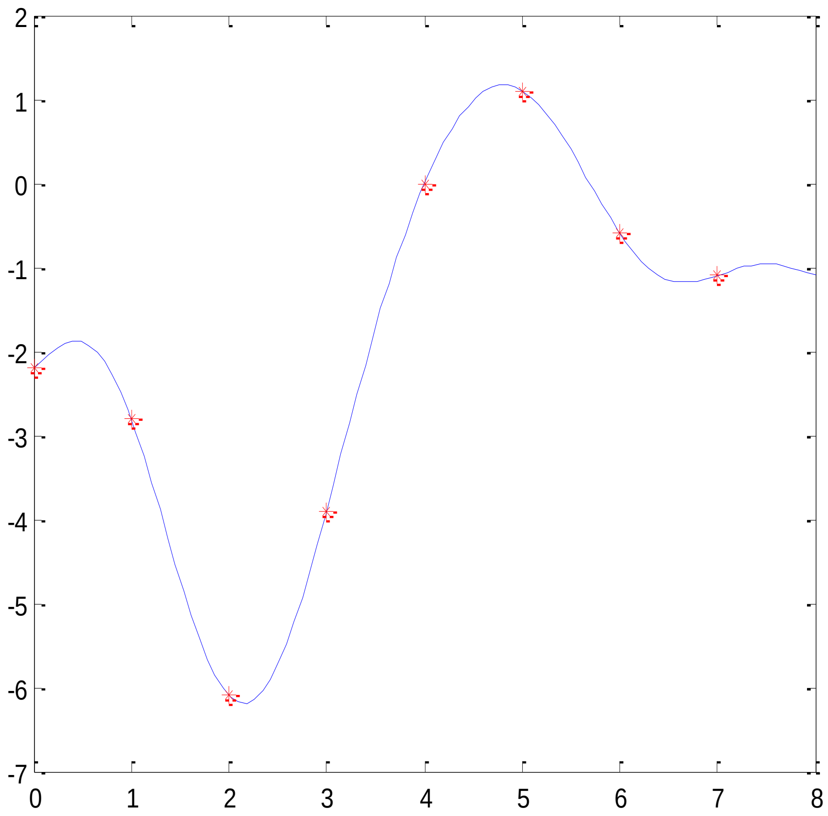
$$x = [x_0, \ldots, x_{n-1}]^T$$

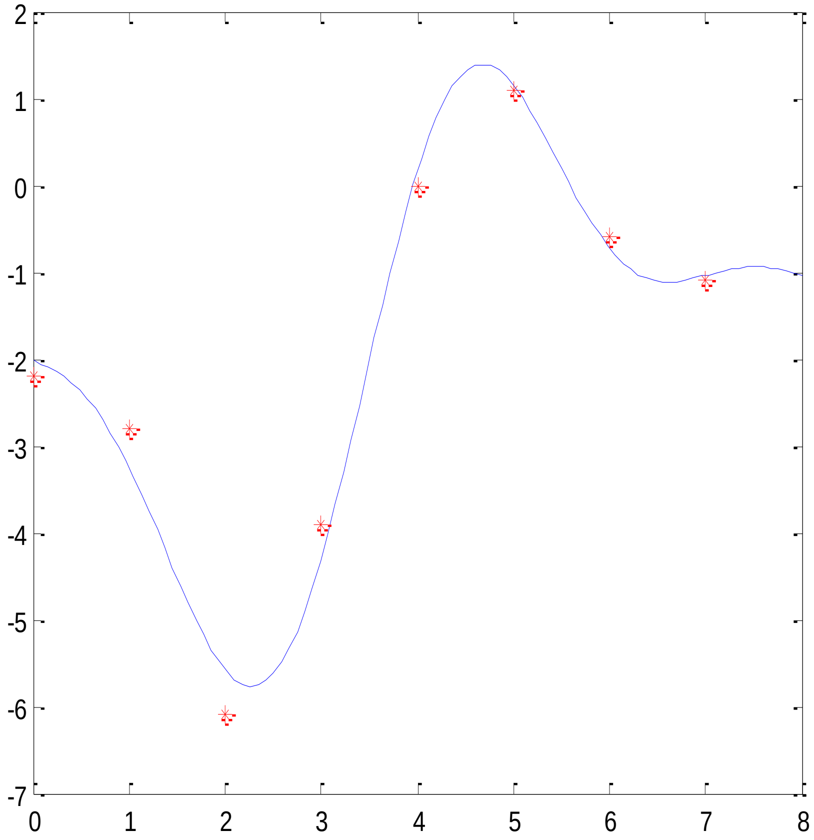
$$y = [y_0, \ldots, y_{n-1}]^T = Cx$$

For any positive integer $m \leq n$, the DCT least squares approximation with m is given by,

$$P_m(t) = \frac{1}{\sqrt{n}} y_0 + \frac{\sqrt{2}}{\sqrt{n}} \sum_{k=1}^{m-1} y_k \cos \frac{k(2t+1)\pi}{2n}$$

DCT Interpolation & Approximation

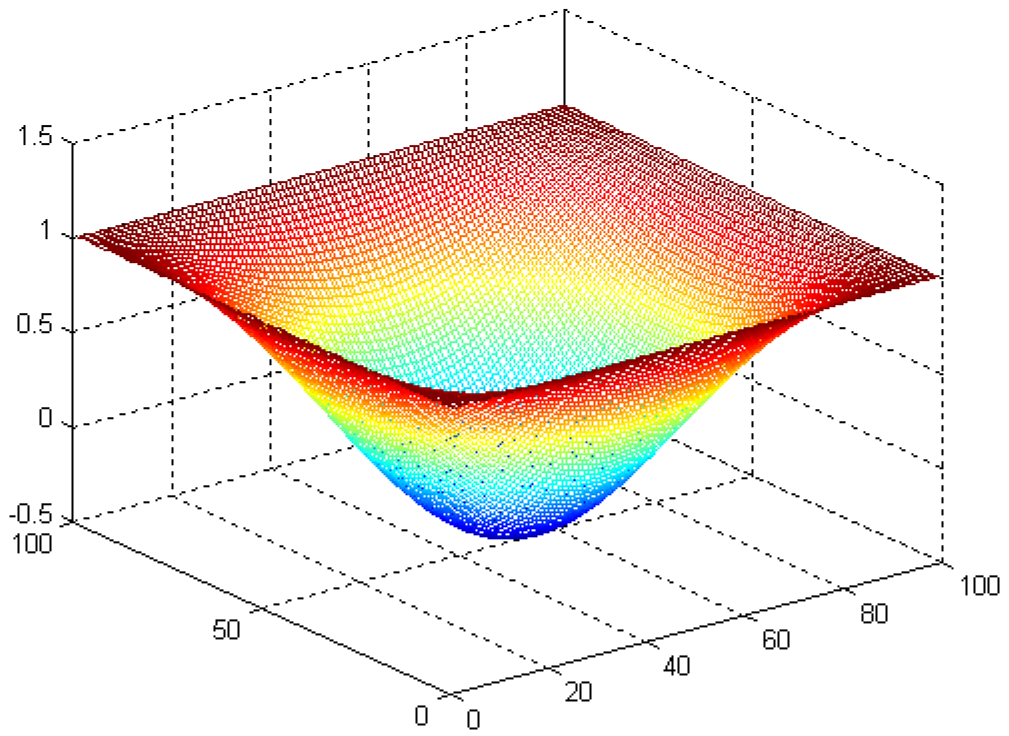




2-D DCT Interpolation

Given a matrix of 16 data points we can plot the surface in 3-D space.

