Basic Image Processing Algorithms

PPKE-ITK, 2015

Lecture 11.

Image and Video Compression

Introduction to Compression

- Compression is the reduction of the number of bits used for the representation of an image or video, while
  - being able to exactly reconstruct the original data (**lossless compression**)
  - maintaining an acceptable quality of the reconstructed data (**lossy compression**)

- Why are the signals compressible?
  - The signals contain **redundancy** (spatial or temporal), they have a structure which can be described in a more compact way.
  - There are parts of the image which are perceptually irrelevant, which can be discarded.

Lossless Compression

Lossless Compression

- Reversible process: the original data can be ***exactly*** reproduced from the compressed data.
- Only limited ***compression ratio*** can be achieved with lossless compression, determined by the ***entropy*** of the source data.

$$\frac{\text{original size}}{\text{compressed size}}$$

Average information per symbol

- There is a tradeoff between:
  - Efficiency (compression ratio)
  - Complexity (required memory, computational power, etc.)
  - Coding Delay (how long does it take to code the signal)

Lossless Compression

- Two main group of lossless coding techniques:
  1. Statistical methods: the statistics of the source is known. (e.g. Huffman coding, Extended Huffman coding)
  2. Universal methods: the statistics of the source is unknown (e.g. Arithmetic methods, Dictionary methods, Adaptive Huffman coding)

Background - Information Theory

- Definitions:
  - **Source**: any information generating process can be viewed as a source that emits random sequence of symbols, from a finite alphabet.

    e.g. 1. natural written languages (in English there are 26 letters + space + numbers + … in the alphabet)

    e.g. 2. 8 bit grayscale image (it has an alphabet with 2^8 symbols)

  - **Discrete Memoryless Source** (DMS): the generated successive symbols are iid.
    - Can be described by its symbols and the associated probabilities.
    - It is the simplest model.
    - Not perfect: in the above examples there is a dependency among the symbols.

Background - Information Theory

- Definitions:
  - **Self Information**:
    - How much information is provided by the emission of a certain symbol?
    - The occurrence of a less probable event provides more information:

$$I(s_i) = \log\left(\frac{1}{p_i}\right) = -\log(p_i)$$

where $S = \{s_1, s_2, \ldots, s_n\}$

$\{p_1, p_2, \ldots, p_n\}$

- Historically the base of the logarithm is 2.
- In case of independent symbols:

$$I(s_1 s_2) = \log\left(\frac{1}{p(s_1 s_2)}\right) = \log\left(\frac{1}{p_1 p_2}\right) = -\log(p_1 p_2) =$$

$$= -\log(p_1) - \log(p_2) = I(s_1) + I(s_2)$$

Background - Information Theory

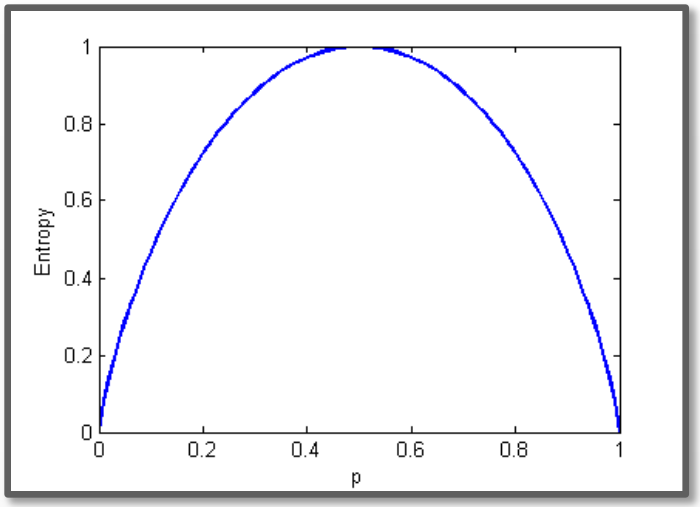
- Definitions:
  - **Entropy:**
    - It is the property of the source not only one symbol.
    - For a DMS it is the average information per symbol:

$$H(S) = \sum_{i=1}^{n} p_i I(s_i) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

    - Simple example for the entropy of a two-symbol alphabet:

$$H = -p\log_2(p) - (1-p)\log_2(1-p)$$

      - The entropy is maximum if the probability of the symbols is equal.
      - The entropy is minimum if one symbol has probability 1, the others 0.



Background - Information Theory

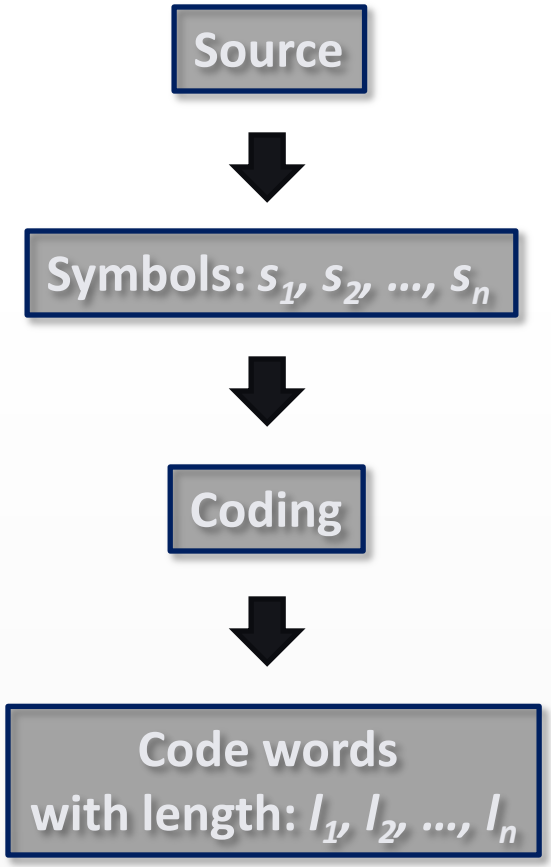
- First order codes:
  - encodes each symbol independently of the others, every symbol has a code word.
- Block codes:
  - group the symbols of the source (***S***) into N length blocks and generate a codeword for each block.
  - It can be regarded as a new source (S_N) that generates symbols from an alphabet with n^N number of symbols (where ***n*** is the number of symbols in ***S***).

