Chapter 1. Digital Data Representation and Communication

Part 2

Department of Computer Science
National Tsing Hua University

Compression

- Digital media files are usually very large, and they need to be made smaller—compressed
- Without compression
  - Won’t have storage capacity
  - Won’t be able to communicate them across networks without overly taxing the patience of the recipients
- Compression Rate
  - A file that is reduced by compression to half its original size **50% compression** or **compression rate is 2:1**
- Compression algorithms can be divided into two basic types: ***lossless compression*** and ***lossy compression***.

Data Compression

Raw image takes about 6M bytes
(without header information)

24k bytes with jpeg, Q=50

Types of compression

- Lossless compression
  - No information is lost between the compression and decompression steps
  - Reduces the file size to fewer bits
  - Method:
    - ✓Run-Length encoding (RLE)
    - ✓Entropy encoding
    - ✓Arithmetic encoding

Types of compression

- Lossy compression
  - Sacrifice some information which is not important to human perception
    - ✓ In image files: subtle changes in color that the eye cannot detect
    - ✓ In sound files: changes in frequencies that are imperceptible to the human ear
  - Method:
    - ✓ Transform encoding

Types of compression

- Dictionary-based methods (e.g. LZW compression) use a look-up table of fixed-length codes.
- Entropy compression uses a statistical analysis of the frequency of symbols and achieves compression by encoding more frequently-occurring symbols with shorter code words.
- Arithmetic encoding is also based on statistical analysis, but encodes an entire file in a single code word rather than creating a separate code for each symbol.
- Adaptive methods gain information about the nature of the file in the process of compressing it, and adapt the encoding to reflect what has been learned at each step.
- Differential encoding is a form of lossless compression that reduces file size by recording the difference between neighboring values rather than recording the values themselves.

Run-Length encoding (RLE)

- RLE is a simple example of lossless compression, being used in image compression.
- For example, BMP file optionally use RLE
- Instead of storing value of each pixel, to store number pairs (c,n).
  - c: grayscale value
  - n: how many consecutive pixels have that value

Run-Length encoding (RLE)

- Example 1
  - Data sequence:

    255 255 255 255 255 255 242 242 242 242 238 238 238 238 238 238 255 255 255 255
  - Encoding result:

    (255, 6), (242, 4), (238, 6), (255, 4)
- Example 2

  If one byte is used to represent each *n*, then the largest value for *n* is 255. If 1000 consecutive whites exist in the file.
  - Encoding result:

    (255, 255), (255, 255), (255, 255), (255, 235)

Choose the size of *n* is important

Run-Length encoding (RLE)

- Example 3
  - Data sequence:

    255 254 253 252 251 250

  - Encoding result:

    (255, 1), (244, 1), (253, 1), (252, 1), (251,1), (250,1)

  - RLE is not suitable for those sequence which has few repetition.
- *Any lossless compression algorithm and for any length file, there will always exist at least one case where the algorithm does not reduce the size of an input file of that length.*

Entropy encoding

- Using fewer bits to encode symbols that occur more frequently, while using more bits for symbols that occur infrequently
- Shannon's equation, below, gives us a way a judging whether our choice of number of bits for different symbols is close to optimal.
  - $H(S) = \eta = \sum_i p_i log_2(\frac{1}{p_i})$
  - S be a string of symbols and p_i be the frequency of the i^{th} symbol in the string.

Entropy encoding

- Applying ***Shannon's entropy equation***, you can determine an optimum value for the average number of bits needed to represent each symbol-instance in a string of symbols, based on how frequently each symbol appears
- Shannon proves that you can't do better than this optimum

Example of entropy encoding

- Example 1

  An image file that has 256 pixels, each pixel of a different color. Then the frequency of each color is 1/256. Thus, Shannon's equation reduces to

$$\sum_{0}^{255} \frac{1}{256}(log_2(\frac{1}{\frac{1}{256}})) = \sum_{0}^{255} \frac{1}{256}(log_2(256)) = \sum_{0}^{255} \frac{1}{256}(8) = 8$$

  This means that the average number of bits needed to encode each color is eight, which makes sense in light of the fact that $\log_2 256 = 8$