$$\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{matrix}$$



2-D Least Squares

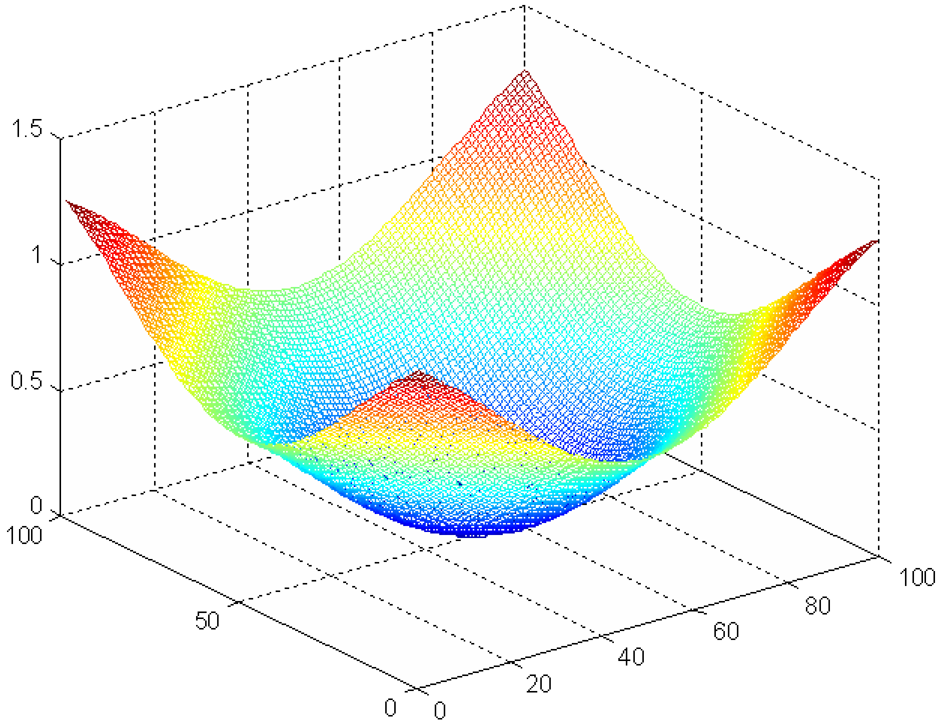
- Done in the same way as with 1-D
- Implement a low pass filter (drop terms)
- Delete the “high-frequency” components

2-D Least Squares

Least Squares

Approximation

1.25	0.75	0.75	1.25
0.75	0.25	0.25	0.75
0.75	0.25	0.25	0.75
1.25	0.75	0.75	1.25



Sizeable Error due to small number of points

3. Find the DCT of the following data vectors x, and find the corresponding interpolating function $P_n(t)$ for the data points $(i, x_i), i = 0, \ldots, n-1$ (you may state your answers in terms of the b and c defined in (11.7)):

(a)

t	x
0	1
1	0
2	1
3	0

(b)

t	x
0	1
1	1
2	1
3	1

(c)

t	x
0	1
1	0
2	0
3	0

(d)

t	x
0	1
1	2
2	3
3	4

(a)

t	x
0	3
1	3

(b)

t	x
0	2
1	−2

(c)

t	x
0	3
1	1

(d)

t	x
0	4
1	−1

Two-Dimensional DCT

- Idea 2D-DCT: Interpolate the data with a set of basis functions
- Organize information by order of importance to the human visual system
- Used to compress small blocks of an image

  (8 x 8 pixels in our case)

Two-Dimensional DCT

Use One-Dimensional DCT in both horizontal and vertical directions.

First direction $F = C*X^T$

Second direction $G = C*F^T$

We can say 2D-DCT is the matrix:

$$Y = C(CX^T)^T$$

Image compression – JPEG encoding

Image compression – JPEG encoding

- Now we have found the matrix

  $Y = C(CX^T)^T$

- Using the DCT, the entries in Y will be organized based on the human visual system.

- The most important values to our eyes will be placed in the upper left corner of the matrix.

- The least important values will be mostly in the lower right corner of the matrix.

Most Important

-304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1

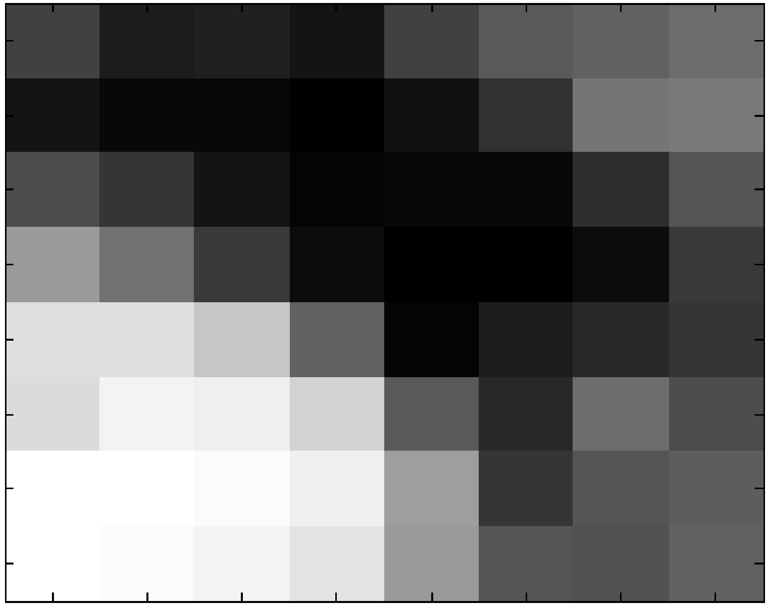
Semi-Important

Least Important

Image compression – JPEG encoding

- Each block of 64 pixels goes through a transformation called the discrete cosine transform (DCT).
- The transformation changes the 64 values so that the relative relationships between pixels are kept but the redundancies are revealed.

8 x 8 Pixels



Image

Image compression – JPEG encoding

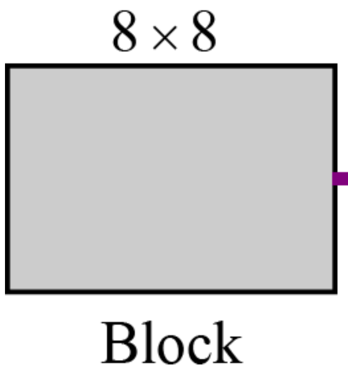
- In Mathematical terms:
  - Let $X = (x_{ij})$ be a matrix of n^2 real numbers
  - $Y = (y_{kl})$ be the 2D-DCT of X
  - $a_0 = 1/sqrt(2) \quad and \quad a_k = 1 \quad for\, k > 0$
  - Then: $$P_n(s,t) = \frac{2}{n} a_s a_t \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} x_{kl} \cos\frac{s(2k+1)\pi}{2n} \cos\frac{t(2l+1)\pi}{2n}$$
  - Satisfies $P_n(k,l) = y_{kl} \; for\, k,l = 0,\dots,n-1$

