Image Processing

INT3404 20

Week 11:

Compression

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.

Email: ngocdiep@vnu.edu.vn

Slide & code: https://github.com/chupibk/INT3404_20

1

Schedule

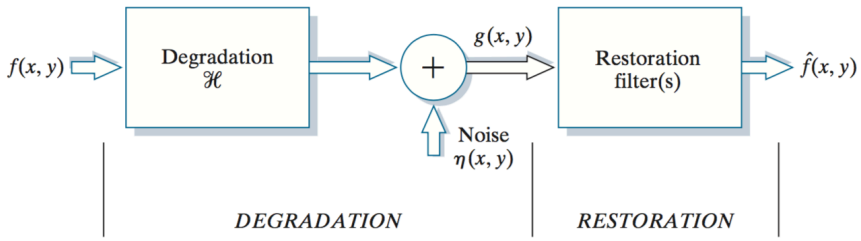
Week	Content	Homework
1	Introduction	Set up environments: Python 3, OpenCV 3, Numpy, Jupyter Notebook
2	Digital image – Point operations Contrast adjust – Combining images	HW1: adjust gamma to find the best contrast
3	Histogram - Histogram equalization – Histogram-based image classification	Self-study
4	Spatial filtering - Template matching	Self-study
5	Feature extraction Edge, Line, and Texture	Self-study
6	Morphological operations	HW2: Barcode detection → Require submission as mid-term test
7	Filtering in the Frequency domain Announcement of Final project topics	Final project registration
8	Color image processing	HW3: Conversion between color spaces, color image segmentation
9	Geometric transformations	Self-study
10	Noise and restoration	Self-study
11	Compression	Self-study
12	Final project presentation	Self-study
13	Final project presentation Class summarization	Self-study

2

2

Image degradation/restoration process

Linear, position-invariant



$$g(x,y) = (h \star f)(x,y) + \eta(x,y)$$

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

Noise comes during acquisition and/or transmission (e.g., environmental factors, quality of sensing elements, ...)

3

Recall week 10: Modeling and restoration

$$g(x,y) = (h \star f)(x,y) + \eta(x,y)$$

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

Degradation estimation Noise modeling

- Restoration:
  - Spatial: Filters (mean, median and theirs variances)
  - Frequency:
    - Notch filter
    - Inverse filter
    - Wiener filter

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

4

Wiener filter

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

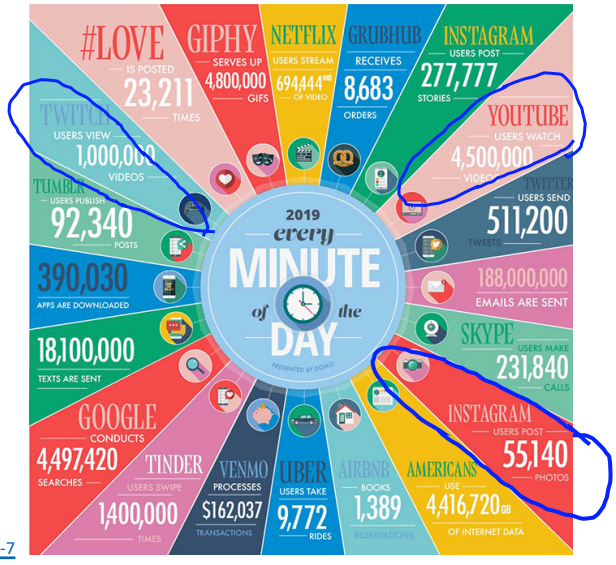
$$\hat{F} = S\,G$$

$$= \left[\frac{1}{H(u,v)}\,\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right] G(u,v)$$

5

Why image compression?