Entropy encoding

- Example 2

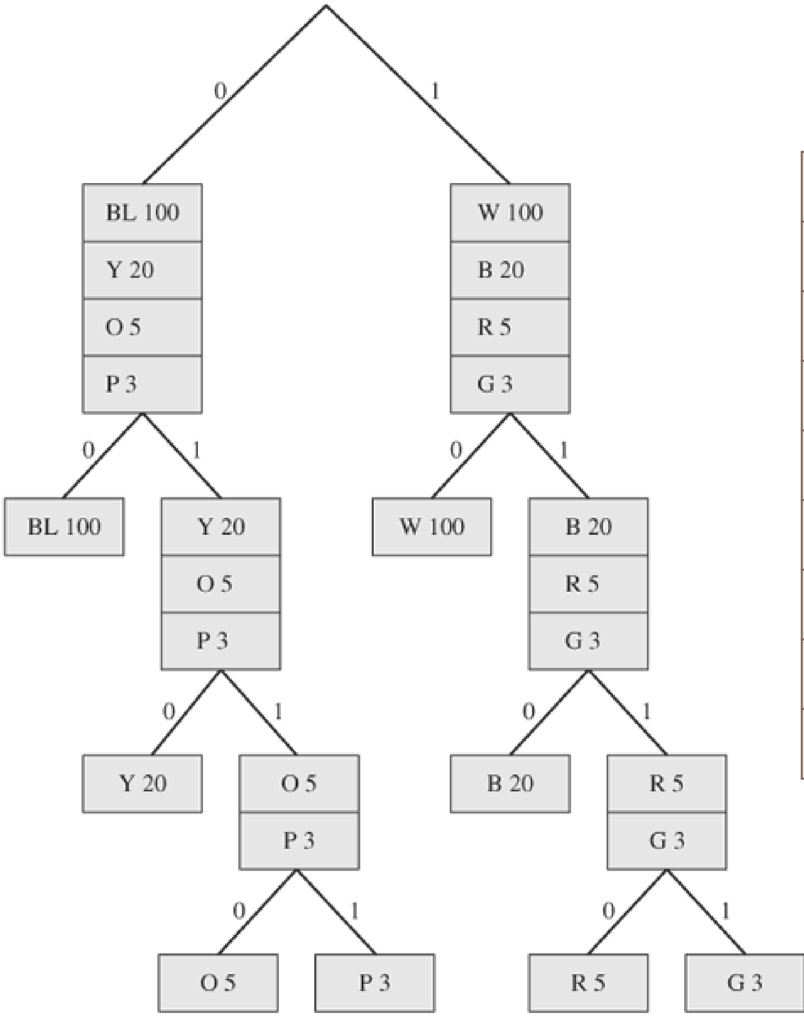
  An image file of 256 pixels and only eight colors

$$H(S) = \eta = \sum_i \underline{p_i} \underline{log_2(\frac{1}{p_i})}$$

Color	Frequency	Optimum Number of Bits to Encode This Color	Relative Frequency of the Color in the File	Product of Columns 3 and 4
black	100	1.356	0.391	0.530
white	100	1.356	0.391	0.530
yellow	20	3.678	0.078	0.287
orange	5	5.678	0.020	0.111
red	5	5.678	0.020	0.111
purple	3	6.415	0.012	0.075
blue	20	3.678	0.078	0.287
green	3	6.415	0.012	0.075

2.006

Result of the Shannon-Fano algorithm applied to compression



Color	Frequency	Code
black	100	00
white	100	10
yellow	20	010
orange	5	0110
red	5	1110
purple	3	0111
blue	20	110
green	3	1111

What are the problems?

- Each symbol must be treated individually
- Each symbol has its own code
- Code must be represented in an integral number of bits

Arithmetic encoding - 1

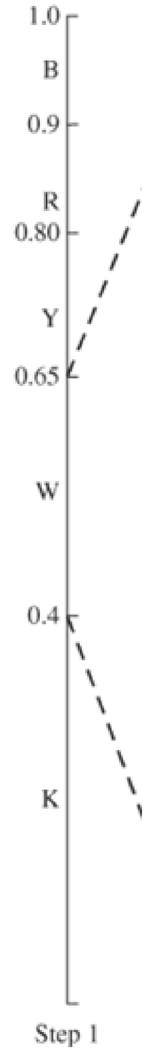
- Also beginning with a list of the symbols in the input file and their frequency of occurrence
- Example