Image compression – JPEG encoding

$$
\begin{bmatrix}
\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\
\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{11\pi}{16} & \cos\frac{13\pi}{16} & \cos\frac{15\pi}{16} \\
\cos\frac{2\pi}{16} & \cos\frac{6\pi}{16} & \cos\frac{10\pi}{16} & \cos\frac{14\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{22\pi}{16} & \cos\frac{26\pi}{16} & \cos\frac{30\pi}{16} \\
\cos\frac{3\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{27\pi}{16} & \cos\frac{33\pi}{16} & \cos\frac{39\pi}{16} & \cos\frac{45\pi}{16} \\
\cos\frac{4\pi}{16} & \cos\frac{12\pi}{16} & \cos\frac{20\pi}{16} & \cos\frac{28\pi}{16} & \cos\frac{36\pi}{16} & \cos\frac{44\pi}{16} & \cos\frac{52\pi}{16} & \cos\frac{60\pi}{16} \\
\cos\frac{5\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{25\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{45\pi}{16} & \cos\frac{55\pi}{16} & \cos\frac{65\pi}{16} & \cos\frac{75\pi}{16} \\
\cos\frac{6\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{30\pi}{16} & \cos\frac{42\pi}{16} & \cos\frac{54\pi}{16} & \cos\frac{66\pi}{16} & \cos\frac{78\pi}{16} & \cos\frac{90\pi}{16} \\
\cos\frac{7\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{49\pi}{16} & \cos\frac{63\pi}{16} & \cos\frac{77\pi}{16} & \cos\frac{91\pi}{16} & \cos\frac{105\pi}{16}
\end{bmatrix}
$$

Image compression – JPEG encoding

- To understand the nature of this transformation, let us show the result of the transformations for three cases.



$$\begin{bmatrix} 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20\\ 20&20&20&20&20&20&20&20 \end{bmatrix} \rightarrow \begin{bmatrix} 160&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

P (x, y) T (m, n)

P(x, y) defines one value in the block, while T(m, n) defines the value in the transformed block.

Figure 15.12 **Case 1: uniform grayscale**

Image compression – JPEG encoding

- To understand the nature of this transformation, let us show the result of the transformations for three cases.

8×8

Block

20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50

P (x, y)

200	–100	0	39	0	–25	–1	22
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

T (m, n)

Figure 15.13 **Case 2: two sections**

Image compression – JPEG encoding

- To understand the nature of this transformation, let us show the result of the transformations for three cases.

8 × 8

Block

20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90

P (x, y)

400	–146	0	–31	–1	3	–1	–8
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

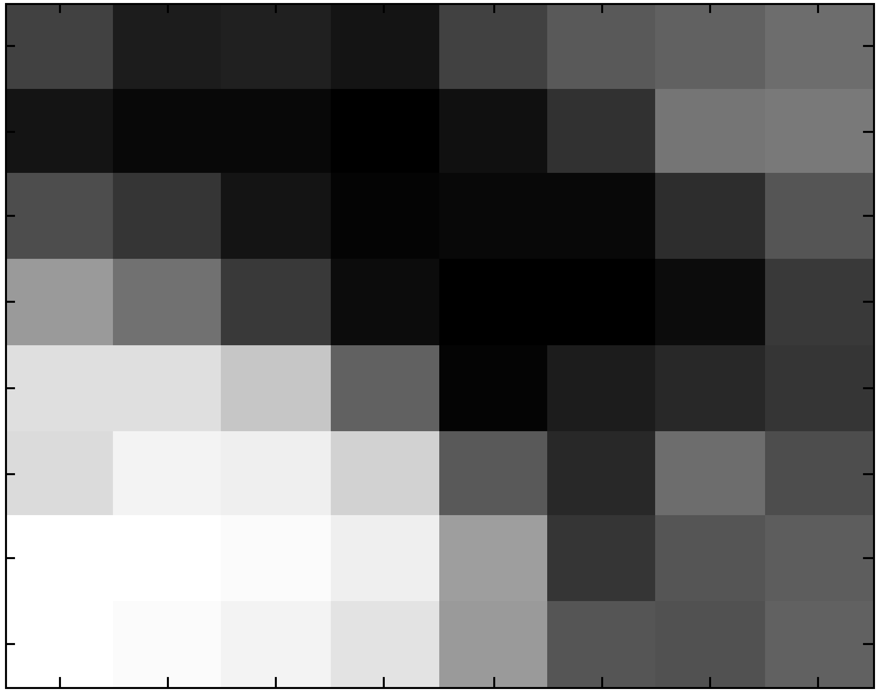
T (m, n)

Figure 15.14 **Case 3: gradient grayscale**

Image compression – JPEG encoding

- Gray-Scale Example: Value Range 0 (black) --- 255 (white)

63	33	36	28	63	81	86	98
27	18	17	11	22	48	104	108
72	52	28	15	17	16	47	77
132	100	56	19	10	9	21	55
187	186	166	88	13	34	43	51
184	203	199	177	82	44	97	73
211	214	208	198	134	52	78	83
211	210	203	191	133	79	74	86