Data amount generated every minute



Efficient Store, Transmit and Display

Source: https://www.domo.com/learn/data-never-sleeps-7

6

Quiz: Data measurement unit

Bit MB PB ZB GB YB KB TB EB Byte BB

7

Data measurement unit

1 Bit = Binary Digit
8 Bits = 1 Byte
1024 Bytes = 1 Kilobyte
1024 Kilobytes = 1 Megabyte
1024 Megabytes = 1 Gigabyte
1024 Gigabytes = 1 Terabyte
1024 Terabytes = 1 Petabyte
1024 Petabytes = 1 Exabyte
1024 Exabytes = 1 Zettabyte
1024 Zettabytes = 1 Yottabyte
1024Yottabytes = 1 Brontobyte
1024 Brontobytes = 1 Geopbyte
1024 Geopbyte=1 Saganbyte
1024 Saganbyte=1 Pijabyte
Alphabyte = 1024 Pijabyte
Kryatbyte = 1024 Alphabyte
Amosbyte = 1024 Kryatbyte
Pectrolbyte = 1024 Amosbyte
Bolgerbyte = 1024 Pectrolbyte
Sambobyte = 1024 Bolgerbyte
Quesabyte = 1024 Sambobyte
Kinsabyte = 1024 Quesabyte
Rutherbyte = 1024 Kinsabyte
Dubnibyte = 1024 Rutherbyte
Seaborgbyte = 1024 Dubnibyte
Bohrbyte = 1024 Seaborgbyte
Hassiubyte = 1024 Bohrbyte
Meitnerbyte = 1024 Hassiubyte
Darmstadbyte = 1024 Meitnerbyte
Roentbyte = 1024 Darmstadbyte
Coperbyte = 1024 Roentbyte

8

Storage space of pictures

1000x1000

24 bits/px ==> 3MB/image

Scanned at 300px/inch, 1 bit/px
25,000 pages × 1,000,000 bits/page = 25 gigabytes

90 min, 640x480, 24bits/px, 24 frames/s
==> 90 ×60 ×24 ×640 ×480 ×3=120 gigabytes.

9

Image compression goal

- Store image data as efficiently as possible
- Ideally, want to
  - Maximize image quality
  - Minimize storage space and processing resources

} Can't have best of both worlds
What are some good compromises

- Applications: HDTV, film, remote sensing and satellite image transmission, network communication, image storage, medical image processing, fax.

10

Why is it possible to compress images?

- Data != information/knowledge
- Data >> information
- Key idea in compression: only keep the info

11

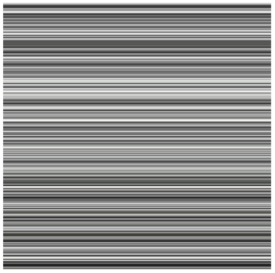
Why data != info?

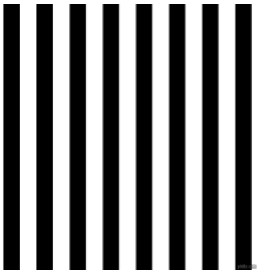
- Answer: redundancy
- Statistical redundancy
  - Spatial redundancy and coding redundancy
- Psychovisual redundancy
  - Greyscale redundancy and frequency redundancy

12

Spatial redundancy

- Pixel values are not spatially independent
- High correlation among neighbor pixels





13

Coding redundancy

- Redundancy when mapping from the pixels (symbols) to the final compressed binary code (information theory)
- Example

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k = 87, 128, 186, 255$	0	—	8	—	0

$$L_{\text{avg}} = 0.25(2) + 0.47(1) + 0.03(3) = 1.81 \text{ bits}$$

14

Redundancy vs Compression ratio

- Data redundancy

$$R = 1 - \frac{1}{C} = 1 - \frac{1}{4.42} = 0.774$$

- Compression ratio

$$C = \frac{b}{b'} = \frac{8}{1.81} \approx 4.42$$

b, b': number of bits (or information carrying units)

15

Psychovisual redundancy

- Human eye does not respond to all visual information with equal sensitivity
- Some data is simply of less relative importance → can be eliminated without introducing any significant difference to the human eye
- Example:
  - color
  - intensity

16

Frequency redundancy

- Human eye functions as a lowpass filter
  - High frequencies in an image can be “ignored” without the HVS noticing
  - Key issue in lossy image compression