  A file contains 100 pixels in five colors

Color	Frequency Out of Total Number of Pixels in File	Probability Interval Assigned to Symbol
black (K)	40/100 = 0.4	0 – 0.4
white (W)	25/100 = 0.25	0.4 – 0.65
yellow (Y)	15/100 = 0.15	0.65 – 0.8
red (R)	10/100 = 0.1	0.8 – 0.9
blue (B)	10/100 = 0.1	0.9 – 1.0

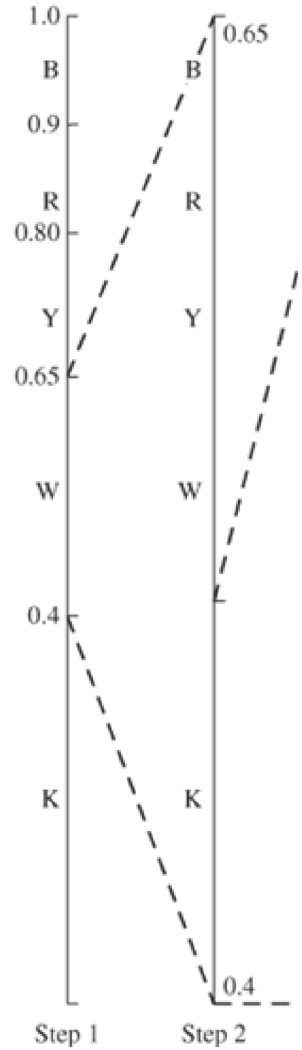
Arithmetic encoding - 2

- Step 1 – W
  interval : 0.4 – 0.65



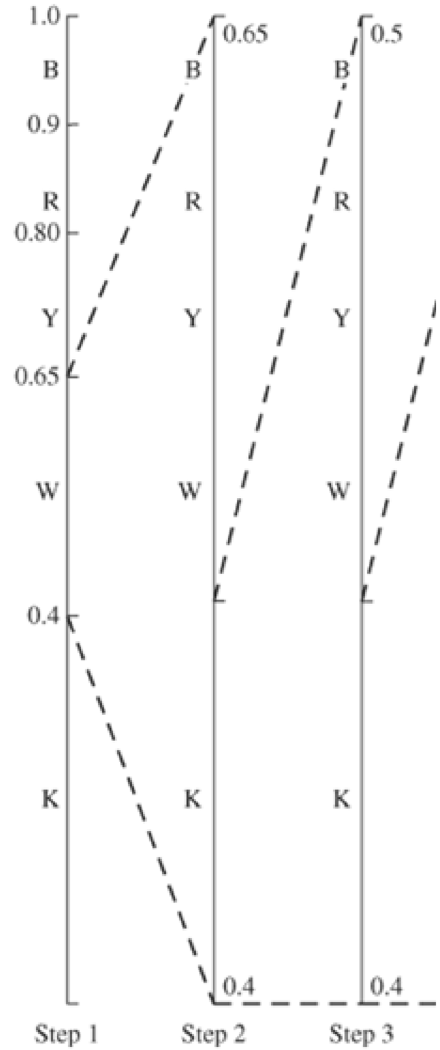
Arithmetic encoding - 3

- Step 1 – W
  interval : 0.4 – 0.65
- Step 2 – K
  0.4+0.25*0.4=0.5
  interval : 0.4 – 0.5