$$H(S_N) = N \cdot H(S)$$

- Non-block codes:
  - Arithmetic codes
  - Lempel-Ziv codes

Background - Information Theory

The lossless coding pipeline:



If we assume DMS, then the alphabet and the probability of each symbol define the code.

The length of the code words can be fix (ASCII) or varying (Morse).

The average code word length is the measure of code efficiency:

$$l_{avg} = \sum_{i=1}^{n} l_i p_i$$

Background - Information Theory

- **Shannon's Source Coding Theorem**:

  Let ***S*** be a source with alphabet size ***n*** and entropy ***H(S)*** and let consider coding ***N*** source symbols into one binary codeword (block coding), then for every ***δ***>0 it is possible by choosing the ***N*** large enough, to construct a code with average number of bits per symbol l_{avg} that satisfies the following inequality:

$$H(S) \leq l_{avg} < H(S) + \delta$$

  This means:

  - Entropy is the lower bound of the code efficiency, we cannot beat it
  - but we can come arbitrarily close to it by increasing ***N***

  Increasing ***N*** results larger dictionary and a delay in decoding.
  In general it is not straightforward to calculate entropy (the formula is only for DMS)

Background - Information Theory

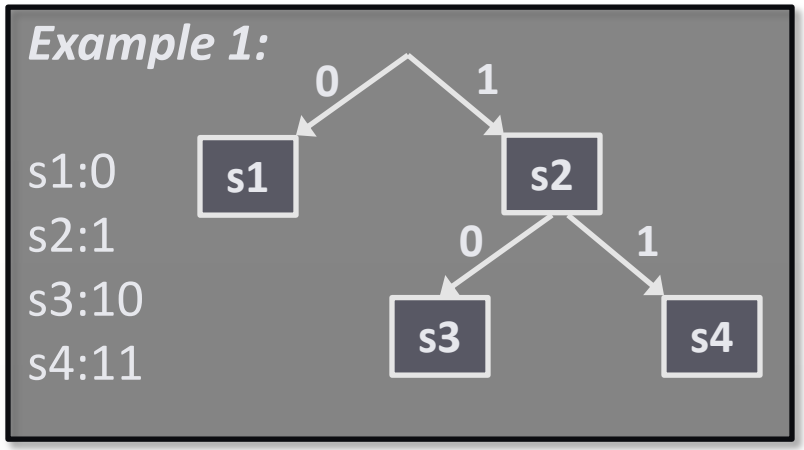
- Examples:
  1. Considering the following sequence of symbols: **1234323456**
     - There are 6 symbols, ***S*** = {1,2,3,4,5,6}, with the following probabilities: {0.1, 0.2, 0.3, 0.2, 0.1, 0.1}
     - Assuming DMS, the entropy is ***H(S)= 2.4464*** bits/symbol
     - But there is a structure in the code and we can use predictor to reveal it: $x(n+1) = x(n) + r_{n,}$ where r_n ***=1111-1-11111***
     - If we code only the residual: S_r={-1,1} with probabilities {0.2, 0.8} and the new entropy is much smaller: $H(S_r)$**= 0.7219** bits/symbol
  2. Considering the following sequence of symbols: **334533334545**
     - P(3)=0.5, P(4)=P(5)=0.25 ⟶ H(S) = 1.5 bits/symbol
     - P(33)=0.5, P(45)=0.5 ⟶ H(S') = 1 bits/symbol
       ⟶ H(S) = 0.5 bits/original symbol

Background - Information Theory

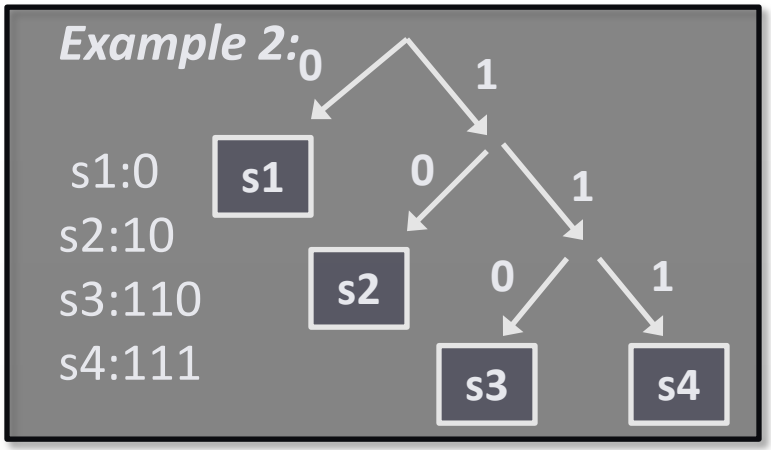
- Definitions:
  - **Uniquely Decodable (UD)**: any finite sequence of code words corresponds to only one message sequence.
    - In general it is hard to tell if a code is UD or not
  - **Prefix Code**: no codeword is a prefix to an other codeword
    - A prefix code is always UD, but a UD code is not necessarily prefix.
    - Prefix codes are easy to design and easy to decode.
    - Every UD code can be converted to a prefix code with same rate.