17

How Compression works

- Compression is done by transforming and removing image data redundancies
- Mathematically this means transforming data to a statistically uncorrelated set

18

Two major types of Compression algorithms

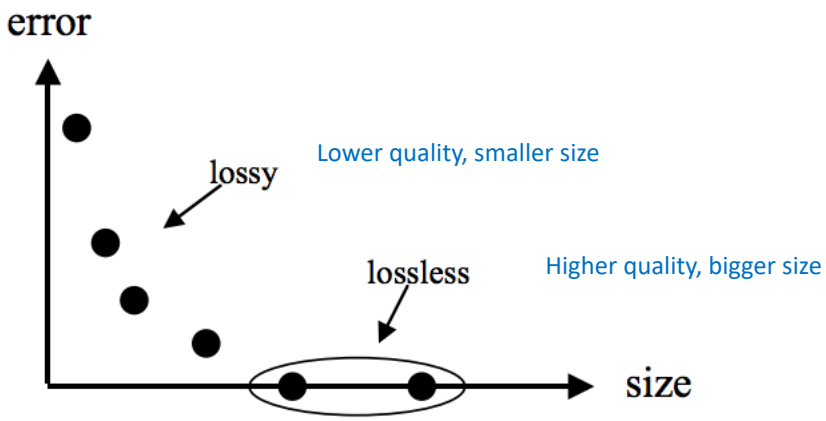
- Lossless compression
  - The original data is reconstructed perfectly
  - Theoretical limit on maximal compression
  - Practical compression ratios of < 10:1 (still images)
  - Uses entropy coding only (or none at all)
  - Examples: BMP, TIFF, GIF
- Lossy compression
  - Decomposition results in an approximation of the original image
  - Maximal compression rate is a function of reconstruction quality
  - Practical compression ratios of > 10:1 (still images)
  - Uses both quantization and entropy coding
  - Usually involves transform into frequency or other domain
  - Examples: JPEG, JPEG-2000

Details

Overall

19

Lossy vs lossless



Lower quality, smaller size

Higher quality, bigger size

20

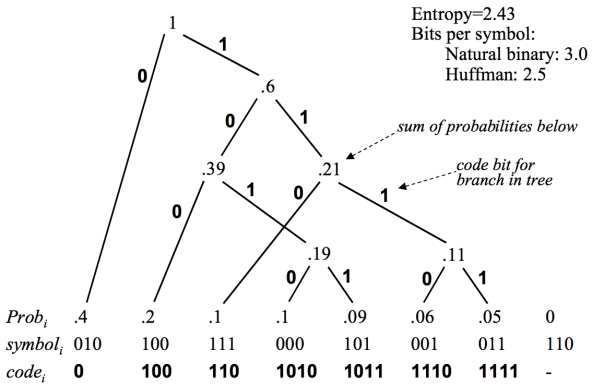
Common lossless compression algorithms

- Huffman
- Run length coding (RLE)
- Variable length coding
- Arithmetic coding
- LZW (Lempel-Ziv-Welch)

22

Huffman coding example

- These are prefix codes assigned to symbols depending on the probability of the symbols



23

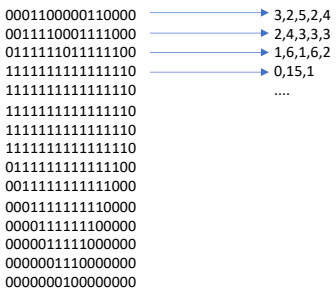
Run-length coding

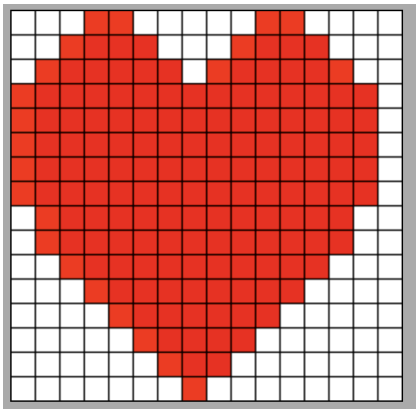
- Similar consecutives symbols are grouped and presented by a code
- Start from a specific code, following by a symbol and the number indicating the number of its copies
- It is a entropy coding technique used in H.264, CAVLC
- For example: aaaabbbddaaddccccccca
- Is is compressed to: 4a3b2d2a2d7c1a