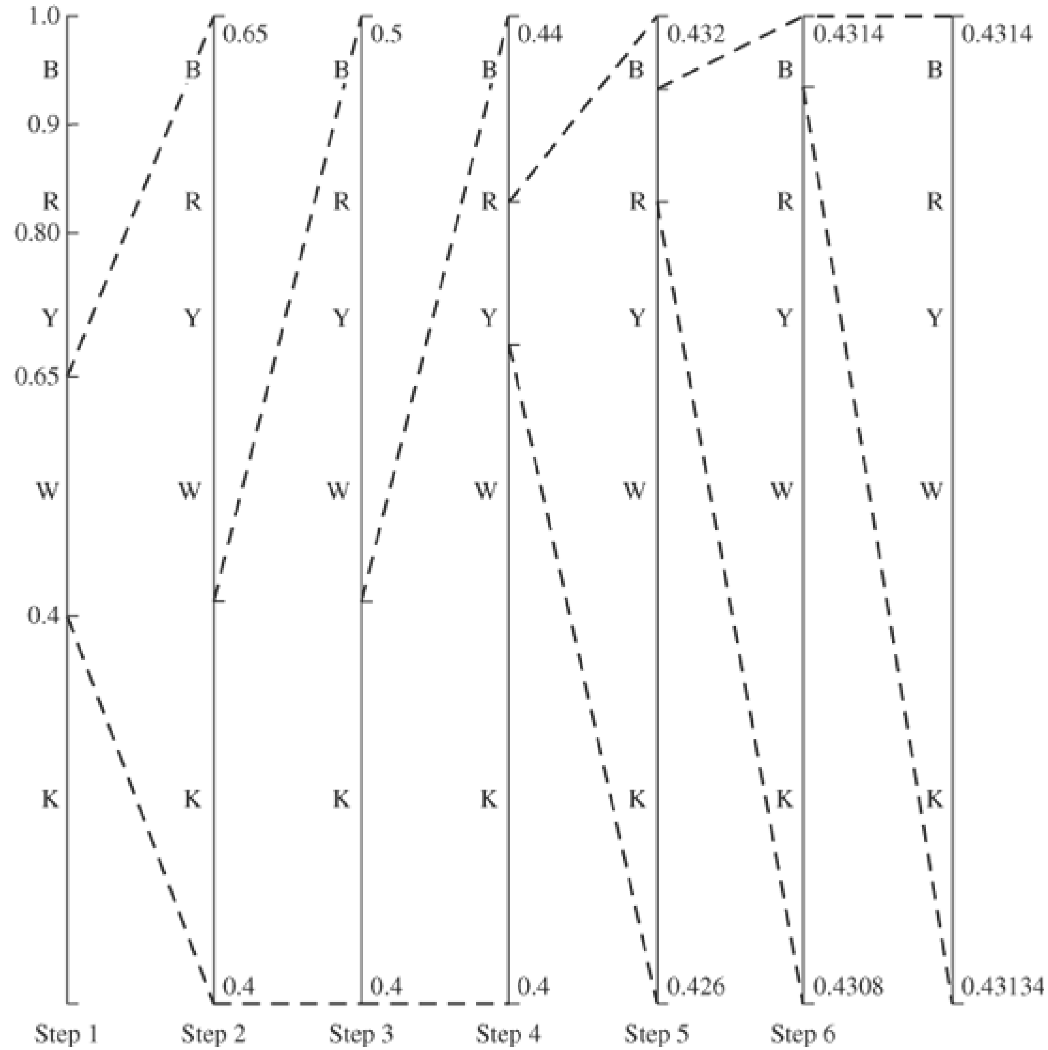
Arithmetic encoding - 4

- Step 1 – W
  interval : 0.4 – 0.65
- Step 2 – K
  0.4+0.25*0.4=0.5
  interval : 0.4 – 0.5
- Step 3 – K
  0.4+0.1*0.4=0.44
  interval: 0.4 – 0.44