Background - Information Theory

- **Binary Tree representation**:
  - At each disjunction we can go left or right, adding 0 or 1 to the codeword:



Not prefix code



Prefix code

  - For a prefix code each node is a leaf node.

Huffman Coding

- Derived by David Huffman in 1951.
- Variable length, prefix code.
- Based on the Morse principle: the more common symbols have shorter code word.

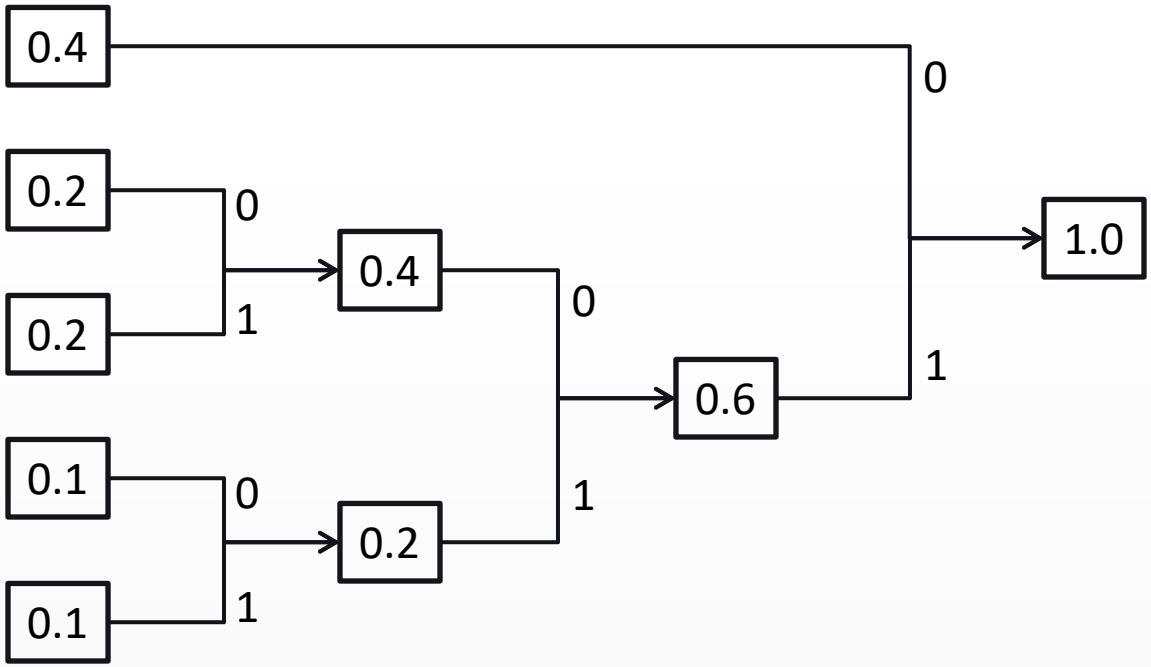
- Algorithm:
  1. Sort the symbols according to their probability.
  2. Combine the two least probable symbol to a composite symbol with probability equal the sum of the probabilities of its components.
  3. Repeat the first two steps until only one composite symbol remains. (if the alphabet has *n* symbols, the algorithm will have *n-1* steps)

  The output of the algorithm is a binary tree that describes the code.

- The result is not unique.

Huffman Coding

- Example:
  - 5 symbols with the following probabilities: 0.2, 0.4, 0.2, 0.1, 0.1



Huffman Coding

- Extended Huffman Coding: uses block code
- Why is it produces a more efficient code?
  - Huffman code bound:

$$H(S) \le l_{avg} < H(S) + 1$$

  - If we use block coding with *N* length blocks:

$$H(S_N) = N \cdot H(S) \quad \text{and} \quad l_{avg_N} = N \cdot l_{avg}$$

$$H(S_N) \le l_{avg_N} < H(S_N) + 1$$

$$N \cdot H(S) \le N \cdot l_{avg} < N \cdot H(S) + 1$$

- The drawback:
  - #codewords increases exponentially (storage, computation, delay)

Arithmetic Coding

- Coding sequences of symbols or blocks together is more efficient than generating codewords for each symbol separately.
- Problem with Huffman coding:
  - The alphabet can get huge easily: if we want to assign codeword to an N-length block of symbols, we have to assign codewords to all possible N-length blocks.
- In arithmetic coding a unique **tag** is generated for a sequence of symbols, then the tag is coded into a binary code.
- One possible source of tags are the numbers between 0 and 1.
- Since there are infinitely many real numbers in this interval, we can generate a tag for arbitrary long sequence of symbols.