24

BMP (bitmap): Mapping pixels to bits

- Only two colors: red & white -> mapping red pixels to 1 and white pixels to 0
- Using RLE:





Ref: https://www.khanacademy.org/computing/computers-and-the-internet/xcae6f4a7ff015e7d:digital-information/xcae6f4a7ff015e7d:data-compression/a/simple-image-compression

25

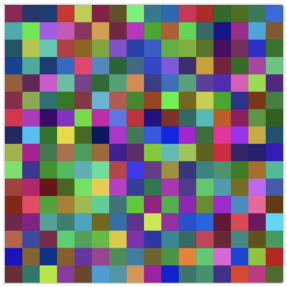
Bitmap compression ratio

Image	Dimensions	Uncompressed	After RLE	Space savings
	16x16	256	228	10.9%
	32x32	1024	532	48.0%
	128x128	16384	2898	82.3%

Image	Dimensions	Uncompressed	After RLE	Space savings
	128x128	16384	2898	82.3%
	128x128	16384	8298	49.4%
	128x128	16384	8730	46.7%

26

Limits of RLE



Different colors, no runs

27

GIF (Graphics interchange format)

- Can use up to 256 colors from 24-bit RGB color space
  - If source image contains more than 256 colors, need to reprocess image to fewer colors
- Suitable for simpler images such as logos and textual graphics, not so much for photographs
- Uses LZW lossless data compression

28

LZW Data compression

Compression algorithm

```
STRING = get input character
WHILE there are still input characters DO
    CHARACTER = get input character
    IF STRING+CHARACTER is in the string table then
        STRING = STRING+character
    ELSE
        output the code for STRING
        add STRING+CHARACTER to the string table
        STRING = CHARACTER
    END of IF
END of WHILE
output the code for STRING
```

Input String = /WED/WE/WEE/WEB/WET

Character Input	Code Output	New code value	New String
/W	/	256	/W
E	W	257	WE
D	E	258	ED
/	D	259	D/
WE	256	260	/WE
/	E	261	E/
WEE	260	262	/WEE
/W	261	263	E/W
EB	257	264	WEB
/	B	265	B/
WET	260	266	/WET
EOF	T		

Example: https://marknelson.us/posts/1989/10/01/lzw-data-compression.html

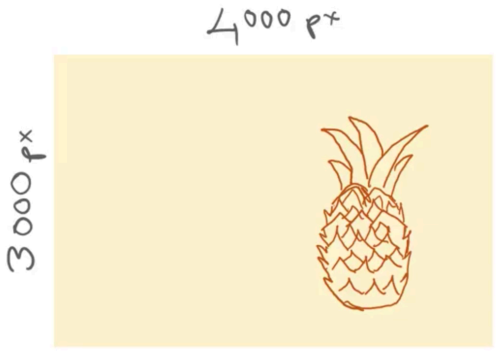
29

JPEG (Joint Photographic Experts Group)

- Most dominant image format today
- Typical file size is about 10% of that of BMP (can vary depending on quality settings)
- Unlike GIF, JPEG is suitable for photographs, not so much for logos and textual graphics

30

Image file size



3000 × 4000 × 3 = 36,000,000

36 MB

31

IMG_4586.jpg

JPEG image - 1.9 MB
Created Friday, May 3, 2019 at 4:06 PM
Modified Friday, May 3, 2019 at 4:06 PM
Last opened Saturday, August 17, 2019 at 10:40 PM
Dimensions 4032 × 3024
Add Tags...

Only 1.9MB!