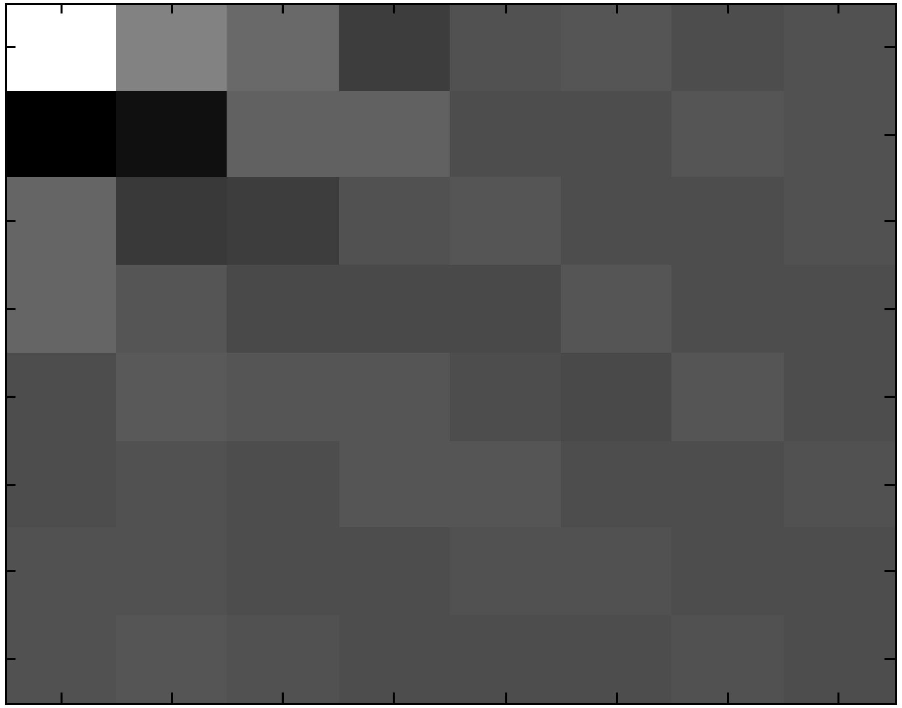
X

Image compression – JPEG encoding

- 2D-DCT of matrix

Numbers are coefficients of polynomial

-304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1

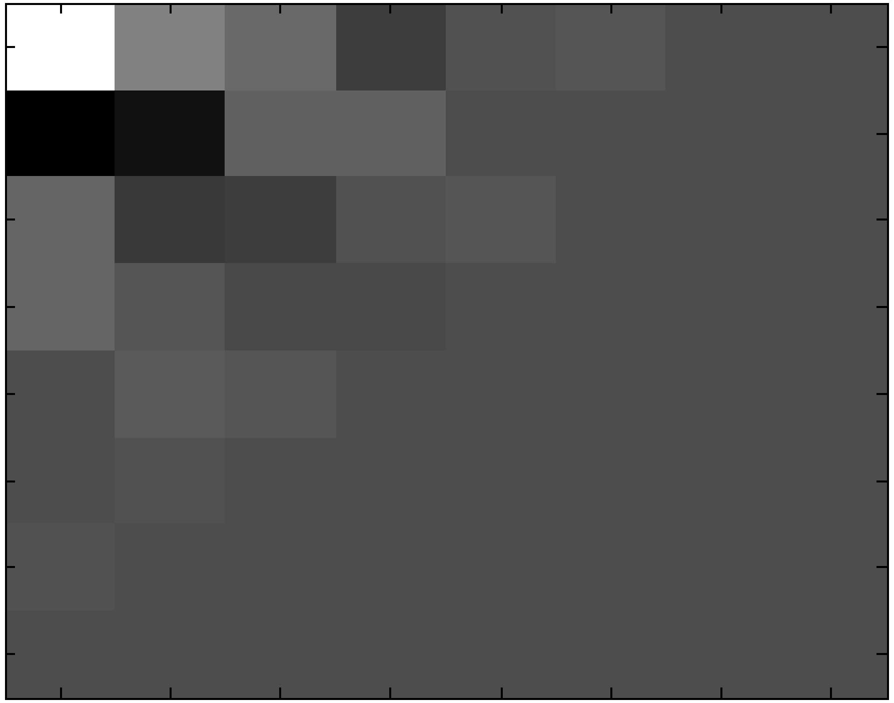


Y

Image compression – JPEG encoding

- Cut the least significant components

-304	210	104	-69	10	20	-12	0
-327	-260	67	70	-10	-15	0	0
93	-84	-66	16	24	0	0	0
89	33	-19	-20	0	0	0	0
-9	42	18	0	0	0	0	0
-5	15	0	0	0	0	0	0
10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



As you can see, we save a little over half the original memory.

Inverse 2D-DCT

2D-DCT gives us $Y = C(CX^T)^T$ which can be rewritten

$$Y = CXC^T$$

Since C is an orthogonal we can solve for X using the fact

$C^{-1} = C^T$

Therefore, $X = C^TYC$

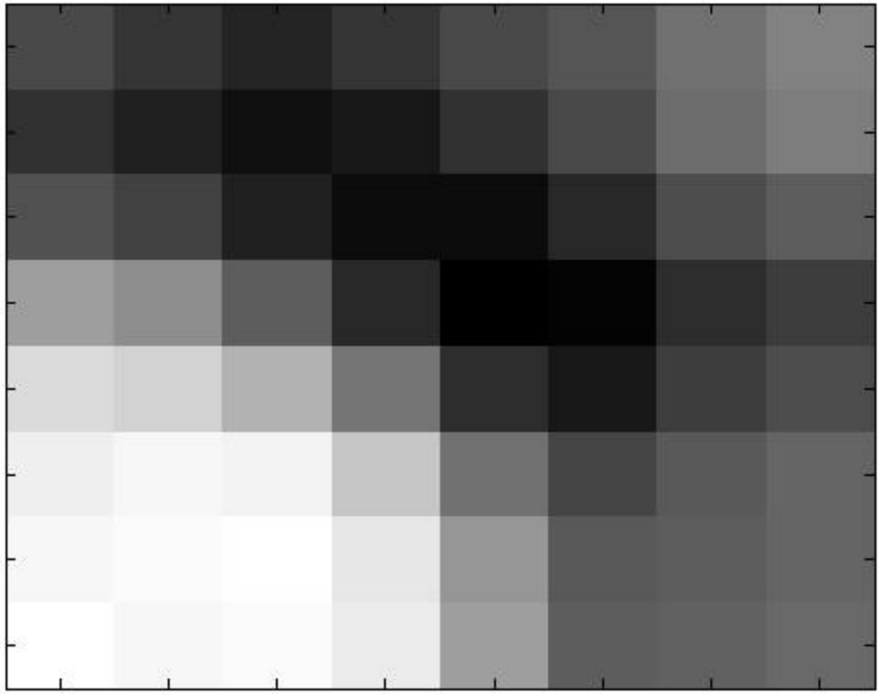
Reconstructing the Image

- In Mathematical terms:
  - Let $X = (x_{ij})$ be a matrix of n^2 real numbers
  - $Y = (y_{kl})$ be the 2D-DCT of X
  - $a_0 = 1/sqrt(2) \quad and \quad a_k = 1 \quad for\, k > 0$
  - Then: $$P_n(s,t) = \frac{2}{n} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} y_{kl} a_k a_l \cos \frac{k(2s+1)\pi}{2n} \cos \frac{l(2j+1)\pi}{2n}$$
  - Satisfies $P_n(i,j) = x_{ij} \;\; for\, i,j = 0, \ldots, n-1$

Reconstructing the Image

- New Matrix and Compressed Image

55	41	27	39	56	69	92	106
35	22	7	16	35	59	88	101
65	49	21	5	6	28	62	73
130	114	75	28	-7	-1	33	46
180	175	148	95	33	16	45	59
200	206	203	165	92	55	71	82
205	207	214	193	121	70	75	83
214	205	209	196	129	75	78	85



Can You Tell the Difference?

Original

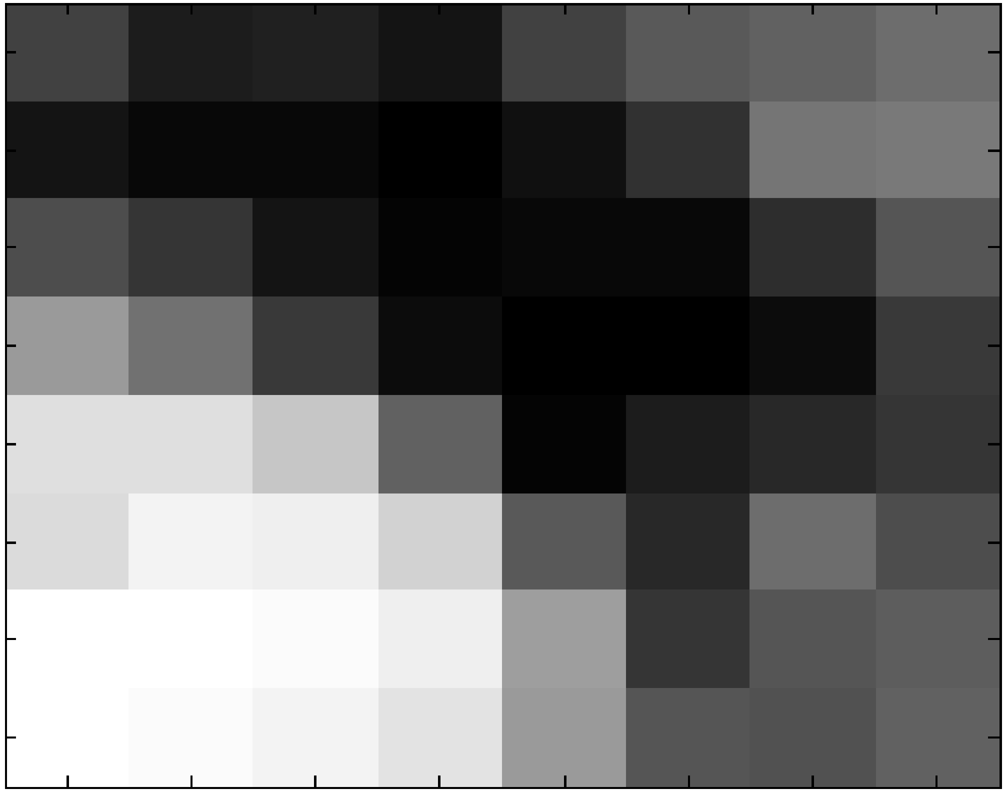
63	33	36	28	63	81	86	98
27	18	17	11	22	48	104	108
72	52	28	15	17	16	47	77
132	100	56	19	10	9	21	55
187	186	166	88	13	34	43	51
184	203	199	177	82	44	97	73
211	214	208	198	134	52	78	83
211	210	203	191	133	79	74	86