Arithmetic Coding

- **Tag generation**:
  - We want to map a sequence of symbols onto an interval
  - The cumulative distribution function will be used.
  - Let *S* be an alphabet with *n* symbols and known probabilities:

$$S = \{s_1, s_2, \ldots, s_n\}; \qquad P(s_i); \qquad \sum_{i=1}^{n} P(s_i) = 1$$

  - Let *X* be a random variable that maps the event of the appearance of a symbol onto the real line:

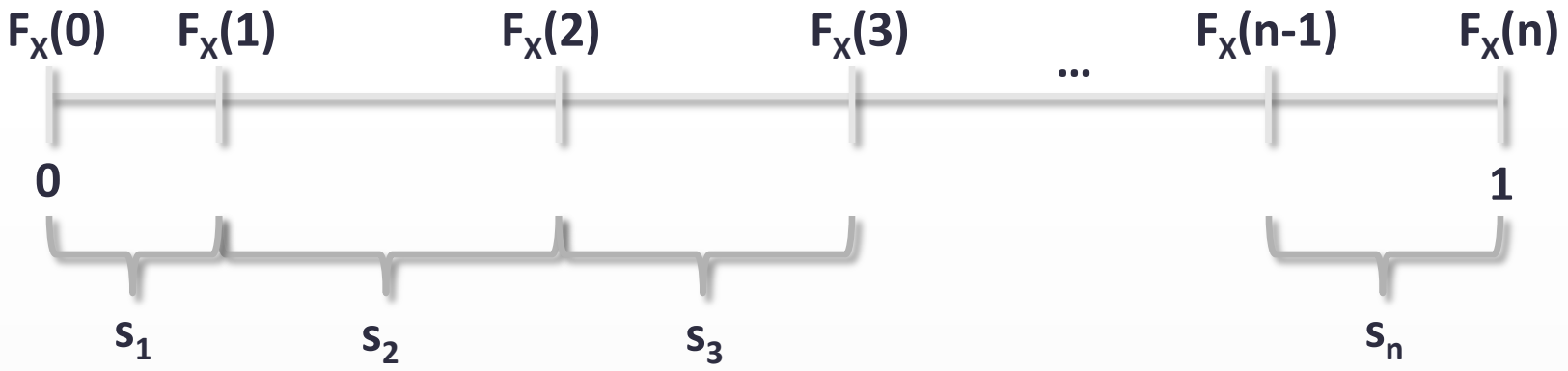
$$X(s_i) = i; \qquad P(X = i) = P(s_i)$$

  - The cumulative distribution function is the following:

$$F_X(k) = \sum_{i=1}^{k} P(s_i)$$

Arithmetic Coding

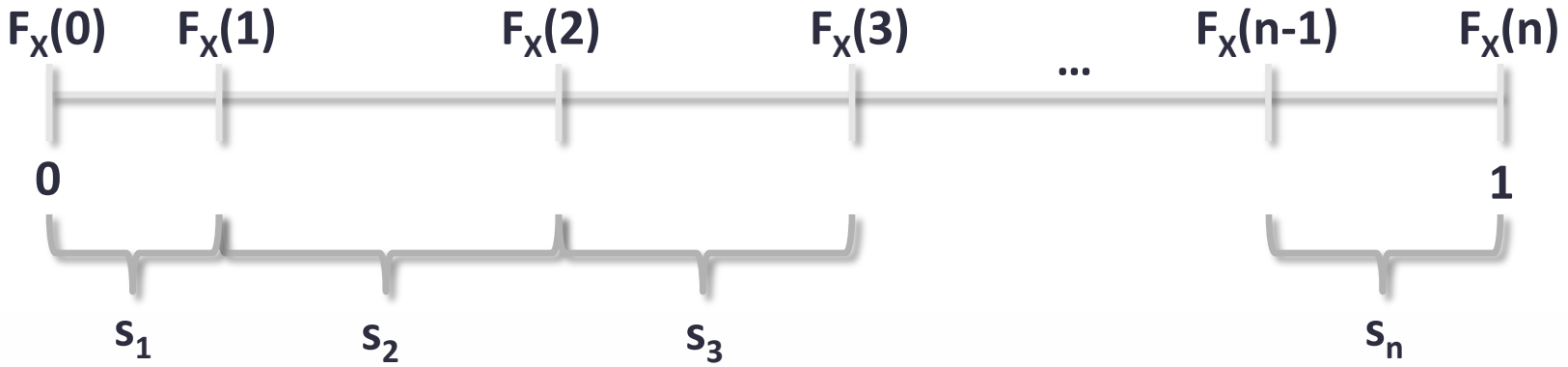
- **Tag generation**:
  - We map a sequence of symbols onto an interval using the cumulative distribution function:



  - A sequence of symbols will be described by a tag (number between 0 and 1) and the number of symbols coded by the tag.
  - The tag itself is coded.

Arithmetic Coding

- **Tag generation**:



1. We can find the first symbol by placing the number on the 0-1 interval and check which symbol's interval it is placed on.
2. Proportionally map the division of the original intervals to the selected interval and check again which interval the tag is on.
3. Repeat the first 2 steps as many times as many symbols are coded by the tag.

Arithmetic Coding

- Comparison with Huffman coding:

$$H(S) \leq R_{Hufmann} < H(S) + \frac{1}{N}$$

$$H(S) \leq R_{AC} < H(S) + \frac{2}{N}$$

  - AC has higher upper bound
  - As the number of the symbols (coded by one tag) increases, the precision of the tag has to be increased too.
  - AC does not suffer from the exponentially increasing alphabet size.
  - In practice better rates can be achieved with AC.

Arithmetic Coding

- Arithmetic coding with progressive transmission is used in JBIG (**J**oint **B**i-level **I**mage processing **G**roup).
- JBIG is standard for bi-level image transmission.
- Arithmetic coding is used taking the local (past) neighborhood statistics into account.
- Progressive transmission:
  - First a low resolution version of the image is transmitted
  - Then if the higher resolution image is required, it can be coded assuming that the low res image is available at the decoding side.

Dictionary Coding

- In many applications there are ***frequently repeated patterns*** emitted by the source.
- It can be efficient to create a list (a dictionary) of the most frequent patterns, so they can be encoded by their address in the dictionary.