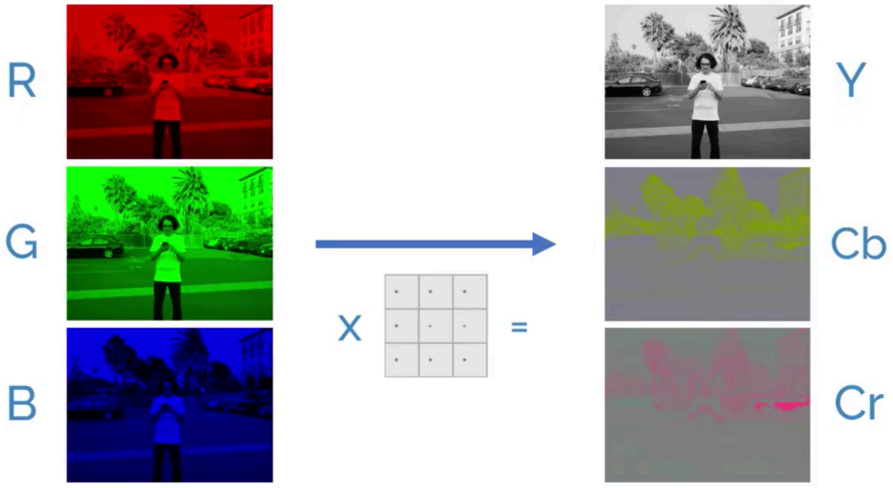
32

Lossy compression

The more information you discard, the worse the image quality gets

33

Color space conversion



34

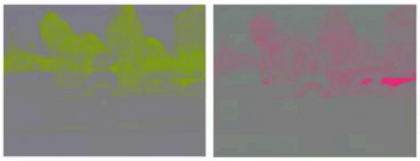
Luminance Chrominance

Our visual system is much more sensitive to the changes in brightness than color
→ We can safely downsample the chroma components to save some space

35

Chroma subsampling

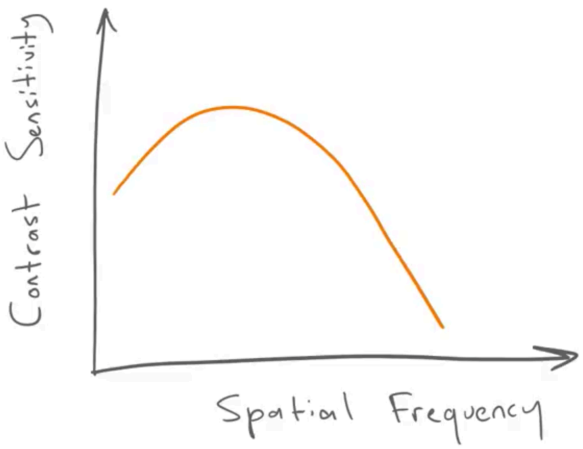
Luminance



Chrominance

36

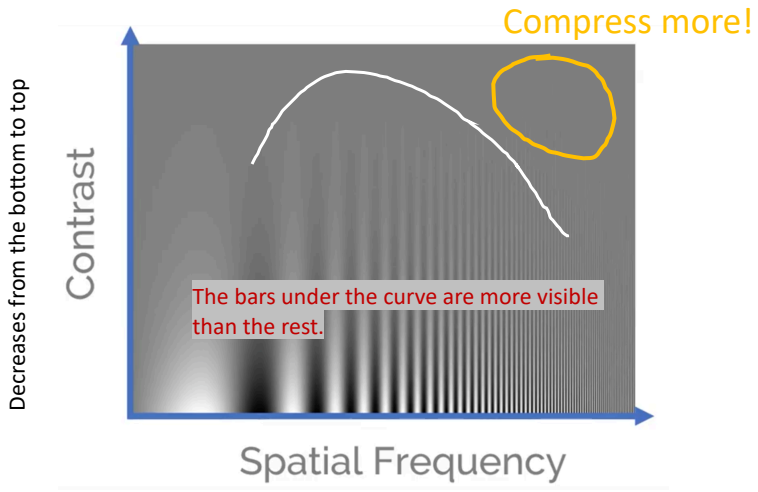
Frequency-dependent contrast sensitivity