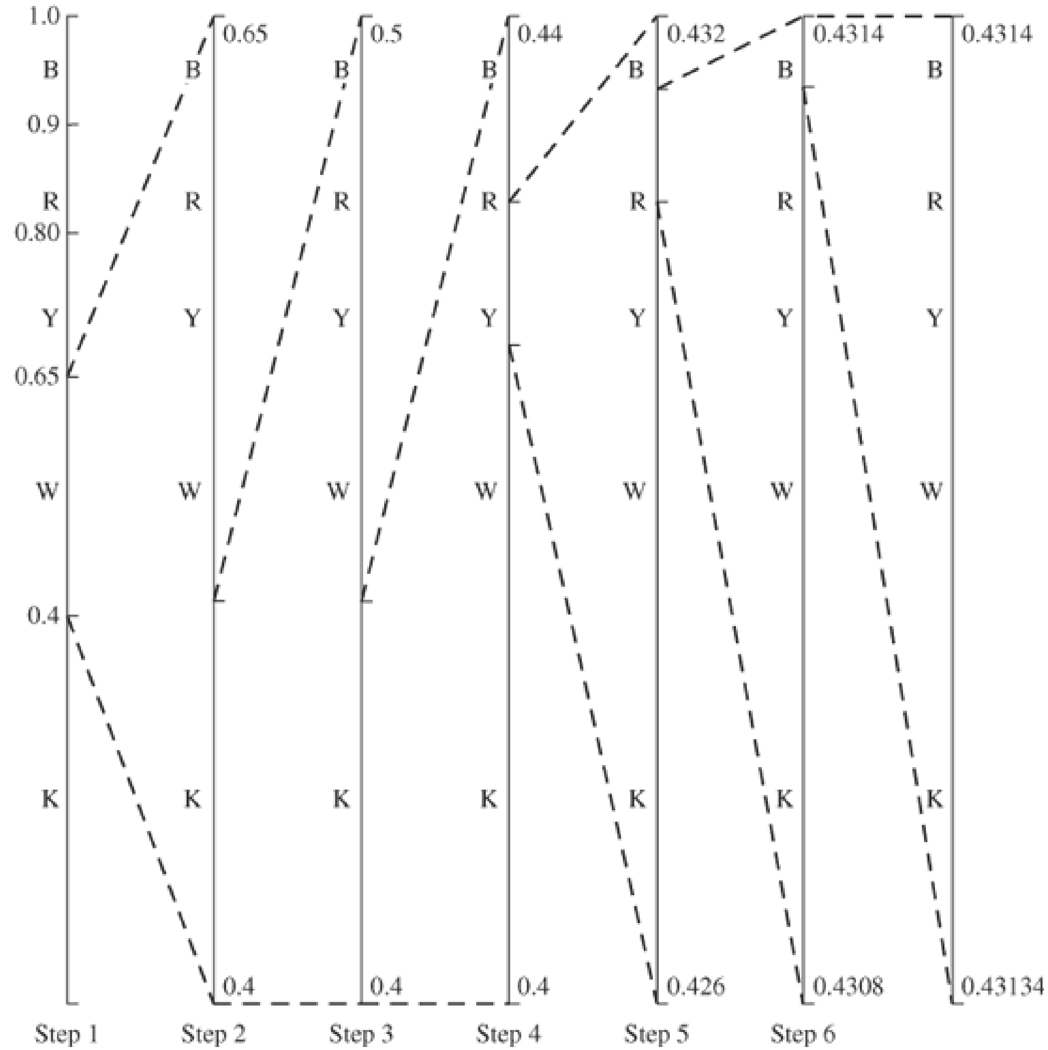
Arithmetic encoding - 5

- Step 1 – W
  interval : 0.4 – 0.65
- Step 2 – K
  0.4+0.25*0.4=0.5
  interval : 0.4 – 0.5
- Step 3 – K
  0.4+0.1*0.4=0.44
  interval: 0.4 – 0.44
- ……



Arithmetic decoding - 1

- Final encoding : 0.43137
  (0.43134+0.4314)/2=0.43137
- 0.43137 fits in the interval assigned to W -> W
- Remove the scaling of W
  (0.43137-0.4)/0.25=0.12548
  -> K
- Remove the scaling of K
  0.12548/0.4=0.3137
  -> K
- ……



Arithmetic decoding - 2

Floating Point Number f, Representing Code	Symbol Whose Probability Interval Surrounds f	Low Value for Symbol's Probability Interval	High Value for Symbol's Probability Interval	Size of Symbol's Probability Interval
0.43137	W	0.4	0.65	0.25
(0.43137 − 0.4)/(0.65 − 0.4) =0.12548	K	0	0.4	0.4
(0.12548 − 0)/(0.4 − 0) =0.3137	K	0	0.4	0.4
(0.3137 − 0)/(0.4 − 0) =0.78425	Y	0.65	0.8	0.15
(0.78425 − 0.65)/(0.8 − 0.65) =0.895	R	0.8	0.9	0.1
(0.895 − 0.8)/(0.9 − 0.8) =0.95	B	0.9	1.0	0.1