- The source is split into two parts:
  - Frequently appearing patterns (coded by the dictionary address)
  - Infrequently appearing patterns (coded with a less efficient technique)

- The dictionary can be static or adaptive.
- Most of the adaptive techniques are based on the 1977 and 1978 papers of Ziv and Lempel.

Predicitve Coding

- In predictive coding we use the „past” of the signal to predict the „present”.
- If we use the same prediction model at encoding and decoding, only the prediction error (the unpredictable part) needs to be coded:

Reconstructed signal = Prediction + Prediction Error

- The prediction error can be coded with some of the lossless compression methods.
- Predictive coding is used in **JPEG-LS**: (Joint Photographic Experts Group)

Predicitve Coding

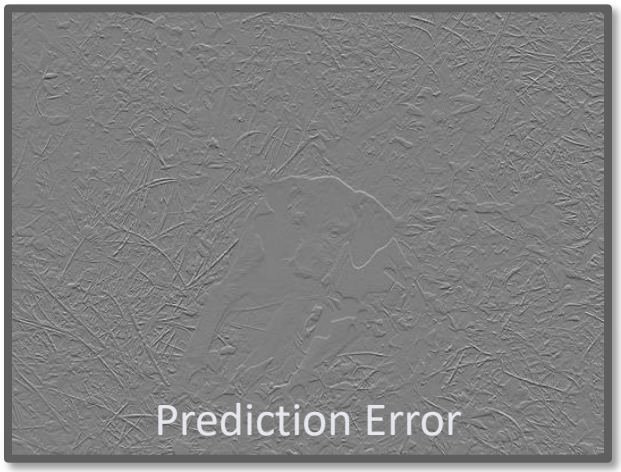
- Why is it better to encode the prediction error instead of the original signal?

Simple Prediction Model:

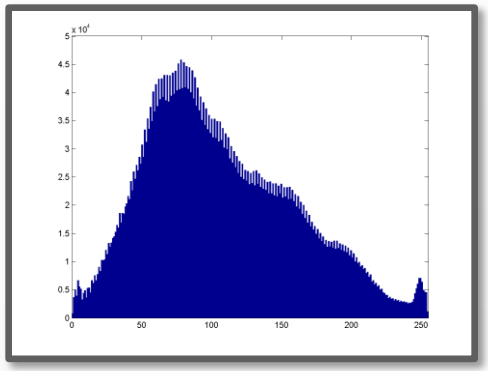
$$f(x) = f(x-1)$$

Prediction Error:

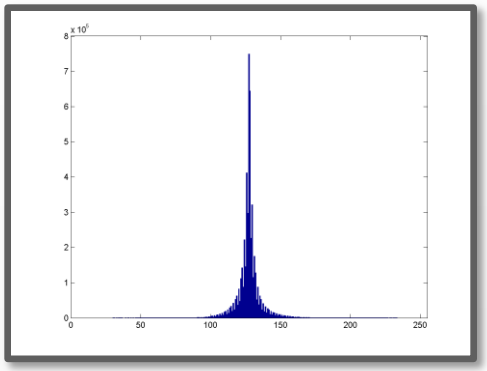
$$e(x) = f(x) - f(x-1)$$



Histogram of the original image and the prediction error:



From the histograms we can see that the entropy of the original image is higher than the entropy of the prediction error.



Lossy Compression

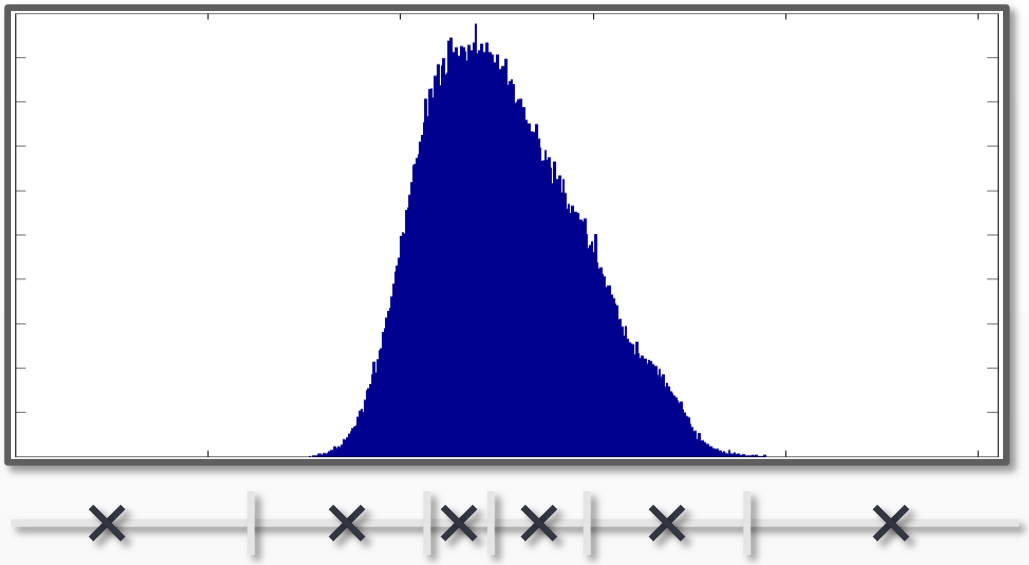
Lossy Compression

- Main steps of a lossy data compression systems:
  1. Redundancy Removal:
     - Predictor or Transformation
  2. Entropy Reduction:
     - Scalar quantization
     - Vector quantization
  3. Lossless Coding:
     - Huffman coding
     - Arithmetic coding
     - LZ coding techniques