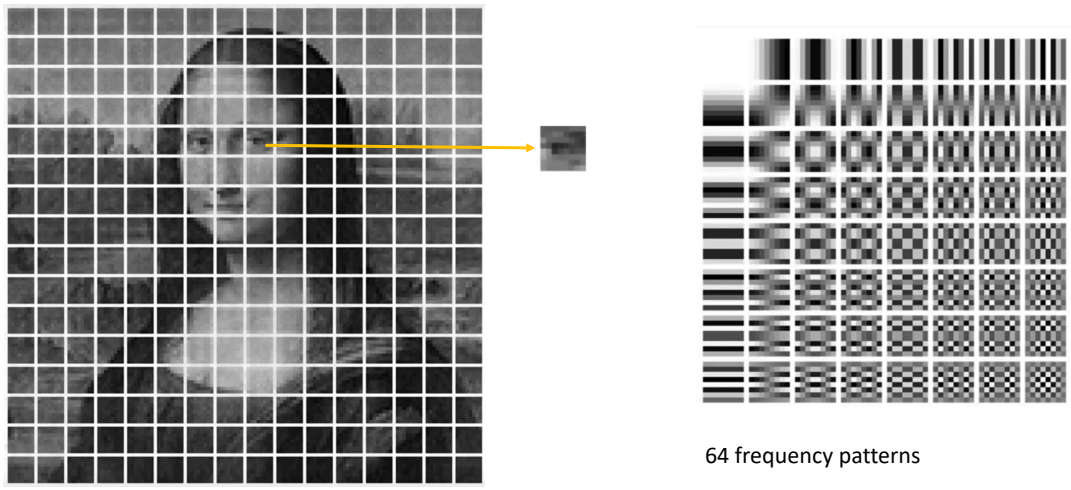
It's easier to miss small objects or fine details in a picture as compared to the large ones