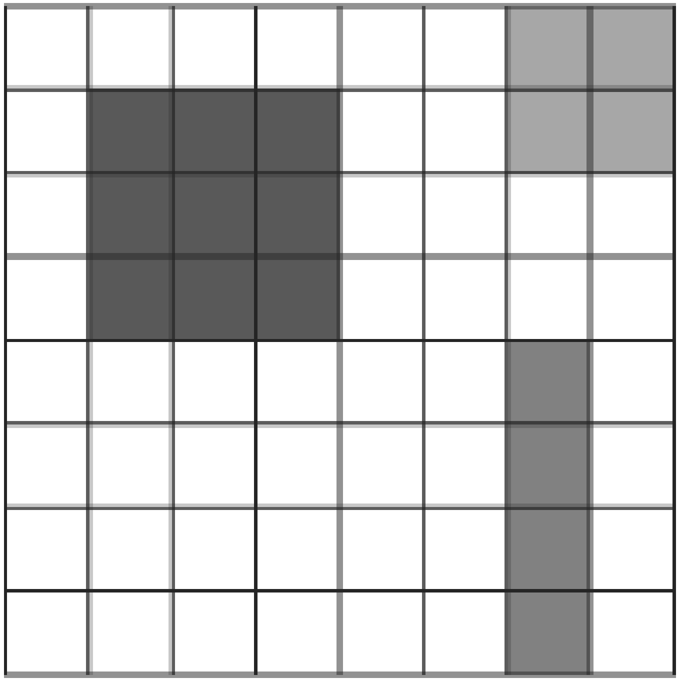
Transform Encoding - 1

- Lossy methods are often based upon ***transform encoding***
- Most commonly used transforms in digital media
  - Discrete cosine transform (DCT)
  - Discrete Fourier transform (DFT)
- No information is lost in the DCT or DFT. When a transform is used, it becomes possible to discard redundant or irrelevant information in later steps, thus reducing the digital file size. This is the lossy part of the process.

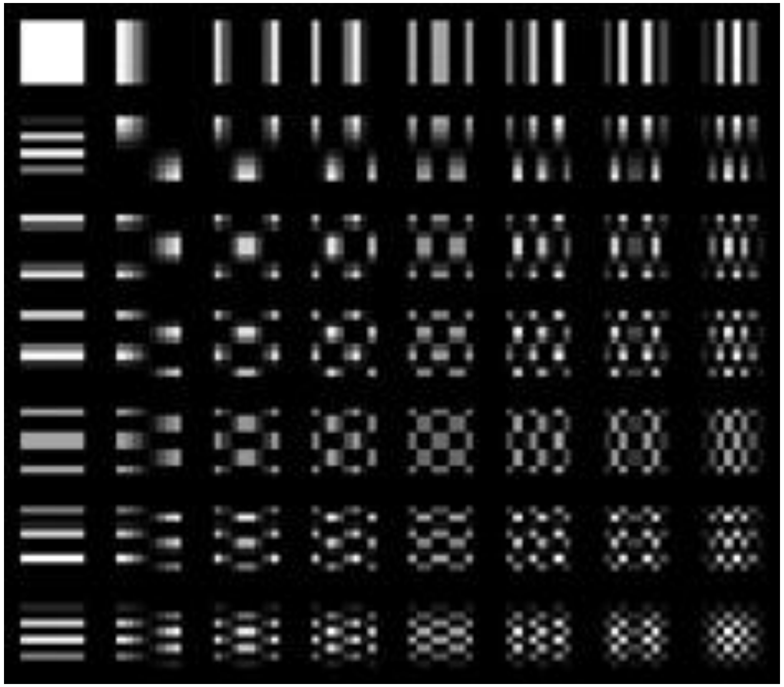
Transform Encoding - 2

- Discrete cosine transform

Origin image (8x8)