Scalar Quantization

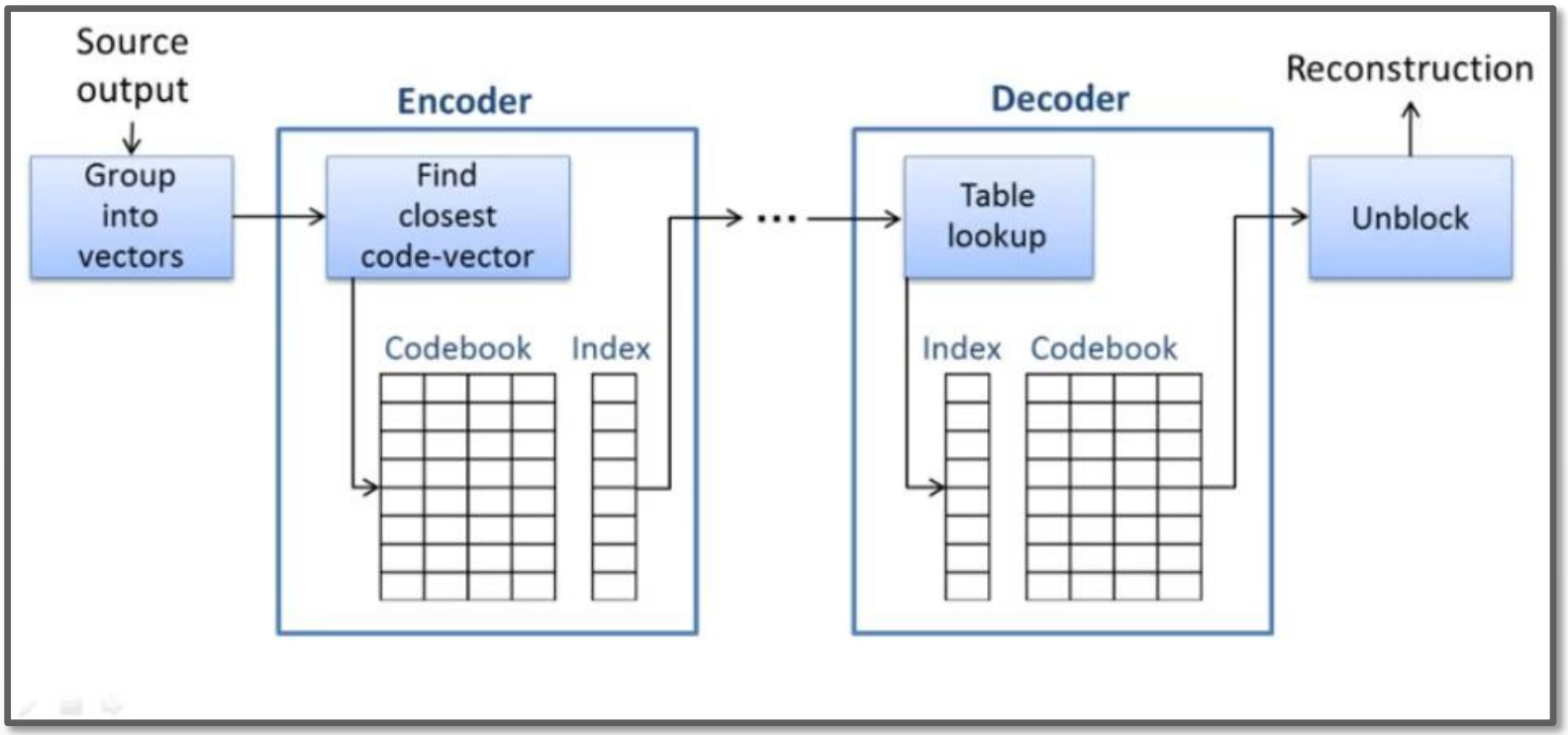
- Uniform quantization:

  - Irreversible process: all the values in the same interval will receive the same quantized value.
- Non-Uniform quantization:
  - The overall error can be reduced by using non-uniform quantizer.
  - The region with more values, can be quantized with higher frequencies.



Vector Quantization

- The input image is divided into small blocks, which are coded using a look-up table:



- The parameters of the codebook defines the compression rate.

Source: Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos

Vector Quantization

- The parameters of the codebook defines the compression rate:
  - If we use ***n*** by ***n*** block size and each pixel is represented by ***b*** bites, and the length of the codebook is $L=2^l$, which means ***l*** bits are needed to code an entry of the codebook, then…

$$rate = \frac{n \cdot n \cdot b}{l}$$

- Vector quantization is more effective than scalar quantization, as it can **take into account the correlation in the data**.

- How can we design an appropriate codebook?

Vector Quantization

- Generalized Lloyd Algorithm
  - With a set of training images it finds a locally optimal codebook:

1. Start with an ***n*** sized random initial codebook.
2. Partition the training vector set using the current codebook by assigning each training vector to the nearest vector in the codebook.
3. Calculate the centroid of each cluster. These centroids will form the new codebook.
4. Repeat step 2 and 3 until the distance between the old and new codebook vectors are below a predefined threshold.

- This algorithm will only find local optimum.
- The final result will be sensitive to the initialization of the codebook.