37

Frequency vs contrast

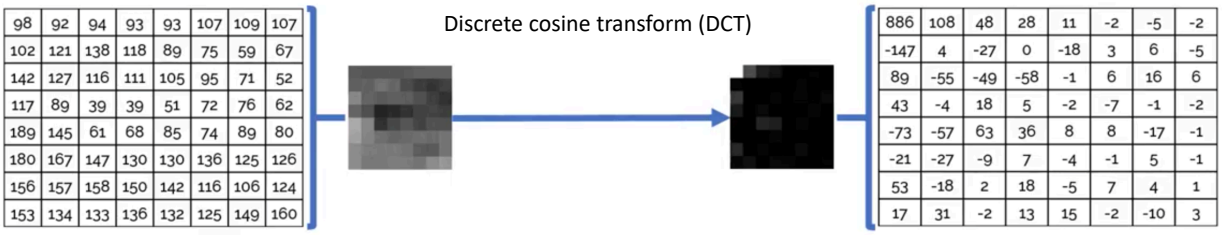


38



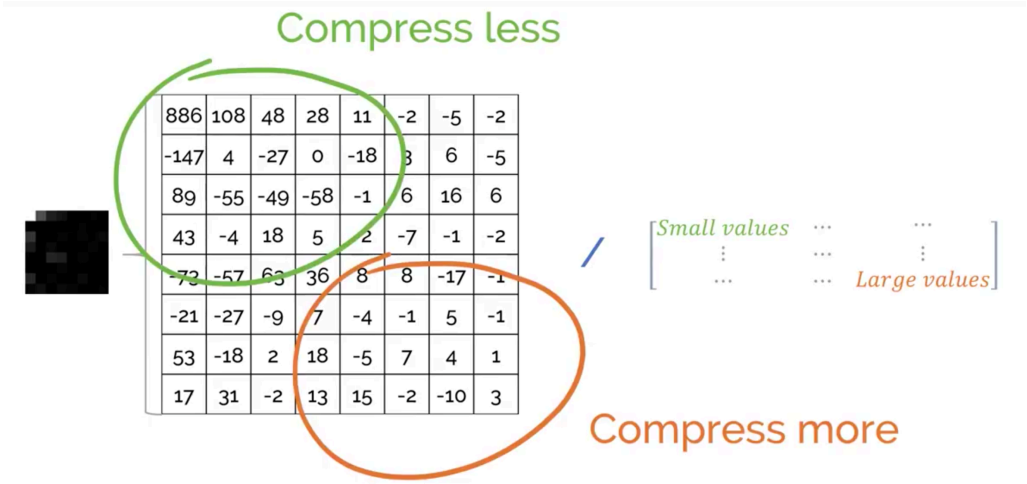
39

JPEG compression



40

Quantization



42

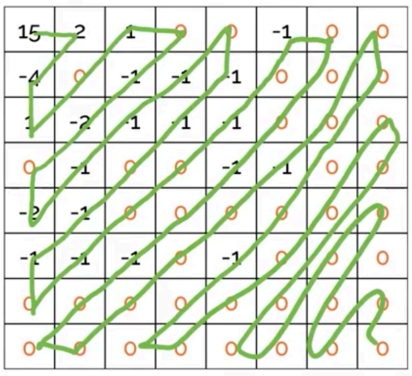
=

15	2	1	0	0	-1	0	0
-4	0	-1	-1	-1	0	0	0
1	-2	-1	-1	-1	0	0	0
0	-1	0	0	-1	-1	0	0
-2	-1	0	0	0	0	0	0
-1	-1	-1	0	-1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

43

Zig-zag arrangement

=



= [15, 2, -4, 1, 0, 1, …, 0, 0, 0, 0, 0, 0, 0, 0]

44

Lossless Encoding

= [15, 2, -4, 1, 0, 1, …, 0, 0, 0, 0, 0, 0, 0, 0]

repeat(0, 18)

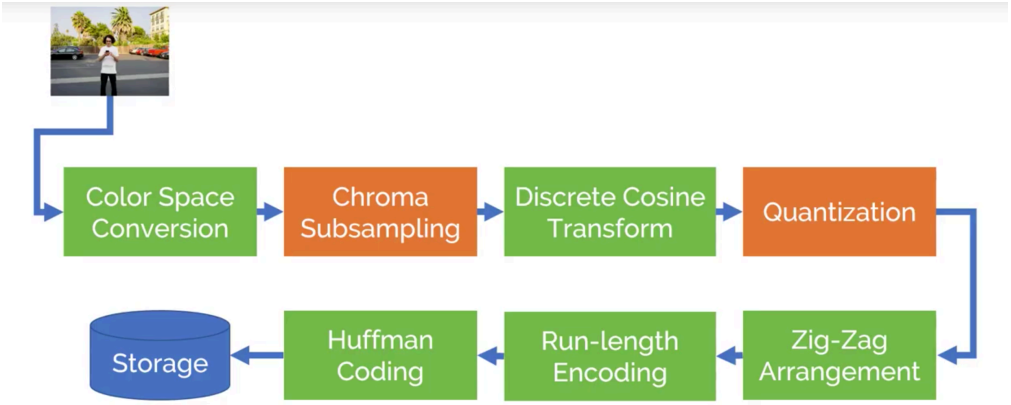
Run-length encoding

Huffman (entropy) encoding

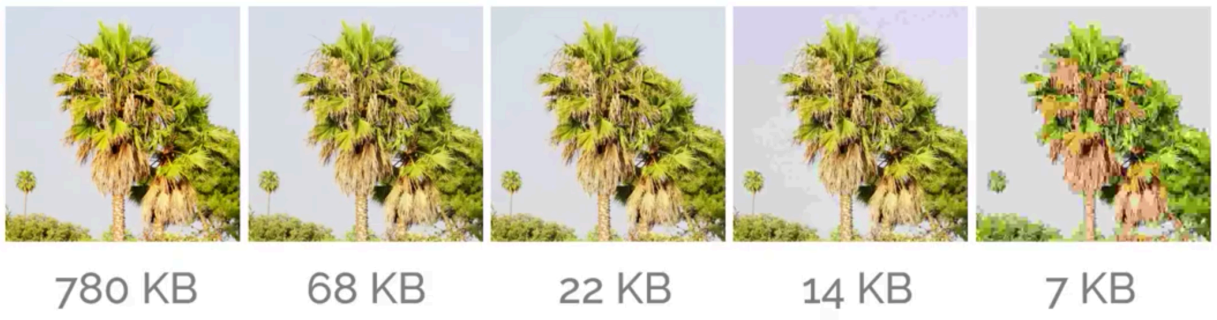
Encode the more frequent values with fewer bits and less frequent values with more bits