Compressed

55	41	27	39	56	69	92	106
35	22	7	16	35	59	88	101
65	49	21	5	6	28	62	73
130	114	75	28	-7	-1	33	46
180	175	148	95	33	16	45	59
200	206	203	165	92	55	71	82
205	207	214	193	121	70	75	83
214	205	209	196	129	75	78	85

Can You Tell the Difference?

Original



Compressed

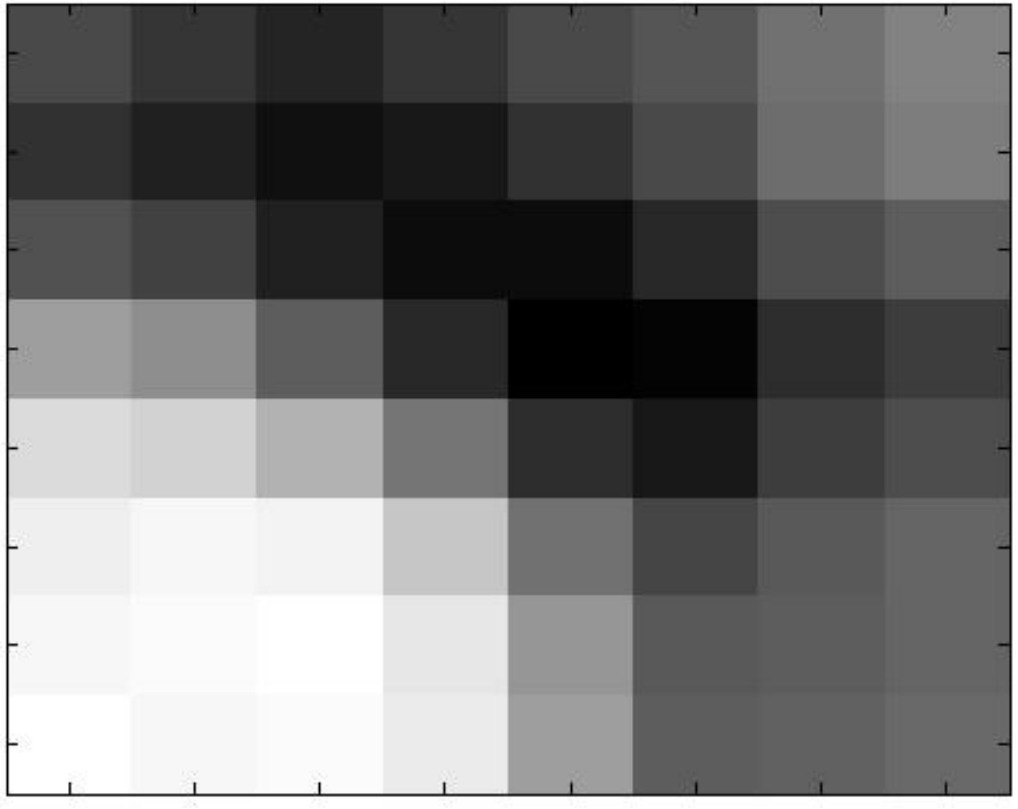


Image Compression

Original

Compressed

Quantization

Lượng tử hóa

Quantization

- After the T table is created, the values are quantized to reduce the number of bits needed for encoding.
- Quantization divides the number of bits by a constant and then drops the fraction. This reduces the required number of bits even more.
- In most implementations, a quantizing table (8 by 8) defines how to quantize each value. The divisor depends on the position of the value in the T table. This is done to optimize the number of bits and the number of 0s for each particular application.

More about Quantization

- **Quantization is the main source for loss**
  - $Q(u, v)$ of larger values towards lower right corner
    - More loss at the higher spatial frequencies
    - Supported by Observations 1 and 2.
  - Q(u,v) obtained from psychophysical studies
    - maximizing the compression ratio while minimizing perceptual losses

Linear Quantization

- We will not zero the bottom half of the matrix.

- The idea is to assign fewer bits of memory to store information in the lower right corner of the DCT matrix.

Linear Quantization

Use Quantization Matrix (Q)

$$q_{kl} = 8p(k + l + 1) \quad \text{for } 0 \leq k, l \leq 7$$

Q = p *

8	16	24	32	40	48	56	64
16	24	32	40	48	56	64	72
24	32	40	48	56	64	72	80
32	40	48	56	64	72	80	88
40	48	56	64	72	80	88	96
48	56	64	72	80	88	96	104
56	64	72	80	88	95	104	112
64	72	80	88	96	104	112	120

Linear Quantization

- p is called the loss parameter
- It acts like a “knob” to control compression
- The greater p is the more you compress the image

Linear Quantization

We divide the each entry in the DCT matrix by the Quantization Matrix

-304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1

8	16	24	32	40	48	56	64
16	24	32	40	48	56	64	72
24	32	40	48	56	64	72	80
32	40	48	56	64	72	80	88
40	48	56	64	72	80	88	96
48	56	64	72	80	88	96	104
56	64	72	80	88	95	104	112
64	72	80	88	96	104	112	120

Linear Quantization

p = 1

-38	13	4	-2	0	0	0	0
-20	-11	2	2	0	0	0	0
4	-3	-2	0	0	0	0	0
3	1	0	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

New Y: 14 terms

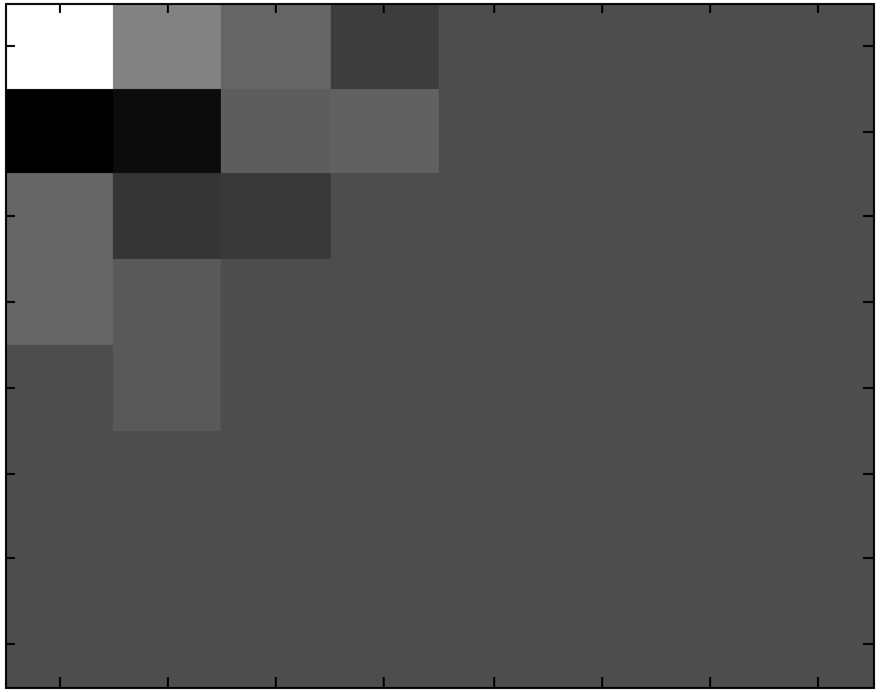
p = 4

-9	3	1	-1	0	0	0	0
-5	-3	1	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
0	0	0	0	0	0	0	0