Discrete cosine transform

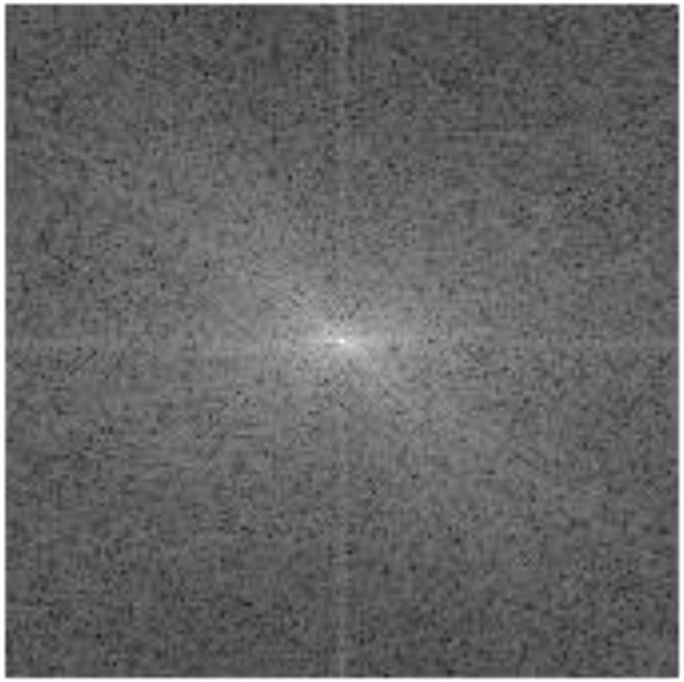


Transform Encoding - 3

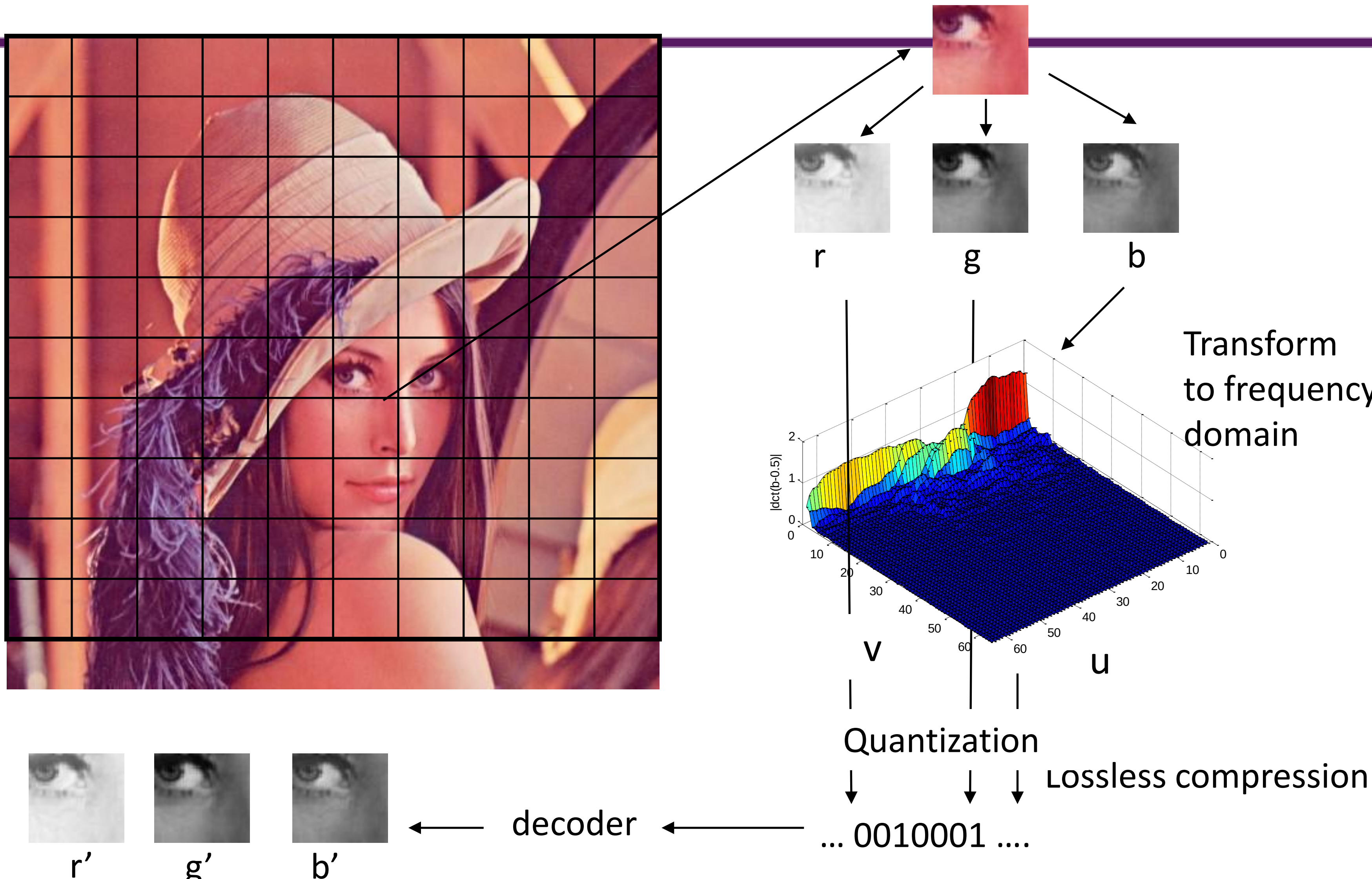
- Discrete Fourier transform

Original image

log(FFT)



“JPEG” Illustration



Standards and Standardization Organizations for Digital Media

- Proprietary standards
- De facto standards
- Official standards
- ITU: International Telecommunication Union
- ISO: International Organizations for Standarization