45

JPEG encoding

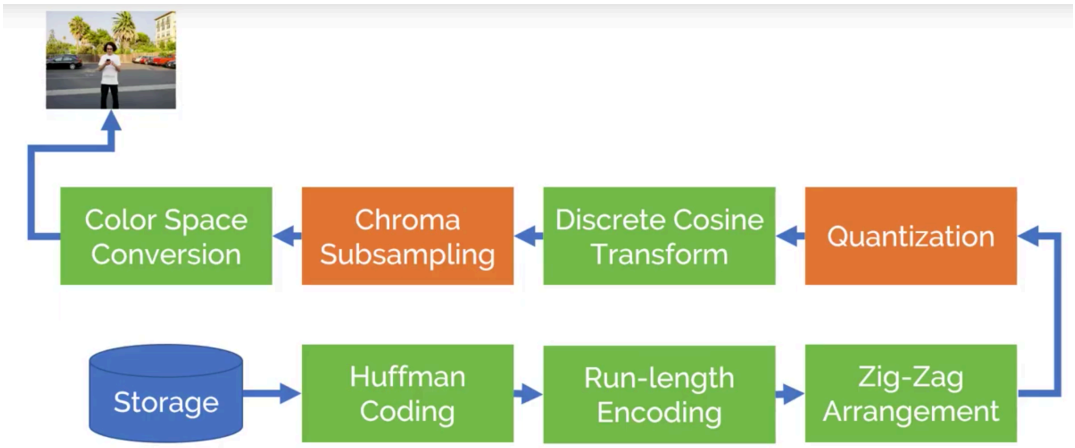


46



47

JPEG decoding



48

Popular compression standards

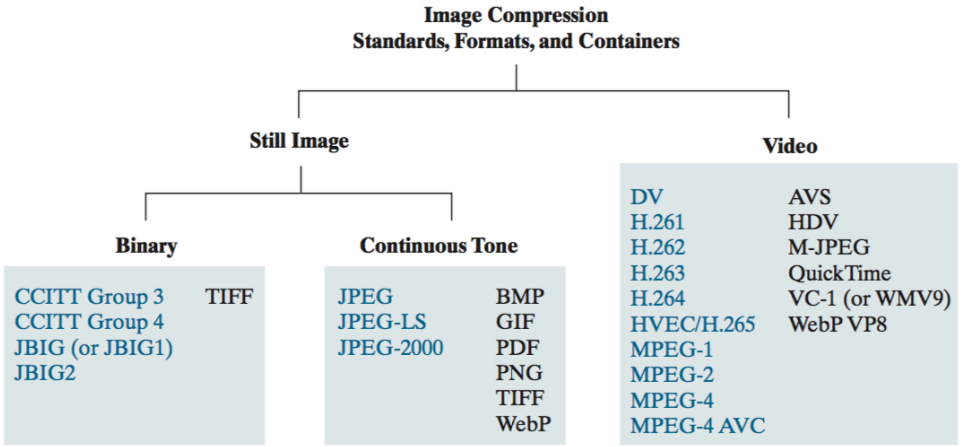


FIGURE 8.6 Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in blue; all others are in black.

49

References

- https://www.youtube.com/watch?v=Ba89cI9eIg8
- R. C. Gonzalez, R. E. Woods, "Digital Image Processing," 4th edition, Pearson, 2018.

50

Homeworks

1. Study how to apply LZW algorithm to image compression
2. Study some variants of JPEG algorithms
3. Study compression algorithms for videos (MPEG, H.265)

51