Transform Coding

- Very popular approach, it is part of most of the current image and video coding standards (JPEG, H.26X, MPEGX)
- Basic idea is to decorrelate the data with a suitable transformation, so that the transformation coefficients will describe the image perfectly.
- The transformation..
  - has to decorrelate the data
  - has to compact the energy of the image
  - has to have image independent basis
  - has to have a fast implementation
- We do a coarse or fine quantization of the transformation coefficients based on their significance (their variance, or their contribution to the total energy of the image).

Transform Coding

- Encoding:

- Decoding:

Transform Coding

Linear transformations

- ***Karhunen-Loeve Transformation:***
  - Statistically optimal
  - Basis functions are image dependent
- ***Discrete Cosine Transform:***
  - Close to KLT for typical images
  - Basis functions are image independent
  - Efficient implementation exist
  - Wildly used in image/video compression standards

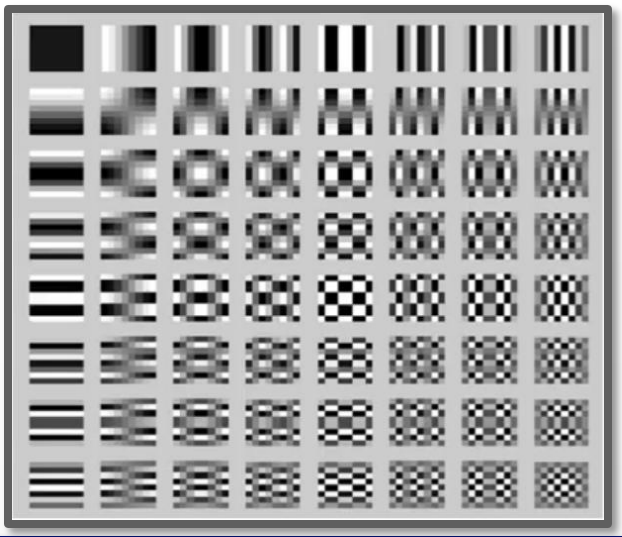
Transform Coding

- Given an *N* by *N* orthonormal matrix set:
  Then any *N* by *N* image can be represented as follows:

$$f = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \cdot \varphi^{(u,v)}$$

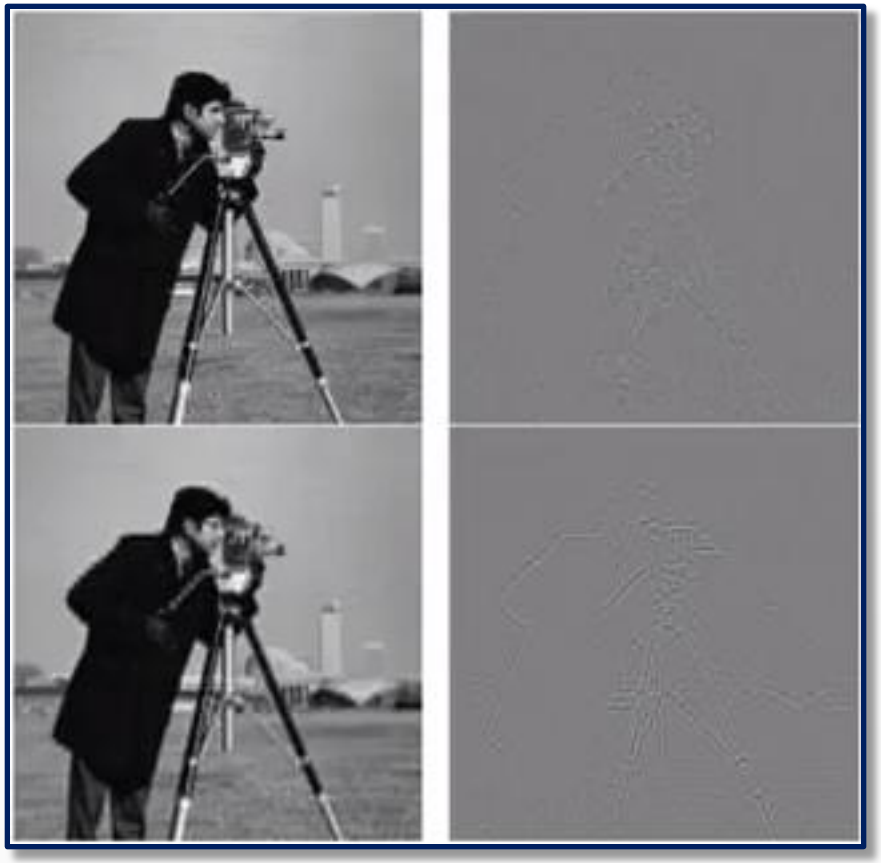
where $F(u,v) = f \cdot \varphi^{(u,v)}$ $\quad u = 0,..,N-1$ $\quad v = 0,..,N-1$

- The 8 by 8 basis matrices for DCT:
  - In the first raw we have a cosine function with increasing horizontal frequency
  - In the first column we have a cosine function with increasing vertical frequency