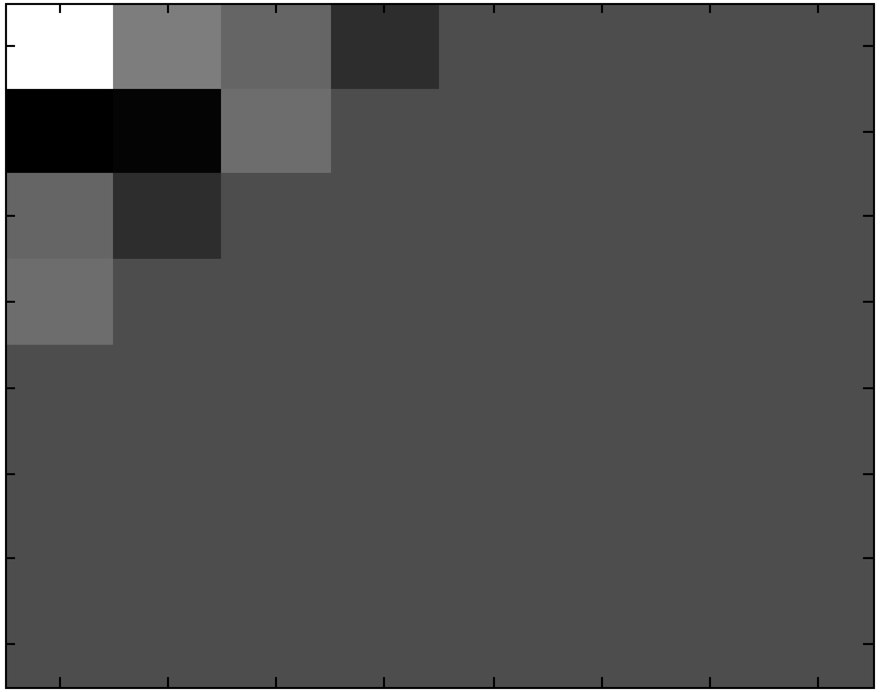
New Y: 10 terms

Linear Quantization

p = 1

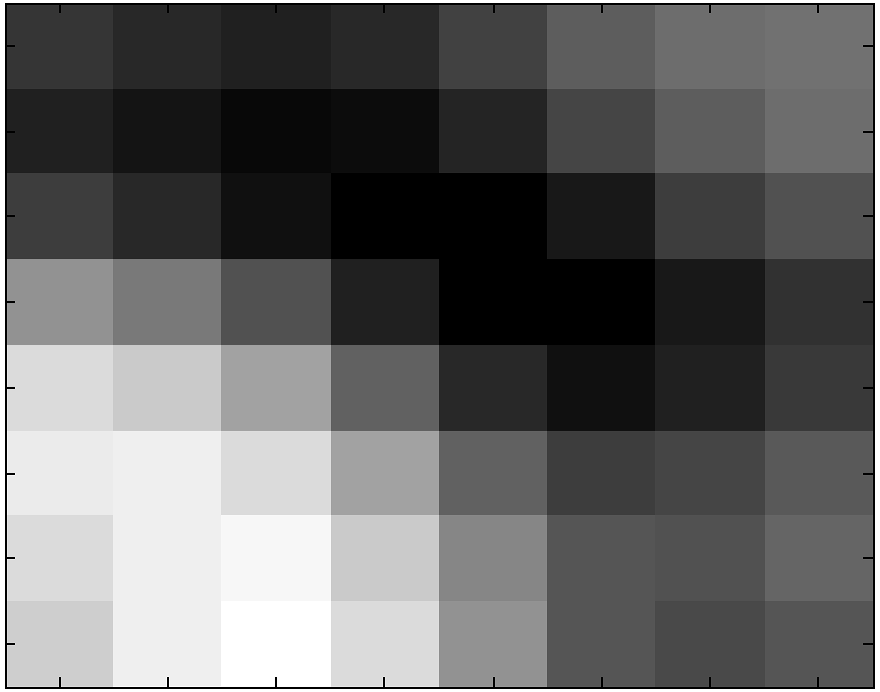


p = 4

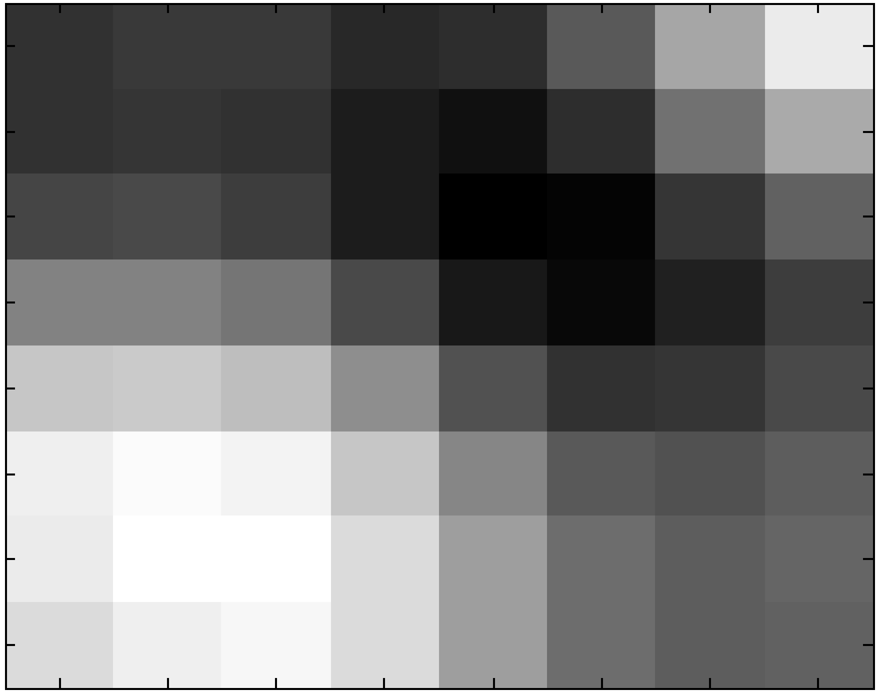


Linear Quantization

p = 1



p = 4



Linear Quantization

p = 1

p = 4

Memory Storage

- The original image uses one byte (8 bits) for each pixel. Therefore, the amount of memory needed for each 8 x 8 block is:
  - $8 \times (8^2) = 512$ bits

Is This Worth the Work?

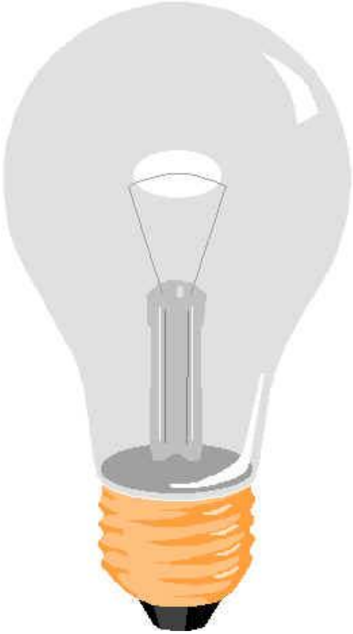
- The question that arises is “How much memory does this save?”

Linear Quantization

p	Total bits	Bits/pixel
X	512	8
1	249	3.89
2	191	2.98
3	147	2.30

JPEG Quantization

<u>Luminance:</u>



$Q_v =$

p

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

JPEG Quantization

- Quantization Example:

$$X=\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & 7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & -44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix}$$

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

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

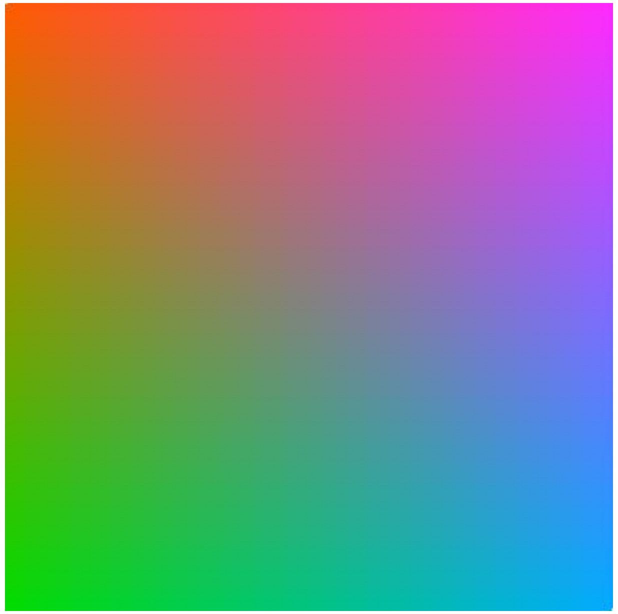
$X_Q = \mathrm{round}(X_{n,m}/Q_{n,m})$

$$\mathrm{round}\left(\frac{-415}{16}\right) = \mathrm{round}\left(-25.9375\right) = -26$$

Time complexity: $O(n^2)$ where n=#columns and rows (n^2 O(1) operations)

JPEG Quantization

<u>Chrominance:</u>



$$Q_C =$$

$$\begin{array}{cccccccc}
\{\,17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\
18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\
24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\
47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99\,\}
\end{array}$$

Luminance and Chrominance

- Human eye is more sensible to luminance
  (Y coordinate).

- It is less sensible to color changes
  (UV coordinates).

- Then: compress more on UV !

- Consequence: color images are more compressible than grayscale ones

Reconstitution

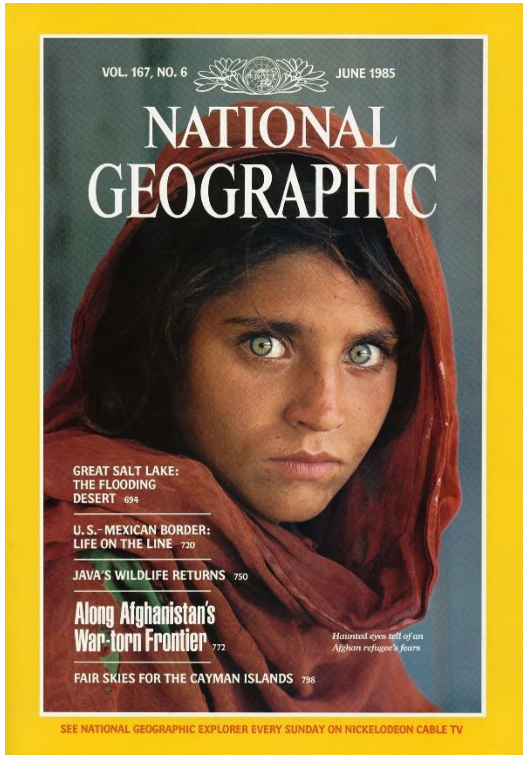
- After compression, Y, U, and V, are recombined and converted back to RGB to form the compressed color image:

B= U+Y

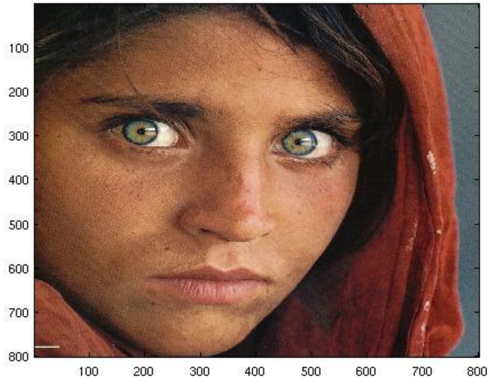
R= V+Y

G= (Y- 0.299R - 0.114B) / 0.587

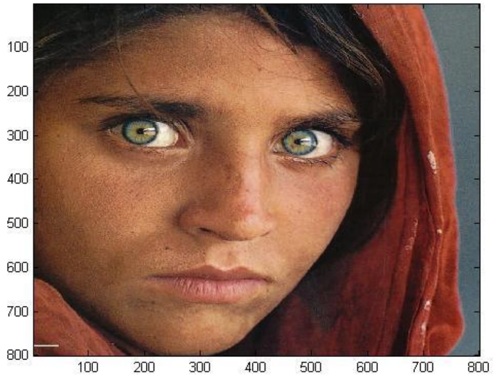
Comparing Compression



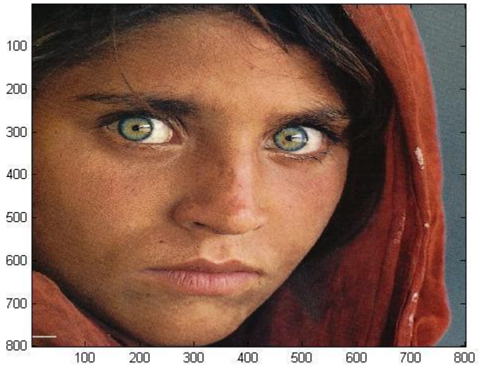
Original



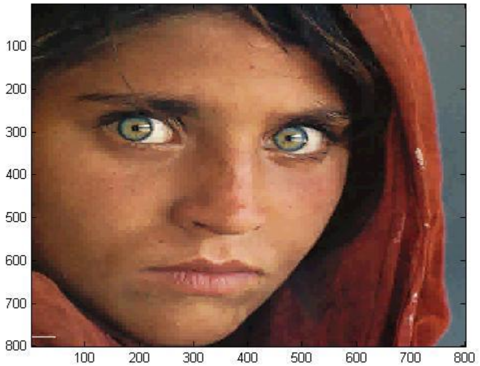
p = 1



p = 4



p = 8



Up Close

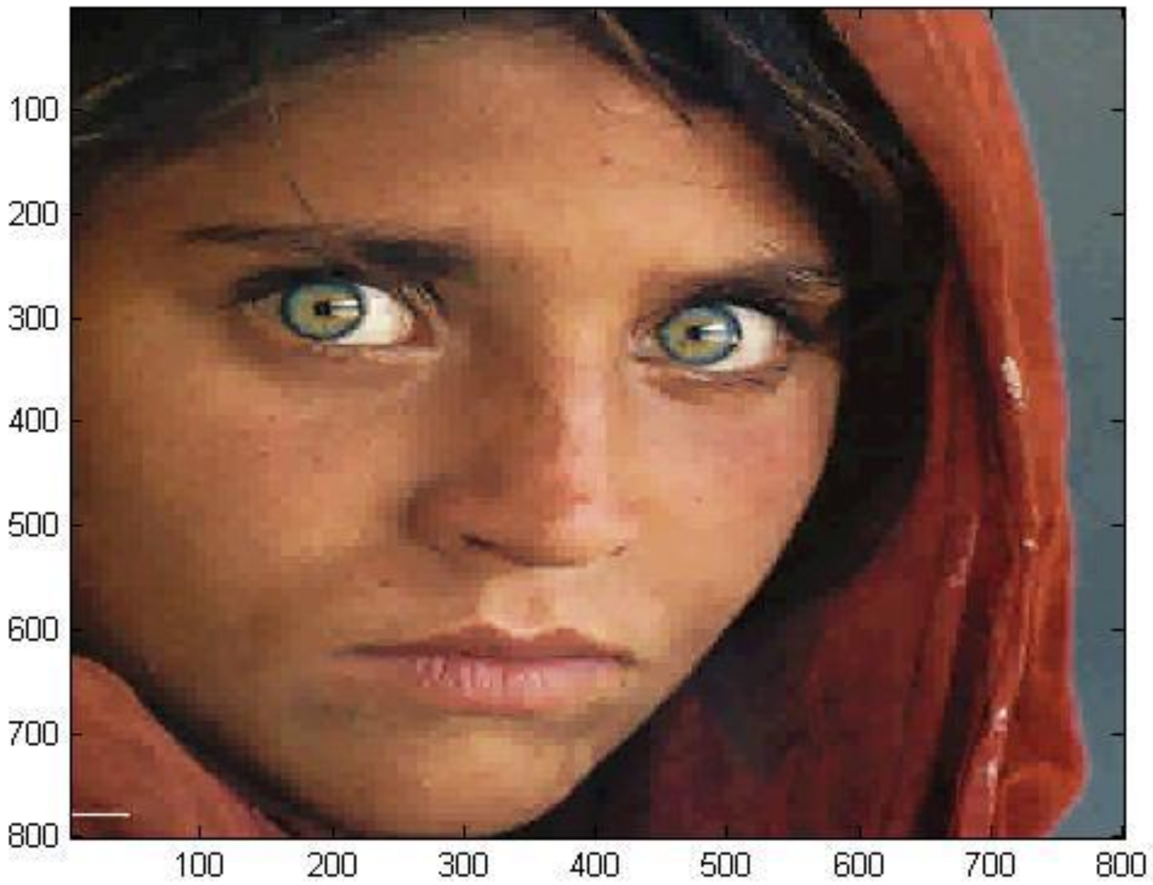


Image compression – JPEG encoding

Compression

- After quantization the values are read from the table, and redundant 0s are removed. However, to cluster the 0s together, the process reads the table diagonally in a zigzag fashion rather than row by row or column by column. The reason is that if the picture does not have fine changes, the bottom right corner of the T table is all 0s.

- JPEG usually uses run-length encoding at the compression phase to compress the bit pattern resulting from the zigzag linearization.

Image compression – JPEG encoding

Compression

- Final Steps: Zig-Zag Ordering
  - Maps an 8x8 block into a 1 x 64 vector
  - Zigzag pattern group low frequency coefficients in top of vector.

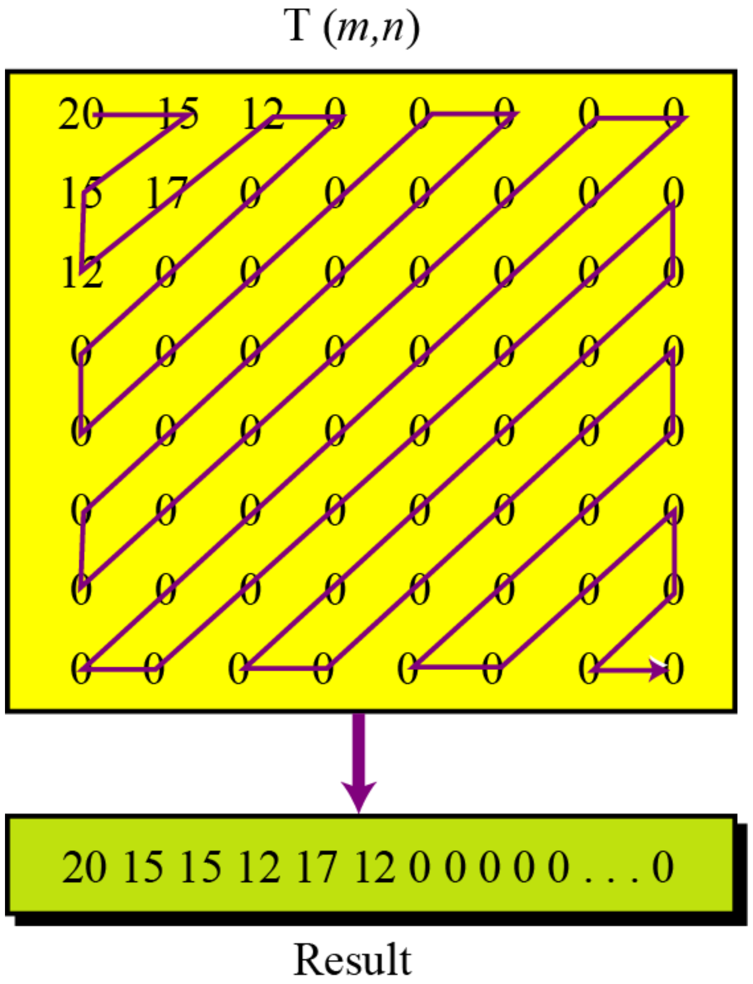


Figure 15.15 **Reading the table**

Image compression – JPEG encoding

Compression

- Final Steps: Zig-Zag Ordering
  - Reorder quantized matrix to put non-zero elements in a sortable sequence

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

-26
-3 0
-3 -3 -6
2 4 1 -4
1 1 5 1 2
-1 1 -1 2 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

→ **[-26, -3 0, -3, …, -1, 0, 0]**

--Do not store zeroes after final element in zigzag row with a non-zero element

Runlength Encoding (RLE)

A typical 8x8 block of quantized DCT coefficients.
Most of the higher order coefficients have been quantized to 0.

12	34	0	54	0	0	0	0
87	0	0	12	3	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Zig-zag scan: the sequence of DCT coefficients to be transmitted:
12 34 87 16 0 0 54 0 0 0 0 0 0 12 0 0 3 0 0 0 .....
DC coefficient (12) is sent via a separate Huffman table.
Runlength coding remaining coefficients:
34 | 87 | 16 | 0 0 54 | 0 0 0 0 0 0 12 | 0 0 3 | 0 0 0 .....

Bài tập

- Cho một ảnh I như sau:

2	5	5	2
3	1	3	9
2	8	0	5
7	2	0	7

- Cho biết kết quả khi áp dụng phép biến đổi DCT:
- a. kích thước 4x4
- b. Kích thước 2x2