Transform Coding

Original

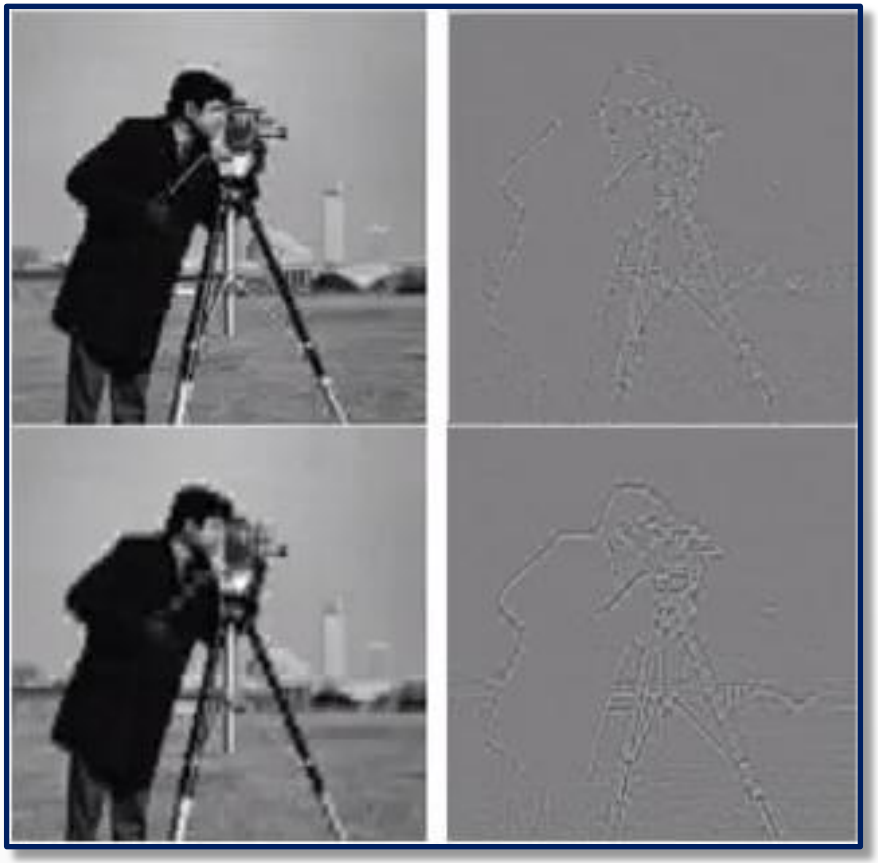


Compressed with 8 coefficients

Source: Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos

Transform Coding

Original

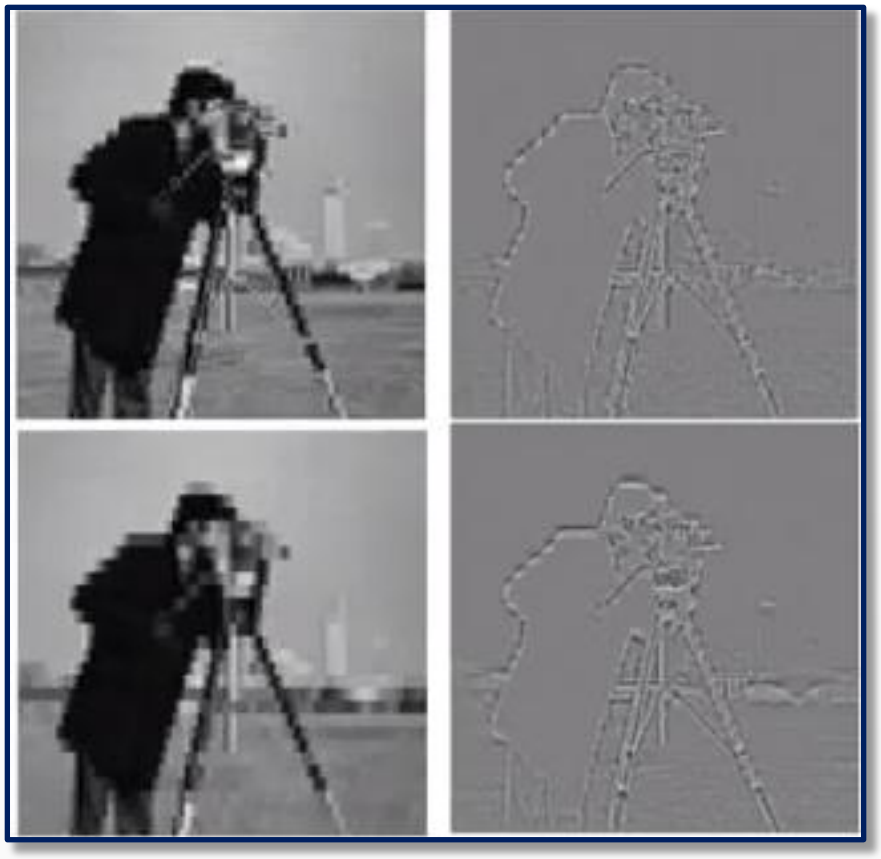


Compressed with 4 coefficients

Source: Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos

Transform Coding

Original



Compressed with 2 coefficients

Source: Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos

JPEG

- Joint Photographic Experts Group:
  - International standard since 1991.
  - Capable of compressing continuous-tone still images (grayscale and color images) with ratio 10-50
- The algorithm:
  - Uses DCT on 8x8 blocks:
    - The blocks' grey level is shifted by -128 to the range [-128, 127].
    - The first coefficient is called DC the rest of the coefficients AC coefficient.
  - The DC coefficients of the blocks are quantized, then coded differentially.
  - The AC coefficients are first quantized, vectorized by zig-zag scan and then entropy coded.
  - The quantizer is uniform, using quantization tables with different step sizes for the different frequencies (in general higher step sizes for the higher frequency coefficients).

JPEG

1.5 bpp

0.5 bpp

0.2 bpp

Main Sources and Further Readings

- Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos
- Khalid Sayood „Introduction to Data Compression”, 3rd edition, Elsevier, 2006
  http://rahilshaikh.weebly.com/uploads/1/1/6/3/11635894/data_compression.pdf